
Feature-Driven Framework for Interpretable Detection and Analysis of Android Malware Through Network and Application Behaviors

Chao Tan¹ and Xinlu Li^{2,*}

¹*College of Information Science and Engineering, Liuzhou Institute of Technology,
Liuzhou City, Guangxi Province 545006, China*

²*College of Civil Engineering and Architecture, Liuzhou Institute of Technology,
Liuzhou City, Guangxi Province 545006, China*

E-mail: xinlu_tech@163.com

**Corresponding Author*

Received 21 April 2026; Accepted 10 July 2026

Abstract

This paper presents a complete framework for feature-driven Android malware detection that combines predictive modeling with explainable artificial intelligence to improve classification accuracy, interpretability, and operational dependability. The system initiates by examining network-flow metrics, including Source Port (SP), Initial Window Bytes Forward (IW), packet rates, and user activity indicators, derived from a publicly accessible dataset including 355,630 cases across four classifications: Adware, Scareware, SMS malware, and benign. Feature preprocessing utilizes the Variance Inflation Factor (VIF) analysis to eliminate multicollinearity to discern the most important variables, hence assuring a non-redundant and highly relevant feature collection. Subsequently, Gradient Boosting Classifier (XGBC) and Decision Tree Classifier (DTC) are augmented with metaheuristic optimization algorithms, Tailor Optimization Algorithm (TOA) and Spotted Deer Optimization

Journal of Cyber Security and Mobility, Vol. 15.5, 1311–1336.

doi: 10.13052/jcsm2245-1439.1557

© 2026 River Publishers

Algorithm (SDOA), to boost convergence, stability, and generalization. Model interpretability is attained by Delta-XAI, which identifies SP, IW, and User Interaction/Write operations as the principal features, collectively representing approximately 80% of the explanatory power of the top features. Assessment by five-fold cross-validation indicates that the XGBC+TOA (XGTO) configuration attains an overall accuracy of 98.0%, exhibiting excellent precision and Matthews Correlation Coefficient (MCC), surpassing all other models. This framework combines feature-centric selection, enhanced learning, and explainable AI to deliver a dependable, interpretable, and pragmatic approach for identifying Android malware, thereby connecting data-driven analysis with practical cybersecurity implementations.

Keywords: Cybersecurity analytics, Delta-XAI sensitivity, network traffic analysis, metaheuristic algorithms, Android malware detection.

1 Introduction

With a score of nearly 87%, Android has become the most influential operating system in the portable world [1], becoming one of the main targets for large-scale malware campaigns and cybercriminals. In today's world, the system's medium is made through the popular use of various third-party tools, thus contributing to the quality improvement in portable technology through numerous innovations. However, the same medium has allowed the perpetrators to double their criminal works as they produce and distribute malicious software at a quicker pace than before [2]. The mounting number of applications has been a factor in the explosion of the malicious code rate, which most Android-related software has experienced in recent years. As the report from Statista shows, in terms of the number of Android applications, the growth has been very significant because it rose from 2.8 million in 2018 to over 3.4 million in 2021 [3, 4].

Besides that, the malware creators usually utilize the infected systems as botnets to secretly acquire sensitive data or to have remote control over the ill-fated machines in such a way that the users are unaware of what is happening. By this means, they might be downloading a pirated version of popular gaming software or some free utility software, whereas in reality, they are being targeted [5]. Java is a programming language that allows various code obfuscation techniques, and the dynamic code loading feature is the primary language used to build Android apps [6]. The rapid development and variety of Android applications have also created potential for harmful

activities. Malware components can exploit security holes in a system to gain entry, steal sensitive data, and even take control of the system, while simultaneously replicating themselves within innocent programs [7, 8]. The fluidity of malware necessitates identifying fundamental behaviors and networking features that indicate malicious intentions and threats. Therefore, implementing fire protection strategies based on feature analysis and monitoring system functioning is indispensable for risk alleviation and user information safety on Android systems [9, 10].

With the rise in the number of malicious apps and the threat to user security, Android malware detection has become a highly sought-after academic research area. Initially, methods were mainly reliant on signature-based detection techniques, which are good tools against already known threats but less adaptable to newly created or encrypted malware. The limitation has resulted in the adoption of machine learning and deep learning techniques for better detection capabilities.

El Hami et al. [11] employed a Random Forest classifier for Android malware detection, which outperformed traditional methods with an average accuracy of 98.47%, sensitivity of 98.60%, and F-score of 98.60% in all experiments. Similarly, Baldangombo et al. [12] developed a static analysis framework that utilized four machine learning methods and five feature selection techniques to analyze permissions and API calls extracted from the Android Package Kit (APK) files, thereby achieving an overall accuracy of 96%. Anand et al. [13] stressed the importance of feature derivation from different code representations by presenting a hybrid approach that leverages hex code features from JAR files for the detection of obfuscated and non-obfuscated malware. Furthermore, feature selection has been improved by using metaheuristic methods.

By using a genetic algorithm to choose the best characteristics, Yerima et al. [14] increased detection accuracy to 92.56% while lowering the computing cost. Similar to this, Hamad et al. [15] emphasized the value of merging numerous classifiers for increased resilience by utilizing ensemble learning with creative static feature sets, attaining 96.24% accuracy and a low 0.3% false-positive rate. Additionally, deep learning approaches have shown promise in addressing evolving infection trends. By capturing both spatial and sequential patterns in application activity, Kotha et al.'s hybrid CNN-RNN model [16] achieved 89.14% accuracy. Bansal et al. [17] employed a light gradient boosting machine for dynamic malware detection, achieving an F1-score of 98.787%, accuracy of 98.008%, and recall of 99.194%, and thus outperforming a number of their previous methods. The studies of Manzil and

Naik [18] and Al-Kaaf et al. [19] deeply investigated and revealed a growing trend of machine learning usage by dividing Android malware detection into signature-based, behavior-based, static, dynamic, and hybrid methods. Singh and Khan [20] also agreed with this point by stressing the necessity of utilizing intelligent models for the examination of APK features as a means of threat prevention. When taken as a whole, this research shows how sophisticated, adaptive machine-learning frameworks have evolved beyond conventional detection, laying the groundwork for reliable Android malware prediction systems.

Although great progress has been made concerning the detection of Android malware, some research gaps still remain, limiting the effectiveness of malware detection methods. Signature methods, despite great success, are inefficient for unknown, polymorphic, and obfuscated malware and require efficient and adaptive methods. Although methods based on machine learning and deep learning algorithms have increased heuristic classification accuracy, most of these methods involve small sets of static features and ignore dynamic behavior patterns, limiting their ability to generalize effectively over complex network settings. Moreover, ensemble and metaheuristic algorithms used for feature selection techniques have improved accuracy, but most have failed to be interpretable, restricting their ability to explain feature contributions and decision-making processes and providing little useful information and indication for realistic implementation and decision-making purposes. Additionally, there has been limited exploration of combinations of static and dynamic features and the incorporation of explainable AI methods, restricting transparency and limiting development and progress within this research area.

Unlike conventional Android malware detection studies that primarily focus on standalone classifiers or isolated optimization strategies, the proposed framework integrates feature validation, optimizer-assisted ensemble learning, and explainable AI into a unified detection architecture. The contribution of this work lies not in introducing a completely new classifier, but in constructing an interpretable and feature-driven cybersecurity framework that combines multicollinearity-aware feature selection, adaptive metaheuristic optimization, and Delta-XAI-based behavioral analysis for Android malware traffic classification. This integrated approach enables simultaneous improvement in predictive accuracy, learning stability, and model interpretability within complex multi-class malware environments. The main aim of this project is to provide a resilient, interpretable, and high-performing machine learning system for the detection and classification of Android malware across many categories, including Adware, Scareware, SMS-based malware,

and innocuous apps. The work aims to systematically integrate feature-based selection, ensemble learning, and metaheuristic optimization strategies, while maintaining model explainability using contemporary interpretability methodologies. The primary goal is to improve prediction accuracy and generalization by developing classifiers that can proficiently identify intricate patterns in network traffic produced by Android applications. This involves closely examining and confirming the features through the Variance Inflation Factor (VIF) to remove multicollinearity and select the most informative variables for model training. K-fold cross-validation is used to increase the trustworthiness and the stability of the trained models on different data splits; thus, it decreases overfitting and bias. Another goal is to use metaheuristic optimization methods, namely the Tailor Optimization Algorithm (TOA) and Spotted Deer Optimization Algorithm (SDOA), for the fine-tuning of the hyperparameters and the structural changes of both gradient boosting classifiers (XGBC) and decision tree classifiers (DTC). This seeks to expedite convergence, enhance decision boundaries, and bolster model resilience in detecting malware-induced abnormalities. The project also seeks to assess feature contributions through Delta-XAI, a model-agnostic explainability approach, to determine the principal factors influencing categorization judgments. The project aims to quantify feature relevance to allow interpretable predictions, enhance efficient feature engineering, and promote the possible implementation of rule-based detection systems.

2 Methodology

The proposed Android malware detection framework follows a sequential feature-driven learning pipeline that transforms raw network traffic records into interpretable malware predictions. Initially, the `Android_Malware.csv` dataset is collected and preprocessed to remove incomplete, duplicated, and inconsistent records. The extracted traffic-flow features, including packet statistics, port information, flow durations, transport-layer signals, and user interaction indicators, are then normalized and prepared for learning. In the second stage, VIF analysis is performed to evaluate multicollinearity among independent variables. Features with high linear dependency are iteratively removed to preserve only informative and non-redundant attributes. The refined feature space is subsequently divided into training and testing subsets through 5-fold cross-validation to ensure reliable model generalization and minimize overfitting. The validated features are then provided as input to two machine learning classifiers: XGBC and DTC. To further improve

convergence behavior, classification boundaries, and learning stability, TOA and SDOA are integrated for hyperparameter optimization and adaptive structural refinement. This produces four optimizer-enhanced models: XGTO, XGSD, DTTO, and DTSD. During model inference, each classifier predicts one of four classes, namely Android Adware, Android Scareware, Android SMS Malware, or Benign traffic. The classification outputs are subsequently evaluated using Accuracy, Precision, P4-metric, F1-score, and Matthews Correlation Coefficient (MCC). Finally, Delta-XAI sensitivity analysis is applied to quantify feature influence and interpret the reasoning behind prediction decisions by identifying the dominant network behavioral indicators contributing to malware detection.

2.1 Models

To identify harmful Android apps, this study used DTC and XGBC. Using extracted behavioral, API-call, and permission-based characteristics, the model was developed, trained, and evaluated.

2.1.1 Extreme gradient boosting classification

XGBoost is an ensemble-based boosting method that successively optimizes weak decision-tree learners to construct a strong classifier gradually. The model minimizes a regularized objective function that takes prediction loss and model complexity into account at each boosting step. In order to achieve faster convergence and more stability, the model learns by fitting the loss function's negative gradients (first-order derivatives) and Hessians (second-order derivatives). In order to avoid overfitting, regularization hyperparameters are used to regulate the gain of feature splits during tree structure optimization. XGBoost effectively captures non-linear feature interactions frequently seen in malware activity patterns by utilizing shrinkage, column sampling, and sophisticated tree-pruning techniques. Because of this, XGBC works especially well for Android malware classification tasks that have a high-dimensional, sparse feature space [21].

2.1.2 Decision tree classification

A foundational approach for interpretable malware detection is the Decision Tree classifier. The program divides the dataset into homogenous subsets according to attribute-driven split criteria (including information gain or Gini index) using a recursive top-down greedy technique. Decision criteria are represented by internal nodes, results are shown by branches, and class

labels (malicious or benign) are assigned by terminal leaves. Pruning is used after tree building to minimize overfitting and reduce model complexity by eliminating branches that only slightly increase accuracy. DTC may have less generalization and is more susceptible to noise in complicated Android malware datasets, despite being computationally lightweight and highly interpretable. To ensure consistent performance comparison, both models were trained and validated under identical experimental conditions. More comprehensive mathematical formulations and theoretical justifications can be found in [22, 23].

2.2 Optimizers

2.2.1 Tailor optimization algorithm

Inspired by personalized decision-adaptation processes, TOA is a new meta-heuristic approach that customizes solutions repeatedly based on various determinants and dynamic feedback. Unlike static search techniques, TOA uses a tailored search mechanism that is comparable to tailoring in behavioral therapies. This means that each potential solution is adaptively changed based on individual performance characteristics, contextual circumstances, and changing search priorities. In complicated, non-linear problem spaces like Android malware detection, where feature interactions and class borders are highly variable, this adaptive customization facilitates effective exploration.

The very first step of TOA is to locally create a set of random candidate solutions (which can be likened to “individual profiles”). At each iteration, solutions are gauged using a fitness function that embodies detection performance. TOA uses customized refinement tactics based on algorithmic “behavioral states,” performance bottlenecks, and the solution’s progress rather than applying homogeneous operators universally. While suboptimal solutions receive guided modification to lessen departure from intended goal profiles, high-performing solutions are reinforced through exploitation. Hierarchical decision rules and dynamic scaling variables that control mutation, learning intensity, and structural modifications are used to achieve tailoring. The general expression for a core update procedure is:

$$X_i^{t+1} = X_i^t + \alpha \times P_i^t + \beta \times A_i^t \quad (1)$$

where X_i^t is the i -th solution at iteration t , P_i^t represents personalized exploitation parameters, A_i^t denotes adaptive exploration influences, and α and β are dynamic learning coefficients adjusted based on performance progression. Initialization (motivation), guided exploration (pre-adjustment),

intense refining (optimization), and stability (maintenance) are the phases of learning that TOA integrates, which are comparable to motivational and maintenance cycles. Together with multi-level decision customization, this organized progression enables TOA to converge toward ideal malware-classification limits while maintaining generalization capability efficiently. The technique is especially useful in high-dimensional cybersecurity environments where static heuristic assumptions are insufficient and behavioral fingerprints fluctuate greatly. For further technical details and comprehensive methodological explanations, readers are referred to [24].

2.2.2 Spotted deer optimization algorithm

A swarm intelligence technique inspired by nature, SDOA was created by simulating the social and ecological dynamics of sika deer populations. SDOA uses a multilayer behavioral framework that combines migratory dynamics, hierarchical social interactions, energy-constrained decision processes, and adaptive perturbation mechanisms, in contrast to traditional meta-heuristics that mainly rely on simplified collective movement or attraction-repulsion patterns. Due to its well-structured behavioral abstraction, SDOA is able to dynamically balance exploration and exploitation, keep population variety, and avoid early convergence.

First, SDOA does the work of initializing a population of individuals representing the potential solutions that are spread over the constrained search space. Then, the individuals are assigned social status and energy attributes, which determine the way they will behave during the optimization process. Seasonal migration is the global search operation, while fall migration is used to intensify the concentrated area around potential locations, and spring migration is used to facilitate wide-range exploratory displacement. In addition, burst migration will be activated if stagnation of convergence is detected, thus allowing for fast redispersal, as they determine population diversity indicators. Dominance hierarchies are modeled by social status adaptation, in which lower-status people adopt adaptive following behavior while high-status persons influence movement decisions. Solution competitiveness is governed by the combat-energy mechanism, which permits competitive position updates while sufficient energy reserves are available and enforces restrained mobility when energy is exhausted. In order to enable fine-grained local search and avoid stagnation in misleading areas of the fitness landscape, learning perturbation provides stochastic refinement around elite solutions. Additionally, pheromone-pressure control offers a population-wide guiding signal that decays to preserve adaptive search

flexibility while progressively converging toward high-fitness zones. Overall, SDOA achieves strong performance in high-dimensional and multimodal optimization domains, such as engineering design, classification parameter tuning, and cybersecurity behavior modeling, thanks to the combination of migration, hierarchy, energy dynamics, and pheromone-guided perturbation. Compared to traditional swarm optimizers, the method provides improved convergence stability, robust global search capability, and prolonged exploration behavior. More specific details and equations are given in [25] for full mathematical formulations, operator definitions, and convergence studies.

2.3 Feature Validation

2.3.1 Variance inflation factor

VIF is a statistical diagnostic tool utilized to identify and address multicollinearity among variables before model building. Multicollinearity arises when independent variables demonstrate significant linear dependency, resulting in exaggerated parameter variances, unstable coefficient estimations, and incorrect model interpretations. VIF quantifies this effect by comparing the variance of a coefficient in a multivariate context with that of a variance from an uncorrelated situation. High VIF values indicate high collinearity, with engineering and data-centric modeling scenarios usually considering thresholds above 5 as problematic. The present work uses a feature iterative removal technique based on VIF. Initially, VIF scores are computed for all variables, and the variable with the highest score above 5 is removed. The VIF values are then updated, and the process continues until all the remaining predictors have a $VIF < 5$. This staged elimination strengthens the model, lowers the chances of duplication, and ensures that the kept features are still able to uniquely contribute to the prediction performance [26, 27].

2.3.2 Cross-validation

To calculate the efficiency of both the performance and the ability of generalization of the proposed model, a cross-validation of a 5-fold type is used for model training. Through this mechanism, overfitting and biased calculation of model performances on certain instances are avoided. In this approach, the available dataset is divided into five equal parts, or folds, and then four of these parts are used for model training, while the remaining one is applied strictly for validation purposes only. Consequently, all five parts are used as validation sets only once and four times as model training

sets, providing average values of model performances, calculated based on all experiments, and concerning accuracy, precision, recall, and F1 measures. As a result, cross-validation of a 5-fold type significantly increases the reliability of model generalization, especially concerning Android malware threats' detection [28].

2.3.3 Δ -XAI (Delta Explainable Artificial Intelligence)

In this regard, a sensitivity-focused interpretability tool, namely Delta-XAI, has been proposed for quantifying prediction sensitivity based on input feature changes. Unlike overall feature attribution methods like Eli5, Delta-XAI is more focused on its local interpretability capability, based on feature modulation and corresponding changes observed during prediction tasks, as indicated by Δ . Delta-XAI is capable of explaining the complex non-linear relationship between variables and stressing their directionality effect on prediction, thus aiding researchers in determining prominent feature effects and model interpretability through visualization of corresponding Δ changes [29, 30].

3 Performance Evaluators

The evaluation of classification models for Android malware necessitates a comprehensive array of criteria to ascertain their reliability, accuracy, and applicability. Accuracy serves as a metric for the proportion of properly categorized examples relative to the total number of occurrences, evaluating the model's overall accuracy and dependability. Precision quantifies the proportion of accurately categorized cases among all instances identified as positive, evaluating the model's capacity to minimize false positives and erroneous alerts. The P4 metric, an enhanced evaluation metric, properly assesses model prediction capacity by highlighting its balancing skill on varied malware classification tasks as a whole. The F1-score computes the harmonic mean of sensitivity and precision, offering a reliable evaluation of model correctness on unbalanced datasets and assessing model dependability and validity across various classification tasks. The final metric is MCC, which evaluates the accuracy of model predictions across various malware classification tasks by considering both true and false classification instances collectively. It is particularly applicable to imbalanced classification tasks due to its effectiveness in calculating the correlation between actual and predicted instances. The equations relating to each of these metrics can be articulated

as follows:

$$Ac = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

$$Pr = \frac{TP}{TP + FP} \quad (3)$$

$$Re = TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \quad (4)$$

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (5)$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (6)$$

$$P_4 = \frac{Ac + Pr + Re + F1}{4} \quad (7)$$

4 Data Description

This research employs the Android malware detection dataset from Kaggle to develop machine learning-based detection algorithms. The dataset, named *Android_Malware.csv* and licensed under the GNU Affero General Public License 3.0, consists of 355,630 instances, each representing a single network traffic record from an Android app. The dataset has 85 attributes and four target classes: *Android_Adware* (147,443 instances), *Android_Scareware* (117,082 instances), *Android_SMS_Malware* (67,397 instances), and *Benign* (23,708 cases), which implies a very imbalanced multi-class classification problem. To minimize the potential bias caused by class imbalance, stratified 5-fold cross-validation was employed during model training and evaluation to preserve the original class distribution within each fold. Furthermore, evaluation was not limited to accuracy alone, since accuracy may produce misleading interpretations under imbalanced classification conditions. Therefore, additional metrics, including Precision, F1-score, P4-metric, and MCC, were incorporated to provide balanced performance assessment across all malware categories. The confusion matrix analysis further confirmed that the proposed models maintained reliable classification capability for minority and majority classes simultaneously, reducing the likelihood of majority-class dominance during prediction.

The dataset is derived from the Canadian Institute of Cybersecurity (CIC), a leading research entity in the field of cybersecurity. The features cover various network-level parameters such as Flow IDs, source and destination IP addresses, port numbers, packet-level statistics, throughput metrics, inter-arrival periods, active and idle durations, and transport-layer control signals (SYN, ACK, FIN, RST, PSH, URG). Additional features describe subflow information, window widths, and up/down ratios, which help in the study of congestion, broadcast, and persistence patterns usually associated with malicious activities. Such extensive network telemetry allows for effective feature extraction through the VIF method and supports the building of models with XGBC and DTC, improved by TOA and SDOA. Evaluation criteria encompass Balanced Accuracy, AUC, P4-metric, Precision, Recall, and F1-score, whilst Δ -XAI-based sensitivity analysis offers elucidative insights into feature influence and model dynamics. All information and data attributes defined above are derived from [31].

4.1 Feature Analysis

Figure 1 highlights the VIF analysis of the derived flow-level features to check for multicollinearity before building a model. VIF calculates how much a particular feature is linearly explained by a group of variables, and when a value exceeds a predetermined threshold ($VIF > 5$ or 10), it reveals that a high degree of collinearity exists in the variables, which may lead to biased model

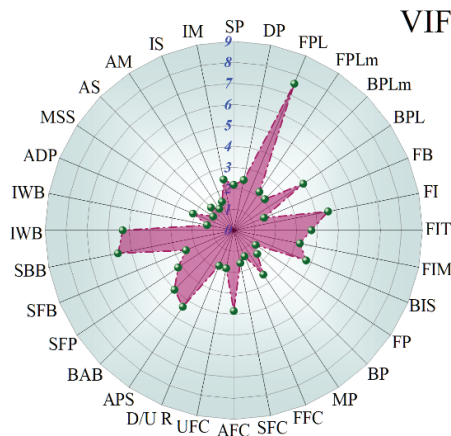


Figure 1 Radar plot representation of the Variance Inflation Factor (VIF) values, used to evaluate the presence and severity of multicollinearity among independent variables.

predictions and interpretations. From Figure 1, it is clear that all variables have a VIF of 1, ensuring that all dominant statistical variables related to flows are orthogonal to each other. This is an indication that learning will take place without redundancy-related biases in the data structure. A small set of metrics related to network flows has a moderate value of VIF, thus revealing an expected dependency of bidirectional packet flows (for instance, bytes sent/received and packet rate variables). In summary, the findings of VIF prove the relevance of the data inputs for an advanced learning process, ensuring that both XGBC and DTC learn within a non-redundant and informative feature space. This outcome directly supports trustworthy model building, enhances interpretability within an explainability-focused analysis, and leads to better generalization in an Android malware detection system.

4.2 Cross-Validation Analyses

Figure 2 shows the 5-fold cross-validation results of two classifiers, XGBC and DTC, on the Android malware dataset. Each part of the bar indicates the classification score achieved in one fold (K1–K5) of the cross-validation. The XGBC model had the fold scores of 0.881, 0.889, 0.901, 0.907, and 0.910, which led to an average performance of 0.8976 and a standard deviation of 0.0115. On the other hand, the DTC model had the scores of 0.873, 0.885, 0.891, 0.896, and 0.900, which gave the mean of 0.8890 and the standard

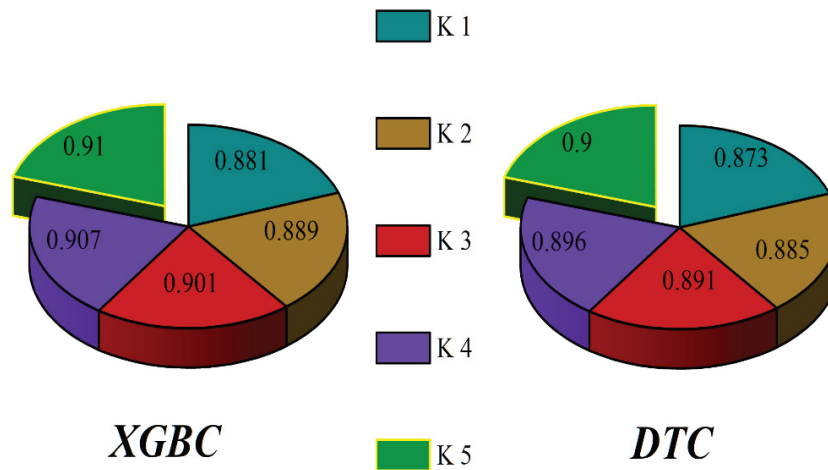


Figure 2 Pie chart illustrating the distribution of data across the K-folds used in cross-validation.

deviation of 0.0102. The plot clearly shows that XGBC outperforms DTC in each fold, which is a sign of higher model stability and better generalization ability. Even though the variance for XGBC is higher, this is a typical behavior of boosting-based frameworks due to adaptive iterative learning. Both models demonstrate stable results with low variance, confirming reliable performance across folds. However, the consistently higher values for XGBC emphasize its enhanced discriminative power in identifying malicious traffic patterns, validating its suitability for real-world Android malware detection.

5 Hypertuning Process

Table 1 delineates the hyperparameters utilized in the construction and optimization of both the baseline and optimizer-enhanced configurations of the models created for Android malware classification. The XGBC was initialized with 100 estimators and a random state of 25. Two variations enhanced by optimization techniques, XGTO and XGSD, were later created, utilizing the TOA and SDOA, respectively. XGTO was equipped with 450 estimators and a random state of 40, whereas XGSD used 390 estimators and a random state of 34. In the DTC, the baseline random state was established at 20, but the optimizer-enhanced variants, DTTO and DTSD, utilized random states of 38 and 30, respectively. As decision trees do not depend on ensemble estimators, optimization mostly affects variance reduction and structural learning. These configurations together illustrate that metaheuristic approaches may significantly improve model initialization and structural learning. By refining estimator depth and random states in XGBC, together with structural parameters in DTC, these methods enhance feature partitioning, bolster classification performance, and augment model stability without introducing additional complexity.

Table 1 Configured hyperparameters and their influence on model performance

Models	Hyperparameters	
	<i>N</i> Estimators	Random State
XGBC	100	25
XGTO	450	40
XGSD	390	34
DTC	–	20
DTTO	–	38
DTSD	–	30

A direct comparison between the baseline and optimizer-enhanced models demonstrates the effectiveness of the applied metaheuristic strategies. For XGBC, the integration of TOA improved the overall accuracy from 0.9100 to 0.9800, representing a performance increase of approximately 7.69%. Similarly, MCC improved from 0.8800 to 0.9733, indicating stronger agreement between predicted and actual malware classes. The SDOA-enhanced XGSD model also improved performance, increasing overall accuracy to 0.9600 and MCC to 0.9467. Comparable improvements were observed for the decision-tree-based models, where DTTO increased overall accuracy from 0.9000 to 0.9700, while DTSD improved it to 0.9500. These improvements demonstrate that optimizer-assisted learning enables better exploration of the hyperparameter search space and more effective adaptation to complex Android network traffic patterns. TOA particularly exhibited superior convergence behavior due to its adaptive exploitation-exploration balancing mechanism, allowing the classifier to identify more discriminative decision boundaries. In comparison, SDOA also improved learning performance but converged more gradually because of its population-diversity preservation behavior. Overall, the comparative analysis confirms that both optimization methods substantially strengthen malware classification capability compared to the original baseline classifiers.

Figure 3 illustrates the convergence process of optimizer-enhanced classifiers in terms of improving the detection of Android malware. From the

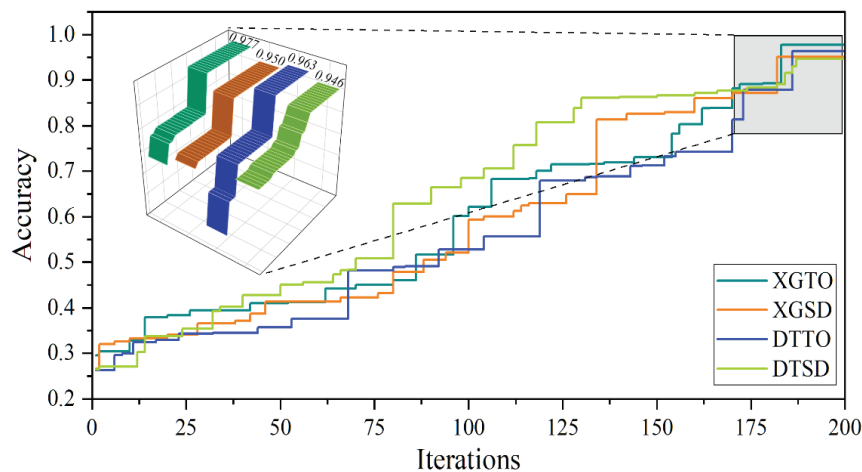


Figure 3 Vertical plot of the model's convergence behavior, showing gradual error reduction and leveling off over successive iterations.

convergence analysis, it is possible to understand how fast and uniformly a learning model converges to its best possible performance in terms of making predictions through iterations, based on how well its learning algorithm performs in terms of learning patterns in data. In fact, based on a TOA, which is utilized in improving its performance, it is clear that the best possible classification accuracy of 0.977 was attained after 200 iterations by the XGTO model. In the DTTO model, it converged at 0.963, although it was slower with a minor oscillation. Although individual decision trees are actually simpler compared to boosted ensembles, it is clear that the optimization step enhanced boundary identification, enabling the identification of anomalies present within network traffic in malware by DTTO. In conclusion, XGSD and DTSD converge at 0.950 and 0.946, respectively, at a slower rate, signifying that SDOA is inefficient compared to TOA when moving through a large feature Android network. In general, based on the trends of convergence, metaheuristic-inspired ensemble methods prove to be faster and converge at a higher performance compared to others, making them apt for use in robust and precise detection of Android malware. This clearly indicates that convergence speed is one of the most important criteria in ensuring robust learning capabilities.

6 Results

Table 2 shows the comparative performance of base and optimizer-enhanced classifiers in detecting Android malware, measured by various metrics. Among the baseline models, XGBC was able to achieve a training accuracy of 0.9081 and a test accuracy of 0.9175, thus just marginally beating DTC, whose training and test accuracy were 0.8750 and 0.9090, respectively. The same patterns are evident in other metrics as well. Precision, F1-score, and the P4-metric exhibit trends consistent with those observed for accuracy, thereby reflecting the model's predictive stability. In contrast, the MCC underlines a stronger correlation of XGBC between the predicted and actual labels as compared to DTC. These findings suggest that gradient boosting ensembles are more capable of capturing intricate network traffic patterns than a single decision-tree structure in the case of Android malware.

There was a substantial performance improvement when metaheuristic optimization was used. The XGTO model, equipped with the TOA, was at its best with an overall accuracy of 0.9800, a precision of 0.9800, and an MCC of 0.9733, thus showing superior discriminative capability and enhanced generalization. A simpler model, such as the decision tree, on the other

Table 2 DTC and XGBC base models got their results through the performance evaluators

Quantifier	Model					
	XGBC			DTC		
	Train	Test	All	Train	Test	All
Accuracy	0.9081	0.9175	0.9100	0.8750	0.9090	0.9000
Precision	0.9082	0.9176	0.9100	0.8750	0.9090	0.9000
P4-metric	0.9081	0.9175	0.9100	0.8750	0.9090	0.9000
F1-score	0.9081	0.9175	0.9100	0.8750	0.9090	0.9000
MCC	0.8775	0.8900	0.8800	0.8750	0.8786	0.9000
Quantifier	XGTO			DTTO		
	Train	Test	All	Train	Test	All
	Train	Test	All	Train	Test	All
Accuracy	0.9793	0.9830	0.9800	0.9625	0.9710	0.9700
Precision	0.9793	0.9831	0.9800	0.9625	0.9710	0.9700
P4-metric	0.9793	0.9830	0.9800	0.9625	0.9710	0.9700
F1-score	0.9792	0.9830	0.9800	0.9625	0.9710	0.9700
MCC	0.9723	0.9773	0.9733	0.9625	0.9613	0.9700
Quantifier	XGSD			DTSD		
	Train	Test	All	Train	Test	All
	Train	Test	All	Train	Test	All
Accuracy	0.9595	0.9620	0.9600	0.9489	0.9545	0.9500
Precision	0.9595	0.9621	0.9600	0.9489	0.9545	0.9500
P4-metric	0.9595	0.9620	0.9600	0.9489	0.9545	0.9500
F1-score	0.9595	0.9620	0.9600	0.9489	0.9545	0.9500
MCC	0.9460	0.9493	0.9467	0.9318	0.9393	0.9333

hand, attained an overall accuracy of 0.9700 and an MCC of 0.9700, thus confirming the positive effect of optimization even on a less complex model.

Moreover, SDOA was able to enhance both XGBC (XGSD) and DTC (DTSD), which reached overall accuracies of 0.9600 and 0.9500, respectively. The consistent increments in P4-metric, F1-score, and MCC imply that optimizer-driven configurations invigorate both ensemble and tree-based models by thoroughly delving into the hyperparameter spaces to identify malware-induced traffic anomalies. Together, these findings demonstrate that XGBC is the model that provides the strongest performance for Android malware detection, particularly when combined with TOA. The integration of an optimizer persistently causes accuracy, precision, and correlation metrics to increase, which is in line with the idea that metaheuristic-guided hyperparameter tuning substantially improves model sensitivity and dependability when it comes to identifying malicious behavior in the complex network traffic datasets.

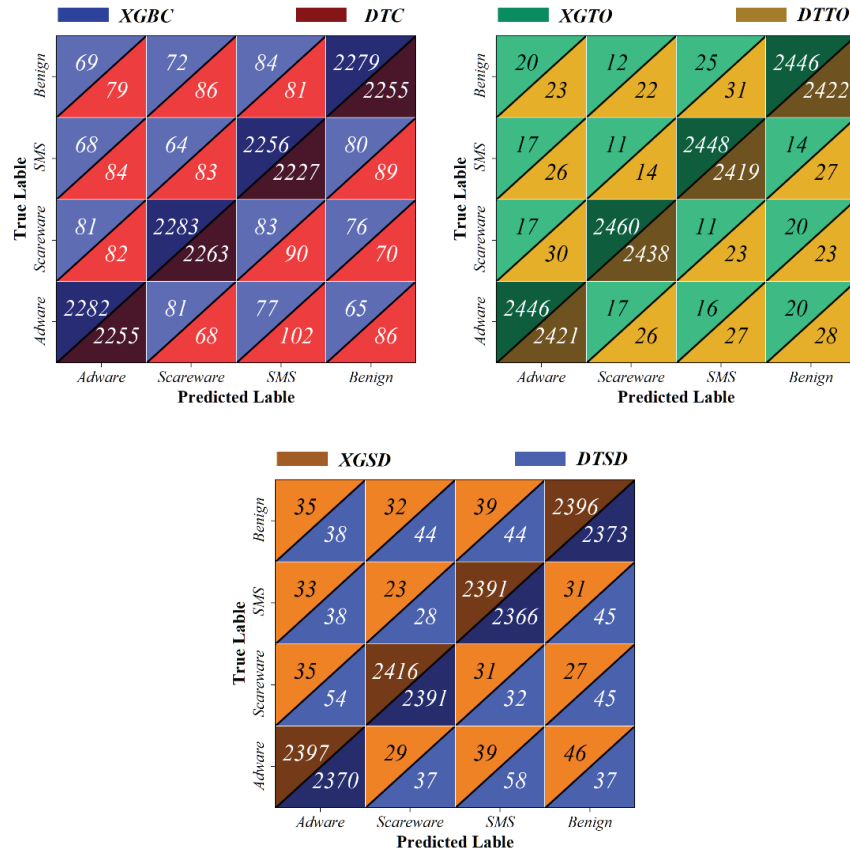


Figure 4 Confusion matrix representing the classification performance of the models under four evaluated conditions.

Figure 4 shows descriptive plots of correctly predicted instances for three classifier settings, as evaluated on the Android malware classification task for four classes of malware, namely Adware, Scareware, and SMS malware, as well as benign ones. These plots reveal details about instances classified correctly on the corresponding diagonal of each of these four confusion matrices. In the baseline settings, XGBC always made more correct classifications than DTC for all four classes, signifying its superior ability to identify patterns of features for malware classification. Likewise, for optimizer-improved approaches, XGTO made more correct classifications than DTTO for all four classes. In the case of XGSD and DTSD, XGSD showed superior classification accuracy for all four classes. Overall, XGTO

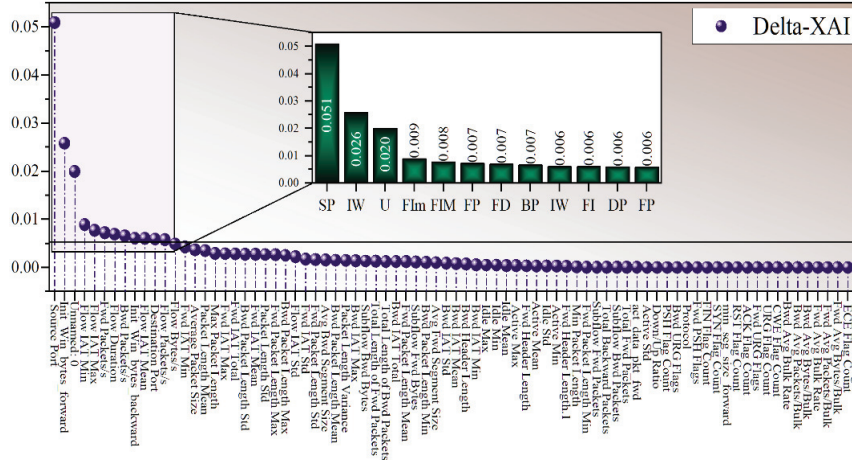


Figure 5 Symbol plot representing the Δ -XAI sensitivity of input features, highlighting their relative influence on model predictions.

has the maximum number of correct classifications across different models, thereby demonstrating its enhanced learning capability and power in handling multi-class Android malware samples. This evidence brings to the fore the effectiveness of the combination of metaheuristic optimization and gradient boosting classification techniques since it results in the best feature processing, the least number of errors, and the highest accuracy for each class. Hence, XGTO is a leading model for accurate and reliable malware classification and detection.

Figure 5 shows the feature significance distribution for the Android malware classification model, which was created with Delta-XAI, a model-agnostic explainability technique. The bar graph depicts the influence of specific factors on the machine learning decision process. It effectively shows the network traffic signals that led to the predictions. Out of the attributes, Source Port (SP) is the single most important factor, with a Delta-XAI score of 0.051, which corresponds to almost 100% relative relevance in the top 10 unit ranking. The next two most important features are IW (Initial Window Bytes Forward, 0.026) and U (Write Operation or User Interaction, 0.020), which together make up more than 80% of the total explanatory power of the top 10 features. The rest of the features -FIm, FM, FP, and FD- have very small contributions (<0.01), but more than 50 other features have very little influence (<0.006), which points to a sparse, long-tail distribution of feature significance. Such a distribution implies that the model relies heavily

on a small set of highly predictive signals, thus enabling it to achieve strong and interpretable classification results. Feature engineering efforts will yield the best results if they are concentrated on SP, IW, and U, while model simplification may be possible without compromising accuracy. Besides, these dominant features can be used as rules for a fallback system to quickly detect malware. Delta-XAI evaluation reveals a focused, interpretable, and efficient model architecture, with potential risks if key aspects are noisy or manipulated.

From a practical cybersecurity perspective, the dominant features identified through Delta-XAI can support the development of efficient and interpretable malware monitoring systems. For example, abnormal SP behavior may indicate unauthorized communication attempts or hidden command-and-control traffic commonly associated with malicious applications. Similarly, unusual IW patterns may reflect manipulated transport-layer behavior during malware communication sessions, while excessive User Interaction or Write Operations may indicate suspicious background activity, unauthorized file modification, or hidden data exfiltration processes. These highly influential features can therefore be integrated into lightweight rule-based intrusion detection systems, mobile antivirus engines, and edge-security monitoring platforms to enable rapid anomaly identification with reduced computational overhead. Furthermore, because the proposed framework identifies a compact set of highly discriminative behavioral indicators, security analysts may utilize these signals for faster forensic investigation, malware profiling, and adaptive threat-response strategies in practical Android cybersecurity environments.

7 Discussion

Android malware remains a significant threat to mobile ecosystems, as it exploits the operating system's vulnerabilities, steals sensitive data, and spreads through infected applications, ultimately compromising device security, user privacy, and the stability of the entire platform. Therefore, it is necessary to resort to sophisticated detection methods, which can be done by implementing powerful and versatile machine learning algorithms that facilitate the distinction of harmful and benign activities even in the presence of various malware families. This research used ensemble and decision tree methods, reinforced by metaheuristics, for this purpose. The proposed system is dependable because of its incorporation of rigorous preprocessing, cross-validation, and feature selection techniques, ensuring that all derivations of

insight from the model are based on helpful and non-redundant information. Delta-XAI interpretability increases the reliability and verification of decision-making of the model through its capability of indicating influential signals of malware classification, thereby facilitating repeatability of the model.

In application, these models can be applied in antivirus software, mobile security platforms, and business network monitoring systems, as they help in the quick identification of malicious traffic and risk stratification. Optimizer-enabled models have learning abilities that can adapt to evolving trends of viruses in large datasets of networks. The research value of this approach lies in combining metaheuristic optimization techniques with gradient boosting and decision tree classification methods, thereby enhancing hyperparameter tuning, model structure learning, and discriminative modeling abilities. The utilization of explainable AI methods effectively connects different aspects of cybersecurity research that were previously separated, thereby providing readable, yet accurate and reliable, predictive results. One of the disadvantages is the susceptibility to hostile feature manipulation, dependence on a small number of key features, and computational expenses related to optimization, which may limit scalability, especially when there are limited resources. Incorporating dynamic feature extraction, hybrid deep learning approaches, and multi-source datasets may, in fact, be able to lift the detection capability as well as the generalizability in future research. These feasible implications encompass the intensification of feature engineering, simplification of model architectures for easy and rapid deployments, and rule-based fallback systems employing prominent features, thus enabling not only rapid and accurate detection but also interpretability to be retained.

Although TOA and SDOA demonstrated strong optimization capability within the proposed framework, additional benchmarking against conventional hyperparameter optimization approaches such as Grid Search, Bayesian Optimization, and Optuna would provide broader comparative insight into optimization efficiency and computational trade-offs. Future studies will therefore investigate comparative optimizer benchmarking under identical Android malware classification conditions to further evaluate convergence stability, computational complexity, and predictive robustness. Although the proposed framework demonstrated strong performance on the CIC-derived Android malware dataset, additional cross-dataset validation would further strengthen the analysis of robustness and transferability. Although the present study concentrated on feature-driven ensemble and decision-tree learning approaches because of their suitability for structured

network-flow data and explainability-oriented analysis, future investigations may incorporate deep learning architectures such as CNNs, RNNs, and Transformer-based models for broader comparative benchmarking. Hybrid frameworks combining deep representation learning with explainable optimization-driven classification may further strengthen Android malware detection capability. Future research will therefore evaluate the proposed optimizer-enhanced framework using additional benchmark Android malware datasets and real-time network traffic environments to further investigate model generalization under heterogeneous malware conditions.

8 Conclusion

This paper provides a comprehensive review of Android malware classification using feature-centric machine learning techniques, combining gradient boosting and decision tree classification algorithms with metaheuristics for optimization. Feature selection analyses, involving the VIF, ensured that only orthogonal and discriminant network traffic features were considered, thereby eliminating redundancy and improving generalizability. Sensitivity analysis using Delta-XAI helped identify features of utmost importance, of which SP, IW, and U feature variables emerged as leaders. These three feature variables, combined, accounted for more than 80% of the cumulative importance of the top 10 most essential features, signifying the importance of effective feature engineering towards efficient Android malware classification. Gradient boosting classifier, XGTO, optimized through metaheuristics, significantly outperformed all other benchmarks and its metaheuristic-variant competitors in all evaluation criteria. XGTO achieved maximum accuracy, precision, and MCC and displayed outstanding discriminability capability on all four malware types, including Adware, Scareware, SMS malware, and benign applications. Optimized convergence, decreased errors, and increased model sensitivity towards complex network traffic anomalies characteristic of malware, via inclusion of iterative ensemble learning, effective feature engineering, and hyperparameter tuning through TOA, help enhance model discriminability and interpretability significantly. Indeed, this paper demonstrates that combining feature validation, model interpretability, and metaheuristics leads to extremely dependable, efficient, and practical methods for malware detection on Android applications. With reliance on SP, IW, and U as primary feature variables, XGTO emerges as the most optimal and dependable approach towards effectively addressing future mobile threats.

Acknowledgement

This work was supported by the project of Enhancing Young and Middle-Aged Teachers' Research Basis Ability in Colleges of Guangxi (No. 2025KY1159).

References

- [1] M. Chau and R. Reith, Smartphone market share, IDC Corp., USA, 2020, 444.
- [2] L. Eze, U. B. Chaudhry, and H. Jahankhani, Quantum-Enhanced Machine Learning for Cybersecurity: Evaluating Malicious URL Detection, *Electronics*, 2025, 14(9): 1827.
- [3] J. Ylitalo, The Interface Between Technology and People in Cybersecurity: Technological Solutions Supporting Humans in Organizational Protection, 2025.
- [4] H. Rathod and S. Agal, A study and overview on current trends and technology in mobile applications and its development, *International Conference on ICT for Sustainable Development*, 2023, pp. 383–395.
- [5] A. Karim, S. A. Ali Shah, R. Bin Salleh, M. Arif, R. M. Noor, and S. Shamshirband, Mobile botnet attacks-An emerging threat: Classification, review and open issues, *KSII Transactions on Internet and Information Systems*, 2015, 9(4): 1471–1492.
- [6] S. Aonzo, G. C. Georgiu, L. Verderame, and A. Merlo, Obfuscapk: An open-source black-box obfuscation tool for Android apps, *SoftwareX*, 2020, 11: 100403.
- [7] D. J. Edwards, Malware Defenses, in *Critical Security Controls for Effective Cyber Defense: A Comprehensive Guide to CIS 18 Controls*, Springer, 2024.
- [8] M. Suleman, T. R. Soomro, T. M. Ghazal, and M. Alshurideh, Combating against potentially harmful mobile apps, *International Conference on Artificial Intelligence and Computer Vision*, 2021, 1377: 154–173.
- [9] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-Rimy, T. A. E. Eisa, and A. A. H. Elnour, Malware detection issues, challenges, and future directions: A survey, *Applied Sciences*, 2022, 12(17): 8482.
- [10] H. A. Abdulbaqi, A. M. Ghandour, and T. A. Jawad, Develop secure software specifications for Android app concealing the information and safeguarding data, *Iraqi Journal of Computer Science and Mathematics*, 2024, 5(4): 243–257.

- [11] A. El Hami, *Methods and Applications of Artificial Intelligence: Dynamic Response, Learning, Random Forest, Linear Regression, Interoperability, Additive Manufacturing and Mechatronics*, John Wiley & Sons, 2025, 2: 1–225.
- [12] U. Baldangombo, Z. Kherlenchimeg, and U. Naidansuren, Android malware detection using machine learning, *Journal of Institute of Mathematics and Digital Technology*, 2024, 6(1): 130–145.
- [13] A. Anand, J. P. Singh, and V. Dhoundiyal, Android malware detection using HexCode features, 2024, pp. 1–26.
- [14] S. Y. Yerima, S. Sezer, and I. Muttik, Android malware detection using parallel machine learning classifiers, *International Conference on Next Generation Mobile Apps, Services and Technologies*, 2014, pp. 37–42.
- [15] R. M. Hamad et al., *COVID-19 Pandemic and its Impact on Sustainable Development Goals: An Observation of South Asian Perspective*, 2020.
- [16] S. Kotha, S. Hariharasitaraman, D. Saravanan, and S. Ahmed, Android malware detection using deep learning, *International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC)*, 2025, pp. 1–4.
- [17] V. Bansal, N. Baliyan, and M. Ghosh, Dynamic Android malware detection using Light Gradient Boosting Machine, *International Conference on Artificial Intelligence and Speech Technology (AIST)*, 2022, pp. 1–6.
- [18] H. H. R. Manzil and S. M. Naik, Detection approaches for Android malware: Taxonomy and review analysis, *Expert Systems with Applications*, 2024, 238: 122255.
- [19] H. A. Al-Kaaf, *Integrated Information Gain with Extra Tree Algorithm for Feature Permission Analysis in Android Malware Classification*, Universiti Teknologi Malaysia, 2022.
- [20] M. P. Singh and H. K. Khan, Malware detection in Android applications using machine learning, *International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS)*, 2023, pp. 105–110.
- [21] X. Y. Liew, N. Hameed, and J. Clos, An investigation of XGBoost-based algorithm for breast cancer classification, *Machine Learning with Applications*, 2021, 6: 100154.
- [22] T. Chen and C. Guestrin, XGBoost: A scalable tree boosting system, *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

- [23] A. Beucher, A. B. MÃ¸ller, and M. H. Greve, Artificial neural networks and decision tree classification for predicting soil drainage classes in Denmark, *Geoderma*, 2019, 352: 351–359.
- [24] T. Hamadneh et al., On the application of Tailor Optimization Algorithm for solving real-world optimization application, *International Journal of Intelligent Engineering Systems*, 2025, 18(1): 1–12.
- [25] J. Zhang, Spotted Deer Optimization Algorithm, *HAL*, 2026, pp. 1–13.
- [26] R. M. O’Brien, A caution regarding rules of thumb for variance inflation factors, *Quality & Quantity*, 2007, 41(5): 673–690.
- [27] N. Kapure, H. Joshi, P. Kumari, R. Mistri, and M. Mali, FRAME: Forward Recursive Adaptive Model Extraction-A technique for advance feature selection, *arXiv preprint arXiv:2501.11972*, 2025. <https://doi.org/10.48550/arXiv.2501.11972>.
- [28] S. Bates, T. Hastie, and R. Tibshirani, Cross-validation: What does it estimate and how well does it do it? *Journal of the American Statistical Association*, 2024, 119(546): 1434–1445.
- [29] A. De Carlo, E. Parimbelli, N. Melillo, and G. Nicora, Introducing δ -XAI: A novel sensitivity-based method for local AI explanations, *arXiv preprint arXiv:2407.18343*, 2024. <https://doi.org/10.48550/arXiv.2407.18343>.
- [30] M. S. Islam, I. Hussain, M. M. Rahman, S. J. Park, and M. A. Hossain, Explainable artificial intelligence model for stroke prediction using EEG signal, *Sensors*, 2022, 22(24): 9859.
- [31] S. Chakraborty, Android Malware Detection, *Kaggle*, 2023.

Biographies



Chao Tan graduated from Wuhan University of Technology, China, in 2015 with a Master’s Degree in Software Engineering. He currently serves as an

associate professor at the School of Information Science and Engineering, Liuzhou Institute of Technology. His primary research areas include Computer Vision and Machine Learning.



Xinlu Li obtained her Master's Degree in Software Engineering from Wuhan University, China, in 2016. She currently serves as a lecturer at the School of Civil Engineering and Architecture, Liuzhou Institute of Technology, specializing in artificial intelligence research.