

Android Malware Detection with Hybrid Feature Selection and Bayesian Optimization

Muhammad Alif Fadhillah¹, Setyo Wahyu Saputro², Muliadi³, Mohammad Reza Faisal⁴,
Radityo Adi Nugroho⁵

^{1,2,3,4,5}Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia

Article Info

Article history:

Received Jan 30, 2026

Revised Feb 23, 2026

Accepted Apr 08, 2026

Keywords:

Android malware detection

Hybrid feature selection

Bayesian optimization

Machine learning

DREBIN-215

ABSTRACT

The increasing dimensionality of Android application features poses significant challenges for accurate and efficient malware detection. This study proposes a hybrid feature selection framework that combines Minimum Redundancy Maximum Relevance (mRMR) and correlation filtering to optimize classification performance on the Drebin-215 dataset. A selected configuration of 175 features with a correlation threshold of 0.7 was evaluated using five classifiers: LSTM, Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN), and XGBoost. The experimental results show that dimensionality reduction improves classification stability and overall predictive performance. SVM exhibits the most notable improvement, with accuracy increasing from 63.05% without feature selection to 98.57% after applying the proposed framework. LSTM achieves 98.57% accuracy with an AUC of 99.86%, while Random Forest, KNN, and XGBoost consistently achieve accuracy above 97%. In addition to performance enhancement, the hybrid feature selection approach substantially improves computational efficiency. SVM training time decreases from 770.75 seconds to 155.88 seconds, and testing time is reduced from 15.581 seconds to 0.3824 seconds. KNN testing time also decreases from 1.623 seconds to 0.4595 seconds..

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Setyo Wahyu Saputro,

Department of Computer Science,

Lambung Mangkurat University,

Jalan Jenderal Ahmad Yani KM 36, Banjarbaru, Kalimantan Selatan 70714.

Email: setyo.saputro@ulm.ac.id

1. INTRODUCTION

Smartphones have become an essential component of modern life, supporting communication, entertainment, financial transactions, and healthcare services [1]. However, their widespread adoption has significantly increased security risks, as mobile devices store highly sensitive personal information that is vulnerable to malware attacks. With more than 2.6 million applications available, the Android ecosystem has become a primary target for malicious developers [2]. Several studies have highlighted that low user awareness of cybersecurity significantly exacerbates Android malware threats [3]–[5]. Android devices commonly store sensitive data such as messages, images, banking credentials, and health records [6], motivating extensive research into machine learning (ML) and deep learning (DL) based approaches to enhance detection accuracy and response to cyber threats.

Despite these advances, Android malware detection remains a challenging task due to the dynamic execution environment and increasingly sophisticated obfuscation techniques. Widely used benchmark datasets, such as DREBIN-215, exhibit several intrinsic limitations, including class imbalance (with an approximate ratio of 1:1.7 between malware and benign samples) and

high-dimensional feature representations comprising 215 attributes. Such characteristics often lead to biased classifiers, increased computational complexity, and degraded generalization performance. Moreover, deep learning models such as Long Short-Term Memory (LSTM) networks, while powerful, impose substantial computational costs, particularly when combined with exhaustive hyperparameter tuning methods such as GridSearchCV, which significantly increases training overhead [7], limiting their practicality for real-time or resource-constrained environments.

To mitigate class imbalance, data-level strategies such as oversampling, undersampling, and hybrid sampling have been extensively explored. Oversampling techniques increase the number of minority-class samples [8], whereas undersampling reduces the dominance of majority-class instances [9]. Hybrid approaches combine both strategies to improve dataset quality [10]. Among these, SMOTE–Tomek Link has been shown to be effective by synthesizing minority samples while simultaneously removing ambiguous majority instances, thereby reducing redundancy and improving classification robustness [11]. Although sampling techniques can alleviate imbalance, they do not directly address the issue of redundant or irrelevant features in high-dimensional malware datasets.

Feature redundancy and high dimensionality pose additional challenges for ML-based malware detection. Models such as Support Vector Machines (SVM), Random Forests (RF), Gradient Boosting, XGBoost, and LSTM are known to be sensitive to noisy or correlated input features, which may inflate training costs and degrade predictive stability [12]–[16]. Consequently, effective feature selection becomes a critical step to reduce dimensionality, suppress redundancy, and improve both classification accuracy and computational efficiency. Nevertheless, many existing studies rely on a single feature selection technique, which may be insufficient to capture complementary relevance and redundancy characteristics within complex Android malware features.

In addition to feature engineering, model performance is strongly influenced by hyperparameter configurations. Hyperparameter optimization plays a crucial role in adapting learning algorithms to specific data characteristics, directly affecting accuracy, stability, and convergence behavior. Bayesian optimization has emerged as a particularly effective approach for hyperparameter tuning, as it constructs a probabilistic surrogate model based on previous evaluations to efficiently search for optimal configurations that minimize an objective function [17]. Its ability to handle expensive, non-differentiable, and complex optimization landscapes makes Bayesian optimization well suited for malware detection pipelines involving multiple models and large feature spaces [18].

Previous studies have demonstrated the potential of combining sampling, feature selection, and optimization strategies for Android malware detection. For example, an LSTM model integrated with SMOTE and GridSearchCV achieved an accuracy of 99.24% [7]. Other works have reported strong performance using LSTM-based architectures compared with ANN, CNN, and GRU models [19], as well as hybrid sampling and feature selection approaches optimized with Bayesian or evolutionary techniques [1], [20], [21]. However, these studies often treat feature selection, sampling, and hyperparameter optimization as isolated components, or evaluate them on a limited set of models, leaving the combined effect of hybrid feature selection and Bayesian optimization across diverse classifiers insufficiently explored.

Motivated by these gaps, this study proposes an integrated Android malware detection framework that combines hybrid feature selection with Bayesian hyperparameter optimization. By leveraging complementary feature relevance and redundancy analysis alongside probabilistic optimization, the proposed approach aims to improve classification accuracy, stability, and computational efficiency across multiple machine learning and deep learning models. This integration is expected to provide a more robust and scalable solution for Android malware detection in high-dimensional and imbalanced data settings.

2. RESEARCH METHOD

This study develops and evaluates an Android malware detection model using the DREBIN-215 dataset. Class imbalance is first addressed using SMOTE-Tomek, followed by a hybrid feature-selection strategy combining Minimum Redundancy Maximum Relevance and correlation-based filtering. The refined dataset is split into training and testing sets, after which SVM, Random Forest, XGBoost, KNN, and LSTM models are trained and optimized via Bayesian hyperparameter tuning. Model performance is then assessed on the testing subset, and the overall workflow is illustrated in Figure 1.

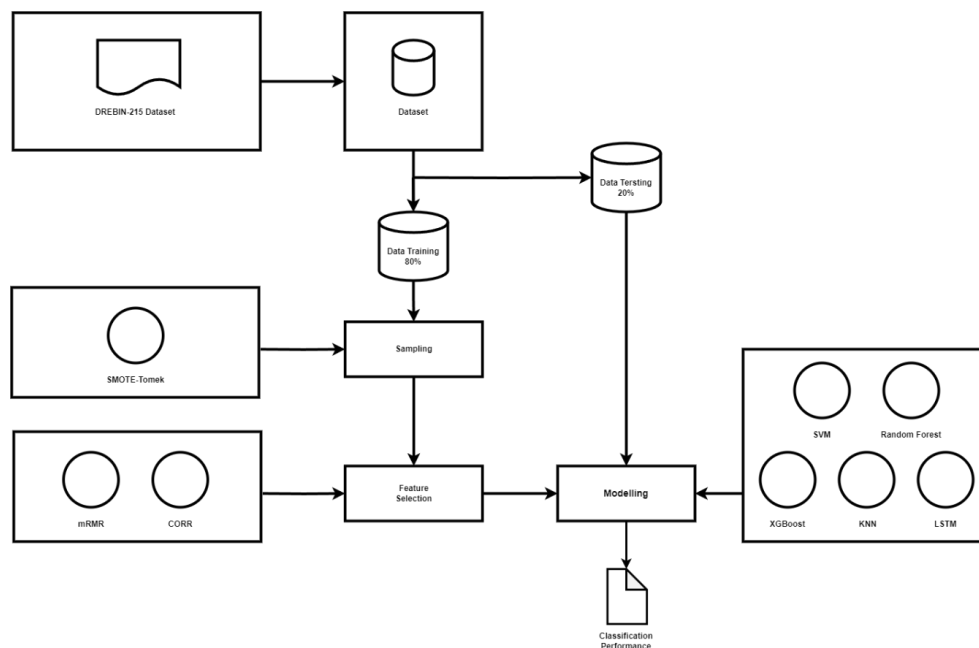


Figure 1. Research Flowchart

2.1. Dataset

The dataset used in this study is the DREBIN-215 Android malware dataset, derived from the static analysis-based Drebin project and widely adopted in Android malware research, including multilevel classifier fusion frameworks [22]. It comprises 15,036 Android applications, with 5,560 malware samples and 9,476 benign samples, each represented by a 215-dimensional feature vector extracted from manifest files and disassembled code. Features include permissions, API calls, intent filters, hardware components, and other static behavioral attributes that characterize application behavior. Due to its comprehensive coverage and balanced representation of benign and malicious applications, DREBIN-215 serves as a standard benchmark for evaluating machine learning approaches in Android malware detection.

2.2. SMOTE-Tomek

The issue of class imbalance in the original DREBIN-215 dataset, where benign samples significantly outnumbered malware instances, presents a challenge that can bias the learning process and reduce a model's ability to correctly identify minority-class patterns. This problem is commonly addressed using sampling strategies, and among them SMOTE is widely adopted because it generates synthetic minority samples by interpolating between nearest neighbors without altering the majority class, although it may occasionally introduce noise in regions where classes overlap [23]. To refine this process, the SMOTE-Tomek method combines SMOTE with Tomek Links, an undersampling technique that removes ambiguous majority-class samples forming overlapping pairs with minority samples, thereby improving class separability and often yielding cleaner decision boundaries [24], [25]. In this research, SMOTE-Tomek is applied after initial preprocessing to correct the imbalance observed in DREBIN-215, as illustrated in Figure 2. SMOTE first synthesizes new malware samples based on their intrinsic neighborhood structure,

and the Tomek component subsequently eliminates overlapping benign–malware pairs. Through this combined operation, the dataset transitions from a heavily skewed distribution to an evenly balanced set of 9,473 samples in each class, providing a more stable foundation for downstream feature selection and model training.

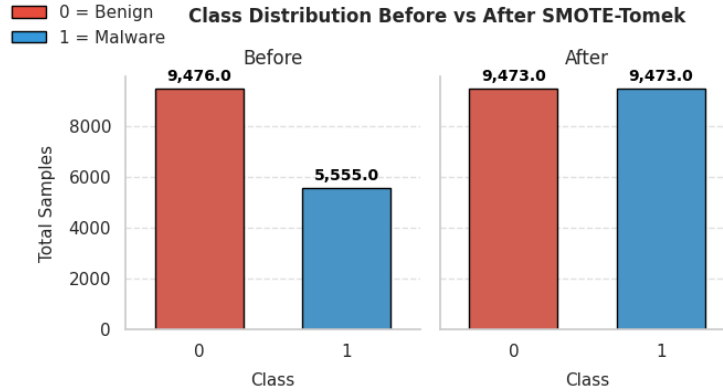


Figure 2. Class distribution before/after SMOTE–Tomek

2.3. Minimum Redundancy Maximum Relevance (mRMR)

Feature selection is performed to enhance classification performance by identifying the most informative features from a larger set. The primary objective is to remove irrelevant or redundant features, forming a more efficient subset that improves model accuracy [26], [27]. The minimum redundancy maximum relevance (mRMR) method is chosen due to its proven effectiveness in handling binary data and its widespread use in prior studies for optimal feature selection [28]–[30]. Analysis is conducted using the built-in feature selection interface in MATLAB.

The mRMR algorithm ranks features based on both relevance and uniqueness, evaluating relationships among features through an internal scoring mechanism. Features with the highest scores are considered most relevant, while those with lower influence are deemed redundant [31], [32]. By maximizing relevance and minimizing redundancy, mRMR generates a ranked feature list, effectively reducing irrelevant features while maintaining high classification accuracy. This subset is subsequently combined with correlation-based filtering in the hybrid feature selection framework used in this study.

2.4. Correlation-Based Feature Selection

Correlation-based feature selection (CFS) is applied to complement the mRMR method by reducing feature redundancy and addressing high dimensionality, thereby improving computational efficiency and classification performance [33]. Unlike conventional methods that evaluate features individually, CFS considers both the correlation of each feature with the target variable and the inter-feature correlations within a subset.

The correlation formula in CFS is expressed as follows:

$$C_{zc} = \frac{nc_{za}}{\sqrt{n + n(n-1)c_{ii}}} \#(1)$$

where C_{zc} represents the correlation between the class variable and the feature subset, n is the number of features in the subset, C_{zi} is the average correlation between features and the class variable, and C_{ii} is the average inter-feature correlation [25].

By selecting features with strong relevance to the target while minimizing redundancy, CFS produces a refined subset that enhances both the efficiency and accuracy of downstream machine learning models. This subset is then combined with the mRMR-selected features to form the hybrid feature set used in this study.

2.5. Random Forest

Random Forest is an ensemble learning algorithm that builds multiple decision trees, each producing an independent prediction. The final class label is determined by majority voting across all trees, which helps reduce overfitting and improves overall predictive accuracy [34].

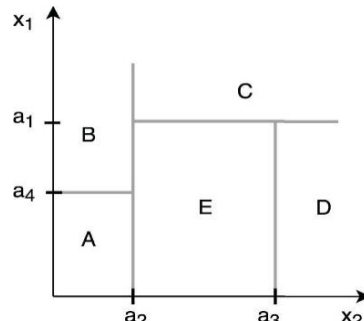


Figure 3. Recursive binary partitioning of a two-dimensional subspace

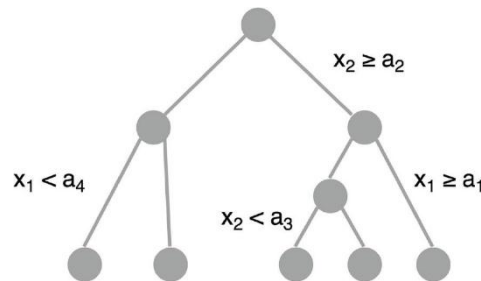


Figure 4. Graphical representation of the decision tree corresponding to Figure 3

Each decision tree recursively partitions the dataset based on feature values until a stopping criterion is reached, with terminal nodes known as leaves. Figure 3 illustrates recursive binary partitioning in a two-dimensional input space using axis-aligned splits. The first split occurs at $x_2 \geq a_2$, followed by further divisions along other axes, as shown in the figure [35]. Figure 4 provides a graphical representation of the corresponding decision tree structure derived from this partitioning process, illustrating how the hierarchical splits are translated into a tree-based model.

This tree-based ensemble approach is particularly effective in handling high-dimensional datasets such as DREBIN-215, as it can capture complex interactions among features and improve classification performance in Android malware detection.

2.6. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm used for both classification and regression tasks. Its primary objective is to identify the hyperplane that maximizes the margin between data classes, thereby enabling linear separation. SVM has proven to be highly effective for datasets with limited training samples, where traditional statistical methods may fail to guarantee an optimal solution [36]. The concept of margin maximization and the role of support vectors in defining the optimal hyperplane are illustrated in Figure 5.

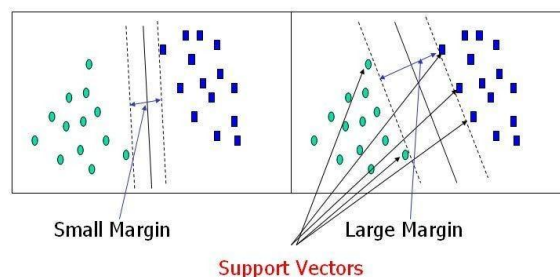


Figure 5. Margin and Support Vector

When linear separation is difficult to achieve, SVM employs kernel functions to project the training data into a higher-dimensional feature space, making separation more feasible. Kernel functions such as the Radial Basis Function (RBF) and polynomial kernels enhance classification accuracy, particularly when dealing with small training sets. Selecting an appropriate kernel function is crucial for accurately determining the optimal hyperplane and minimizing classification errors. The effectiveness of SVM largely depends on the type and scale of the kernel, as well as the smoothness of the kernel surface, which influences the clarity and precision of the separation [37].

2.7. XGBoost

Extreme Gradient Boosting (XGBoost) is a scalable tree-boosting system introduced by Chen et al. [38], recognized for its high efficiency and predictive accuracy. XGBoost improves upon the traditional Gradient Boosting Decision Tree (GBDT) algorithm by building an ensemble of decision trees to address both classification and regression tasks effectively.

In the gradient boosting process, each iteration focuses on correcting the residuals of the previous model by minimizing a defined objective function. Let $f_k(x_i)$ denote the prediction for the i -th sample produced by the k -th tree. The series of functions f_k is learned by minimizing the following objective function:

$$Obj = \sum_{i=1}^n L(\hat{y}_i, y_i) + \sum_{k=1}^K \Omega(f_k) \quad (2)$$

where L represents the loss function measuring the difference between the predicted value $\hat{y}_i$ and the actual target y_i , and $\Omega(f_k)$ is the regularization term controlling the complexity of tree f_k . Specifically,

$$\Omega(f_k) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 \quad (3)$$

Here, T is the number of leaves in the tree, w_j denotes the weight of leaf j , and γ and λ are regularization parameters [39].

By combining gradient boosting with regularization, XGBoost achieves both high predictive performance and improved generalization, making it suitable for high-dimensional and imbalanced datasets such as DREBIN-215 in Android malware detection.

2.8. K-Nearest Neighbor (KNN)

K-Nearest Neighbor (KNN), originally introduced by Cover and Hart, is a non-parametric supervised learning algorithm widely used for classification tasks [40]. In Android malware detection, KNN assigns a class label to an application based on the majority class among its k nearest neighbors in the feature space. Unlike parametric models, KNN does not require an explicit training phase and instead relies on distance-based similarity between feature vectors.

Given a test sample represented as an n -dimensional feature vector, the KNN algorithm computes the distance between this sample and all instances in the training set. The k closest samples are then selected, and the final class label is determined through majority voting. Since datasets such as DREBIN-215 consist of high-dimensional static features, the choice of distance metric significantly affects classification performance.

In this study, four commonly used distance metrics are considered: Euclidean, Manhattan, Minkowski, and Chebyshev distances [41]. The Euclidean distance measures the straight-line distance between two points in an n -dimensional space [42], is defined as:

$$d(x, y) = \sqrt{\sum_{i=1}^n |x_i - y_i|^2} \quad (4)$$

The Manhattan distance, also known as the city-block distance [43], computes distance as the sum of absolute differences across dimensions:

$$d(x, y) = \sum_{i=1}^n |x_i - y_i| \quad \#(5)$$

The Minkowski distance generalizes both Euclidean and Manhattan distances [44] and expressed as:

$$d(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^c \right)^{1/c} \quad \#(6)$$

where p is a positive real parameter that controls the metric behavior. The Chebyshev distance, also referred to as the chessboard distance [45], measures the maximum absolute difference across all feature dimensions:

$$d(x, y) = \max_i |x_i - y_i| \quad \#(7)$$

Each distance metric exhibits different sensitivity to feature scaling and dimensionality. Therefore, in this research, both the number of neighbors (k) and the distance metric are treated as hyperparameters and optimized using Bayesian optimization to improve classification robustness and accuracy in distinguishing benign and malicious Android applications.

2.9. Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNNs) are deep learning models designed to capture sequential dependencies through internal memory mechanisms, distinguishing them from traditional feedforward neural networks that treat inputs as independent observations. By incorporating temporal context, RNNs are particularly suitable for sequential and time-dependent data [46]. However, conventional RNNs often suffer from the vanishing gradient problem, which limits their ability to learn long-term dependencies effectively.

To overcome this limitation, the Long Short-Term Memory (LSTM) network was introduced by Hochreiter and Schmidhuber [47] and later refined by Graves [48]. LSTM architectures are explicitly designed to preserve relevant information over long sequences by regulating information flow through gated memory units, making them highly effective for sequence modeling and classification tasks.

In an LSTM network, the input vector at time step t , denoted as $x^{(t)}$, together with the hidden state from the previous time step $h^{(t-1)}$, is processed by an LSTM cell to generate a new cell state s_t and a hidden state $h^{(t)}$. For classification tasks, the final hidden state or an aggregation of hidden states is typically used as the model output [49]. Figure 6 illustrates the high-level architecture of the LSTM network, while the internal structure of an LSTM cell is depicted in Figure 7.

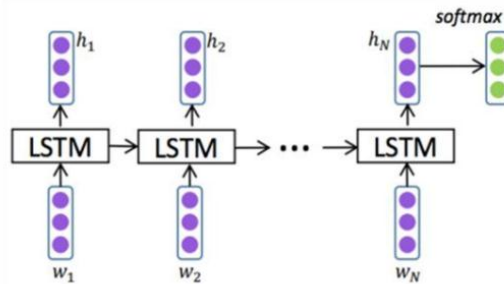


Figure 6. High-level architecture of the LSTM network

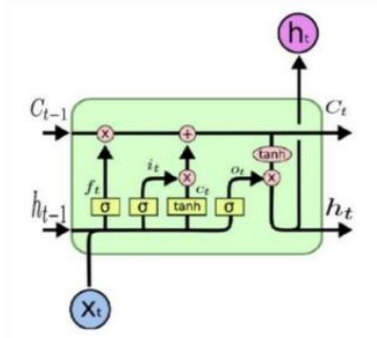


Figure 7. Internal structure of an LSTM cell

An LSTM cell consists of three primary gates: the input gate, forget gate, and output gate, which collectively regulate the flow of information within the cell [50], [51]. The mathematical formulation of the LSTM unit is defined as follows:

$$\begin{aligned}
 g_i^{(t)} &= \sigma(W_i h^{(t-1)} + u_i X^{(t)} + b_i) \\
 g_f^{(t)} &= \sigma(W_f h^{(t-1)} + u_f X^{(t)} + b_f) \\
 g_o^{(t)} &= \sigma(W_o h^{(t-1)} + u_o X^{(t)} + b_o) \\
 \hat{s}^{(t)} &= f_s(W_s h^{(t-1)} + U_s X^{(t)} + b_s) \\
 s^{(t)} &= g_f^{(t)} \odot s^{(t-1)} + g_i^{(t)} \odot \hat{s}^{(t)} \\
 h^{(t)} &= g_o^{(t)} \odot f_h(s^{(t)}) \#(8)
 \end{aligned}$$

where $\sigma(\cdot)$ denotes the sigmoid activation function, $f_s(\cdot)$ and $f_h(\cdot)$ typically represent the hyperbolic tangent (tanh) function, and $\odot$ indicates element-wise multiplication. The matrices W and U denote weight parameters associated with the previous hidden state and current input, respectively, while b represents bias vectors. The subscripts i , f , and o correspond to the input, forget, and output gates.

This formulation follows the standard LSTM equations and is mathematically equivalent to the compact representations commonly adopted in RNN literature, including the formulation summarized in Fang et al. [50]. Through this gated mechanism, LSTM networks effectively mitigate gradient degradation and retain salient information over long input sequences. Consequently, LSTM is well suited for modeling high-dimensional representations of Android malware features, where capturing complex feature interactions can contribute to improved detection performance.

3. RESULTS AND DISCUSSION

This section presents the experimental results and discussion of the proposed Android malware detection framework. The evaluation is conducted on the DREBIN-215 dataset to assess the effectiveness of the hybrid feature selection approach combined with SMOTE-Tomek sampling and Bayesian hyperparameter optimization. Several machine learning and deep learning classifiers, including Random Forest, SVM, XGBoost, KNN, and LSTM, are compared using standard performance metrics such as accuracy, precision, recall, F1-score, and AUC. The results are analyzed to highlight the impact of feature selection, class balancing, and optimization strategies on classification performance and computational efficiency.

3.1. Feature Selection

This research adopts a hybrid feature selection strategy that integrates Minimum Redundancy Maximum Relevance (mRMR) with correlation-based filtering to ensure that the resulting feature subset is both highly informative and minimally redundant. The rationale for using

a two-stage approach arises from the high dimensionality and heterogeneous nature of the DREBIN-215 dataset, where many features capture overlapping behavioral signals of malicious applications. While single-method feature selection can remove irrelevant attributes, it often fails to fully address multicollinearity and subtle inter-feature dependencies. Therefore, combining mRMR’s relevance-driven ranking with a correlation-based redundancy check allows the selection process to first identify statistically meaningful features and then refine them into a more compact, non-redundant, and stable set. This hybrid strategy not only enhances model generalization but also reduces computational overhead during training, making it a suitable approach for large-scale Android malware detection.

3.1.1. Hyperparameter Optimization for Hybrid Feature Selection

To determine the optimal configuration for the hybrid feature selection pipeline, a systematic grid search experiment was conducted to evaluate the combined effect of mRMR ranking depth (Top-K) and correlation filtering threshold on model performance. The experiment explored three levels of Top-K values (150, 175, and 200) and three correlation thresholds (0.5, 0.6, and 0.7), resulting in nine distinct feature subsets. Each configuration was evaluated using the SVM classifier with Bayesian-optimized hyperparameters, and performance was assessed based on accuracy, AUC, precision, recall, F1-score, training time, and testing time.

The experimental results, summarized in Table 1, reveal several important patterns. First, increasing the correlation threshold consistently reduces the final feature count, as more highly correlated features are removed. Second, configurations with higher Top-K values and stricter thresholds tend to produce stronger discriminative performance, with Top-K: 200 and threshold = 0.7 achieving the highest accuracy (98.60%) and AUC (99.78%). However, this configuration also results in the largest feature subset (161 features) and imposes substantially higher computational costs during testing (0.9775 seconds), nearly triple the testing time of more compact configurations.

Table 1. Grid Search Results for Hybrid Feature Selection Hyperparameters

Top-K	Threshold	Final Feature	Accuracy	AUC	Train Time (s)	Test Time (s)
150	0.5	90	0.9717	0.9946	198.71	0.3560
150	0.6	105	0.9830	0.9975	337.47	0.4329
150	0.7	119	0.9837	0.9971	288.23	0.3921
175	0.5	110	0.9771	0.9955	348.92	0.8210
175	0.6	125	0.9824	0.9972	267.89	0.4190
175	0.7	140	0.9857	0.9978	155.88	0.3824
200	0.5	126	0.9790	0.9958	320.06	0.5366
200	0.6	144	0.9827	0.9976	253.62	0.6149
200	0.7	161	0.9860	0.9978	225.12	0.9775

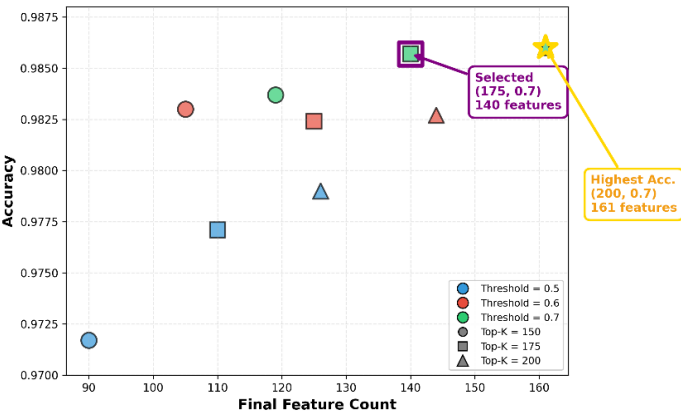


Figure 8. Performance Vs. Feature Dimensionality Trade-off

Figure 8 showing the relationship between final feature count and accuracy across nine hyperparameter configurations. The selected configuration (purple square) achieves the optimal balance between performance and efficiency, while the highest accuracy configuration (gold star)

is also shown for comparison. Different colors represent correlation thresholds (blue: 0.5, red: 0.6, green: 0.7), while marker shapes indicate Top-K values (circle: 150, square: 175, triangle: 200).

The analysis reveals a critical trade-off between predictive performance and computational efficiency. While the Top-K: 200, Threshold: 0.7 configuration achieves marginally higher accuracy (98.60% vs. 98.57%), both configurations share the identical AUC of 99.78%, indicating equivalent discriminative power in distinguishing malware from benign applications. However, the Top-K: 175, Threshold: 0.7 configuration demonstrates substantially superior computational efficiency, achieving 1.44× faster training (155.88s vs. 225.12s) and 2.56× faster inference (0.3824s vs. 0.9775s) while maintaining 15% fewer features (140 vs. 161). Conversely, overly aggressive filtering (Top-K: 150, Threshold: 0.5) produces the most compact subset (90 features) but sacrifices substantial predictive power, with accuracy dropping to 97.17% and AUC to 99.46%.

Based on these findings, the configuration of Top-K: 175 and Threshold: 0.7 was selected for the main experiments reported in this study. This configuration achieves an optimal balance, delivering state-of-the-art discriminative performance (AUC: 99.78%, tied for highest among all configurations) alongside strong accuracy (98.57%) and exceptional computational efficiency. The resulting 140-feature subset represents a 34.9% reduction from the original 215 dimensions, demonstrating that the hybrid mRMR–correlation approach successfully identifies a compact yet highly informative feature representation.

3.2. Results and Performance Evaluation

This section presents a comprehensive evaluation of the proposed Android malware detection framework, focusing on the impact of hybrid feature selection combined with SMOTE-Tomek sampling and Bayesian hyperparameter optimization. Five machine learning and deep learning classifiers Long Short-Term Memory (LSTM), Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbors (KNN), and Extreme Gradient Boosting (XGBoost) are evaluated on the DREBIN-215 dataset using standard performance metrics including accuracy, precision, recall, F1-score, and area under the ROC curve (AUC). Additionally, computational efficiency is measured through training time and testing time to assess the framework's practicality for real-world deployment. The evaluation compares two experimental configurations: models trained on the full 215-dimensional feature set and models trained on the hybrid-selected 140-feature subset (Top-K = 175, Threshold = 0.7), enabling direct quantification of feature selection benefits.

3.2.1. Model Performance Comparison

Table 2 summarizes the classification performance of all evaluated models under both experimental configurations. The results demonstrate that the hybrid mRMR correlation feature selection approach consistently enhances model performance across nearly all metrics, with particularly substantial improvements observed for margin-based and distance-based learners.

Table 2. Performance Comparison: With vs. Without Hybrid Feature Selection

Model	Config.	Acc	Precision	Recall	F1	AUC	Train (s)	Test (s)
SVM	No FS	0.6305	0.0000	0.0000	0.0000	0.9697	770.75	15.581
	175/0.7	0.9857	0.9794	0.9820	0.9807	0.9978	155.88	0.3824
LSTM	No FS	0.9677	0.9747	0.9370	0.9555	0.9895	26.66	0.513
	175/0.7	0.9857	0.9776	0.9838	0.9807	0.9986	169.36	0.3692
XGBoost	No FS	0.9431	0.9288	0.9163	0.9225	0.9849	1.23	0.0251
	175/0.7	0.9774	0.9711	0.9676	0.9693	0.9962	11.51	0.0092
RF	No FS	0.9411	0.9422	0.8956	0.9183	0.9816	1.21	0.0152
	175/0.7	0.9737	0.9674	0.9613	0.9643	0.9965	21.03	0.1082
KNN	No FS	0.9644	0.9692	0.9334	0.9509	0.9933	3.39	1.623
	175/0.7	0.9840	0.9741	0.9829	0.9785	0.9963	4.48	0.4595

Among all evaluated classifiers, LSTM and SVM achieve the highest accuracy of 98.57% when trained with the hybrid-selected feature subset, representing a substantial improvement over their respective baseline performances. LSTM attains the best AUC of 99.86%, demonstrating exceptional discriminative capability in distinguishing malware from benign applications. The deep

learning architecture's gated memory mechanism enables effective capture of complex nonlinear feature interactions, contributing to robust classification even though DREBIN-215 consists of static rather than sequential features. When trained without feature selection, LSTM achieves 96.77% accuracy and 98.95% AUC, indicating strong but not optimal performance. The hybrid feature selection improves LSTM's accuracy by 1.80 percentage points and AUC by 0.91 percentage points, while simultaneously reducing testing time from 0.5130 seconds to 0.3692 seconds, a 1.39 fold acceleration. Most notably, recall increases by 4.68 percentage points (from 93.70% to 98.38%), enhancing the model's sensitivity to malware instances, a critical attribute for security applications where false negatives carry higher risk than false positives.

SVM exhibits the most dramatic transformation among all classifiers, improving from complete classification failure to state of the art performance through the application of hybrid feature selection. Without feature selection, SVM achieves only 63.05% accuracy with precision, recall, and F1-score all registering 0.0000, indicating that the model classifies all test samples into a single class despite maintaining reasonable AUC (96.97%). This behavior reflects SVM's severe sensitivity to the curse of dimensionality; in the high-dimensional, redundant 215-feature space, the quadratic programming algorithm fails to identify a meaningful maximum-margin hyperplane, resulting in a degenerate decision boundary. Upon applying the hybrid mRMR correlation approach to reduce the feature space to 140 dimensions, SVM recovers completely, achieving 98.57% accuracy, 97.94% precision, 98.20% recall, 98.07% F1-score, and 99.78% AUC. This represents an accuracy gain of 35.52 percentage points—the largest improvement observed among all models. Furthermore, training time decreases dramatically from 770.75 seconds to 155.88 seconds (4.94 fold faster), and testing time plummets from 15,581 seconds (approximately 4.3 hours) to 0.3824 seconds, delivering an extraordinary 40,745-fold speedup. This transformation demonstrates the critical importance of feature quality for kernel-based methods and positions SVM as one of the top-performing and most computationally efficient classifiers under the proposed framework.

3.2.2. Impact of Hybrid Feature Selection

The experimental comparison clearly demonstrates that the proposed hybrid feature selection strategy produces a substantial positive impact on predictive performance across all evaluated classifiers. By reducing the feature space from 215 dimensions to a compact 140-feature subset, the framework eliminates redundant and weakly informative attributes while preserving discriminative behavioral characteristics of Android applications.

The most significant improvement is observed in the Support Vector Machine model. Without feature selection, SVM achieves only 63.05 percent accuracy, with precision, recall, and F1-score equal to 0.0000. This indicates that the classifier collapses into predicting a single class despite maintaining an AUC of 96.97 percent. Such behavior reflects the sensitivity of margin-based classifiers to high-dimensional feature redundancy, where irrelevant correlations distort the construction of an optimal separating hyperplane. After applying the hybrid mRMR and correlation filtering approach, SVM achieves 98.57 percent accuracy and 99.78 percent AUC, fully restoring its discriminative capability. Precision and recall increase to 97.94 percent and 98.20 percent, respectively, confirming that dimensionality reduction effectively stabilizes the decision boundary.



Figure 9. Accuracy Improvement with Hybrid Feature Selection

The comparative performance results are illustrated in Figure 9. A consistent performance enhancement is also observed in LSTM. Accuracy improves from 96.77 percent to 98.57 percent, while AUC increases from 98.95 percent to 99.86 percent. The recall improvement from 93.70 percent to 98.38 percent is particularly important for malware detection, as it directly reduces false negatives. Ensemble methods such as Random Forest and XGBoost likewise benefit from the refined feature representation, achieving higher accuracy and more balanced precision recall trade offs. KNN also demonstrates improved robustness, indicating that distance-based learners benefit from reduced feature noise.

3.2.3. Computational Efficiency

Beyond predictive accuracy, the hybrid feature selection strategy substantially improves computational efficiency, which is critical for scalable malware detection systems. Reducing the feature dimension from 215 to 140 decreases model complexity and accelerates both training and inference processes.

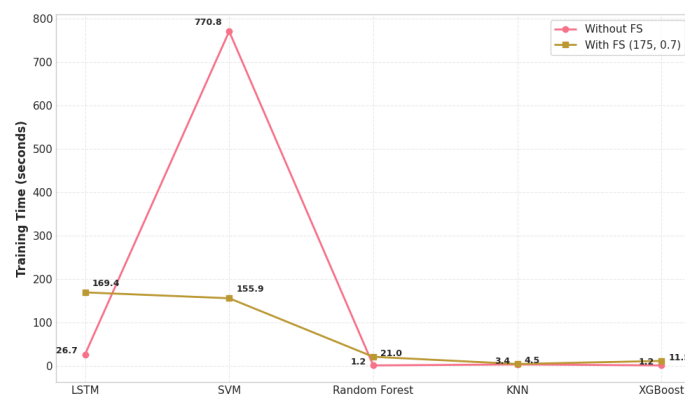


Figure 10. Training Time Comparison

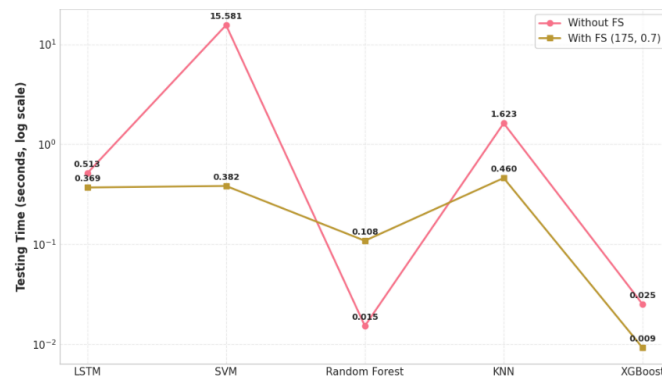


Figure 11. Testing Time Comparison (log scale)

The computational efficiency analysis is presented in Figures 10 and 11. Figure 10 illustrates the training time comparison, where the SVM model exhibits a dramatic reduction from 770.75 seconds to 155.88 seconds after feature selection. Meanwhile, Figure 11 presents inference time on a logarithmic scale to clearly emphasize computational differences. The substantial reduction in SVM testing time from 15.581 seconds to 0.3824 seconds highlights the scalability improvement achieved through dimensionality reduction. This represents approximately 4.94 times faster training and over 40 times faster inference. Such improvement highlights the computational burden imposed by redundant high-dimensional features on kernel-based optimization.

KNN also shows substantial efficiency gains. Testing time decreases from 1.623 seconds to 0.4595 seconds, demonstrating that distance calculations in lower-dimensional space significantly reduce computational overhead. LSTM exhibits reduced testing time from 0.5130 seconds to 0.3692 seconds, while maintaining higher predictive performance. Although LSTM training time

increases due to enhanced representational learning on the refined feature subset, the overall inference efficiency remains improved.

Ensemble methods such as Random Forest and XGBoost maintain competitive training times under both configurations, but still benefit from reduced inference latency when feature selection is applied. These findings confirm that the proposed hybrid approach not only improves classification accuracy but also enhances practical deployability by reducing computational requirements.

In summary, the integration of mRMR ranking and correlation filtering produces a compact and information. Rich feature subset that simultaneously strengthens predictive performance and improves computational efficiency. This dual advantage supports the suitability of the proposed framework for real-world Android malware detection environments where both accuracy and processing speed are essential.

3.3. Comparative Discussion with Related Work

The comparative analysis between the findings of this study and existing literature demonstrates how the proposed hybrid mRMR–CORR feature selection and Bayesian optimization pipeline positions itself within the broader landscape of Android malware detection research. Prior works utilizing deep learning, such as Pachhala et al. [19] and Alhussen [7], achieved very high accuracies of 98% and 99.24%, respectively, driven by more complex LSTM architectures and extensive optimization, whereas the LSTM model in this study, intentionally designed with a moderated configuration, obtains slightly lower accuracy compared to heavily optimized LSTM models, reflecting a deliberate balance between model complexity and generalization, reflecting a deliberate balance between model complexity and generalization. Studies employing feature-selection-driven machine learning, including Kitanovski et al. [20], reported strong precision and recall using correlation-based filtering with XGBoost, which aligns with the improvements observed in this work where the hybrid mRMR–CORR approach enhances classical models efficiently. Similarly, research by Yilmaz and Bakır [1], which integrated SMOTE-Tomek balancing with Random Forest, achieved 98.46% accuracy, and the ensemble framework proposed by Taha and Barukab [21] attained 94.15% accuracy through genetic algorithm optimized classifiers. Although the models in this study do not surpass the highest accuracies reported in prior works, they deliver competitive performance with substantially reduced computational overhead due to the hybrid feature selection, yielding a more efficient and scalable detection pipeline. A detailed comparison of these studies and the results obtained in this work can be seen in Table 3, which highlights the methodological differences and resulting performance outcomes across the referenced literature.

Table 3. Performance Comparison with Related Studies

Ref	Method (Best)	Dataset	Result
[1]	RF + SMOTE-Tomek	DREBIN-215	Acc 98.46%
[7]	Opt LSTM	DREBIN-215	Acc 99.24%
[19]	LSTM	DREBIN-215	Acc 98%
[20]	XGBoost + CORR	DREBIN-215	Preci 94%
[21]	SVM	DREBIN-215	Acc 94.15%
This Study	LSTM + mRMR + CORR	DREBIN-215	Acc 98.57%

4. CONCLUSION

This study addressed the challenge of high-dimensional feature representation in Android malware detection, which can lead to model instability, performance degradation, and excessive computational cost. As outlined in the Introduction, the primary objective was to develop an optimized detection framework capable of improving predictive performance while maintaining computational efficiency. To achieve this objective, a hybrid feature selection strategy integrating Minimum Redundancy Maximum Relevance and correlation filtering was systematically evaluated on the Drebin-215 dataset using five classifiers: LSTM, SVM, Random Forest, KNN, and XGBoost.

The experimental results confirm that the proposed framework successfully fulfills the research objective. Dimensionality reduction improves classification robustness and enhances

predictive performance across all evaluated models. Among them, LSTM achieves the highest overall predictive performance, reaching 98.57% accuracy and 99.86% AUC, along with the highest recall and F1-score. These results indicate strong discriminative capability and balanced sensitivity in detecting malware instances. SVM demonstrates the most substantial recovery after feature selection, improving from severe performance degradation to competitive classification accuracy.

From a computational perspective, the hybrid feature selection approach significantly reduces processing time. SVM training time decreases from 770.75 seconds to 155.88 seconds, while testing time drops from 15.581 seconds to 0.3824 seconds. Although LSTM achieves superior predictive metrics, XGBoost demonstrates the fastest inference time at 0.0092 seconds, making it particularly suitable for real-time or resource-constrained deployment environments.

Overall, the integration of mRMR and correlation filtering provides an effective balance between predictive performance and computational efficiency. LSTM is identified as the best-performing model in terms of classification accuracy and robustness, while XGBoost offers the most efficient deployment profile.

Future research may explore adaptive feature selection thresholds, integration with dynamic or behavioral features, and evaluation on larger and more heterogeneous malware datasets to further improve generalization across evolving threat landscapes. Additionally, real-world deployment validation may provide deeper insight into operational scalability and long-term robustness.

ACKNOWLEDGEMENTS

The author would like to acknowledge the guidance and constructive feedback provided by the supervisor throughout the research process. The author also gratefully acknowledges the support and facilities provided by Lambung Mangkurat University that contributed to the completion of this study.

REFERENCES

- [1] E. K. Yılmaz and H. Bakır, "Hyperparameter tuning and feature selection methods for malware detection," *Journal of Polytechnic*, vol. 27, no. 1, pp. 343–353, 2024, doi: 10.2339/politeknik.1243881.
- [2] A. Dağlıoğlu and İ. A. Doğru, "Android İşletim Sisteminde Kötücül Yazılım Tespit Sistemleri," *DÜMF Mühendislik Dergisi*, vol. 11, no. 2, pp. 499–511, 2020, doi: 10.24012/dumf.559205.
- [3] McAfee, "Mobile Threat Report" 2021. [Online]. Available: <https://www.mcafee.com/content/dam/global/infographics/McAfeeMobileThreatReport2021.pdf>.
- [4] C. S. Yadav, J. Singh, A. Yadav, H. S. Pattanayak, R. Kumar, A. A. Khan, M. A. Haq, A. Alhussen, and S. Alharby, "Malware analysis in IoT & Android systems," *Electronics*, vol. 11, no. 15, p. 2354, 2022, doi: 10.3390/electronics11152354.
- [5] A. Al-Marghilani, "Comprehensive analysis of IoT malware evasion techniques," *Engineering, Technology & Applied Science Research*, vol. 11, no. 4, pp. 7495–7500, 2021, doi: 10.48084/etasr.4296.
- [6] V. Kouliaridis, G. Kambourakis, D. Geneiatakis, and N. Potha, "Two anatomists are better than one—Dual-level Android malware detection," *Symmetry*, vol. 12, p. 1128, 2020, doi: 10.3390/sym12071128.
- [7] A. Alhussen, "Advanced Android malware detection through deep learning optimization," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14552–14557, 2024, doi: 10.48084/etasr.7443.
- [8] G. A. Pradipta, R. Wardoyo, A. Musdholifah, and I. N. Sanjaya, "Radius-SMOTE: A new oversampling technique of minority samples based on radius distance for learning from imbalanced data," *IEEE Access*, vol. 9, pp. 74763–74777, 2021, doi: 10.1109/ACCESS.2021.3080316.

- [9] M. Kamaladevi, V. Venkataraman, and K. Sekar, "Tomek link undersampling with stacked ensemble classifier for imbalanced data classification," *Annals of the Romanian Society for Cell Biology*, vol. 25, no. 4, pp. 2182–2190, 2021. [Online]. Available: <http://annalsofrscb.ro/index.php/journal/article/view/2751/2283>
- [10] Z. Xu, D. Shen, T. Nie, and Y. Kou, "A hybrid sampling algorithm combining M-SMOTE and ENN based on random forest for medical imbalanced data," *Journal of Biomedical Informatics*, vol. 107, p. 103465, Jul. 2020, doi: 10.1016/j.jbi.2020.103465.
- [11] H. Hairani, A. Anggrawan, and D. Priyanto, "Improvement performance of the random forest method on unbalanced diabetes data classification using SMOTE-Tomek link," *International Journal on Informatics Visualization*, vol. 7, no. 1, pp. 258–264, 2023, doi: 10.30630/joiv.7.1.1069.
- [12] S. H. Imanuddin, K. Adi, and R. Gernowo, "Sentiment analysis on Satusehat application using support vector machine method," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 5, no. 3, pp. 143–149, 2023, doi: 10.35882/jeemi.v5i3.304.
- [13] A. Dhiyaussalam, F. A. Wibowo, E. A. Sarwoko, and I. M. A. Setiawan, "Classification of headache disorder using random forest algorithm," in *Proc. 4th Int. Conf. Informatics and Computational Sciences (ICICoS)*, 2020, doi: 10.1109/ICICoS51170.2020.9299105.
- [14] G. Mitrentsis and H. Lens, "An interpretable probabilistic model for short-term solar power forecasting using natural gradient boosting," *Applied Energy*, vol. 309, p. 118473, Mar. 2022, doi: 10.1016/j.apenergy.2021.118473.
- [15] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, no. 3, 2020, doi: 10.1007/s10462-020-09896-5.
- [16] R. Huang et al., "Well performance prediction based on long short-term memory (LSTM) neural network," *Journal of Petroleum Science and Engineering*, vol. 208, p. 109686, Jan. 2022, doi: 10.1016/j.petrol.2021.109686.
- [17] W. Zhang, C. Wu, H. Zhong, Y. Li, and L. Wang, "Prediction of undrained shear strength using extreme gradient boosting and random forest based on Bayesian optimization," *Geoscience Frontiers*, vol. 12, pp. 469–477, 2021, doi: 10.1016/j.gsf.2020.03.007.
- [18] L. D. Hansen, M. Stokholm-Bjerregaard, and P. Durdevic, "Modeling phosphorous dynamics in a wastewater treatment process using Bayesian optimized LSTM," *Computers and Chemical Engineering*, vol. 160, p. 107738, 2022, doi: 10.1016/j.compchemeng.2022.107738.
- [19] N. Pachhala, S. Jothilakshmi, and B. P. Battula, "Android malware classification using LSTM model," *Revue d'Intelligence Artificielle*, vol. 36, no. 5, pp. 761–767, Dec. 2022, doi: 10.18280/ria.360514.
- [20] A. Kitanovski, H. M. Trpcheska, and V. Dimitrova, "Detecting malware in Android applications using XGBoost," in *Proc. 20th Int. Conf. Informatics Inf. Technol.*, 2023. [Online]. Available: <http://hdl.handle.net/20.500.12188/27385>.
- [21] A. Taha and O. Barukab, "Android malware classification using optimized ensemble learning based on genetic algorithms," *Sustainability*, vol. 14, no. 21, p. 14406, Nov. 2022, doi: 10.3390/su142114406.
- [22] S. Y. Yerima and S. Sezer, "DroidFusion: A novel multilevel classifier fusion approach for Android malware detection," *IEEE Transactions on Cybernetics*, vol. 49, no. 2, pp. 453–466, Feb. 2019, doi: 10.1109/TCYB.2017.2777960.
- [23] G. Tamami, W. A. Triyanto, and S. Muzid, "Sentiment Analysis Mobile JKN Reviews Using SMOTE Based LSTM," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 19, no. 1, pp. 13–24, Jan. 2025, doi: 10.22146/ijccs.101910
- [24] E. F. Swana, W. Doorsamy, and P. Bokoro, "Tomek link and SMOTE approaches for machine fault classification with an imbalanced dataset," *Sensors*, vol. 22, no. 9, p. 3246, Jan. 2022, doi: 10.3390/s22093246.
- [25] A. Khleel and K. Nehéz, "A novel approach for software defect prediction using CNN and GRU based on SMOTE Tomek method," *Journal of Intelligent Information Systems*, vol. 60, no. 3, pp. 673–707, May 2023, doi: 10.1007/s10844-023-00793-1.

- [26] E. Başaran, "A new brain tumor diagnostic model: Selection of textural feature extraction algorithms and convolution neural network features with optimization algorithms," *Computers in Biology and Medicine*, vol. 148, p. 105857, Sep. 2022, doi: 10.1016/j.compbiomed.2022.105857.
- [27] R. Chen, C. Dewi, S. Huang and R. E. Caraka, "Selecting critical features for data classification based on machine learning methods," *Journal of Big Data*, vol. 7, no. 1, Jul. 2020, doi: 10.1186/s40537-020-00327-4.
- [28] J. Hermo, V. Bolón-Canedo, and S. Ladra, "Fed-mRMR: A lossless federated feature selection method," *Information Sciences*, vol. 669, p. 120609, Apr. 2024, doi: 10.1016/j.ins.2024.120609.
- [29] G. Wang, F. Lauri, and A. H. E. Hassani, "Feature selection by mRMR method for heart disease diagnosis," *IEEE Access*, vol. 10, pp. 100786–100796, Jan. 2022, doi: 10.1109/ACCESS.2022.3207492.
- [30] S. Xie, Y. Zhang, D. Lv, X. Chen, J. Lu, and J. Liu, "A new improved maximal relevance and minimal redundancy method based on feature subset," *J. Supercomput.*, vol. 79, no. 3, pp. 3157–3180, Aug. 2022, doi: 10.1007/s11227-022-04763-2.
- [31] M. Agar, S. Aydin, M. Cakmak, M. Koc, and M. Togacar, "Detection of Thymoma Disease Using mRMR Feature Selection and Transformer Models," *Diagnostics*, vol. 14, no. 19, p. 2169, Sep. 2024, doi: 10.3390/diagnostics14192169.
- [32] M. Toğaçar, B. Ergen, and Z. Cömert, "Detection of lung cancer on chest CT images using minimum redundancy maximum relevance feature selection method with convolutional neural networks," *Biocybern. Biomed. Eng.*, Nov. 2019, doi: 10.1016/j.bbe.2019.11.004.
- [33] E. S. Alomari *et al.*, "Malware Detection Using Deep Learning and Correlation-Based Feature Selection," *Symmetry*, vol. 15, no. 1, p. 123, Jan. 2023, doi: 10.3390/sym15010123.
- [34] D. C. E. Saputra, Y. Maulana, T. A. Win, R. Phann, and W. Caesarendra, "Implementation of Machine Learning and Deep Learning Models Based on Structural MRI for Identification Autism Spectrum Disorder," *J. Ilm. Tek. Elektro Komput. Inform.*, vol. 9, no. 2, pp. 307–318, May 2023, doi: 10.26555/jiteki.v9i2.26094.
- [35] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning," *Stata J.*, vol. 20, no. 1, pp. 3–29, Mar. 2020, doi: 10.1177/1536867x20909688.
- [36] A. Tajali, T. H. Saragih, M. I. Mazdadi, I. Budiman and A. Farmadi, "The Impactness of SMOTE as Imbalance Class Handling for Myocardial Infarction Complication Classification using Machine Learning Approach with Data Imputation and Hyperparameter," *Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics*, pp. Vol. 6, No. 4, pp. 227-239, 2024, doi: 10.35882/ijeemi.v6i4.13
- [37] S. Talukdar *et al.*, "Land-Use Land-Cover Classification by Machine Learning Classifiers for Satellite Observations—A Review," *Remote Sensing*, vol. 12, no. 7, p. 1135, Apr. 2020, doi: 10.3390/rs12071135.
- [38] T. Chen and C. Guestrin, "XGBoost: a Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, vol. 1, no. 1, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.
- [39] A. Asselman, M. Khaldi, and S. Aammou, "Enhancing the prediction of student performance based on the machine learning XGBoost algorithm," *Interactive Learning Environments*, vol. 31, no. 6, pp. 1–20, May 2021, doi: 10.1080/10494820.2021.1928235.
- [40] Y. Tang, Y.-C. Chang, and K. Li, "Applications of K-nearest neighbor algorithm in intelligent diagnosis of wind turbine blades damage," *Renewable Energy*, vol. 212, pp. 855–864, Aug. 2023, doi: 10.1016/j.renene.2023.05.087.
- [41] Y. Hamed, A. Ibrahim Alzahrani, A. Shafie, Z. Mustaffa, M. Che Ismail, and K. Kok Eng, "Two steps hybrid calibration algorithm of support vector regression and K-nearest neighbors," *Alexandria Engineering Journal*, vol. 59, no. 3, pp. 1181–1190, Jun. 2020, doi: 10.1016/j.aej.2020.01.033.

- [42] B. Du, S. Wang, N. Wang, L. Zhang, D. Tao, and L. Zhang, "Hyperspectral signal unmixing based on constrained non-negative matrix factorization approach," *Neurocomputing* (Amsterdam), vol. 204, pp. 153–161, Sep. 2016, doi: 10.1016/j.neucom.2015.10.132.
- [43] H. A. Abu Alfeilat et al., "Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review," *Big Data*, vol. 7, no. 4, pp. 221–248, Dec. 2019, doi: 10.1089/big.2018.0175.
- [44] I. Lee and C. Torpelund-Bruin, "Geographic knowledge discovery from Web Map segmentation through generalized Voronoi diagrams," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9376–9388, Aug. 2012, doi: 10.1016/j.eswa.2012.02.129.
- [45] P. Cunningham and S. J. Delany, "k-Nearest Neighbour Classifiers - A Tutorial," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–25, Jul. 2021, doi: 10.1145/3459665.
- [46] S. Abbaspour, F. Fotouhi, A. Sedaghatbaf, H. Fotouhi, M. Vahabi, and M. Linden, "A Comparative Analysis of Hybrid Deep Learning Models for Human Activity Recognition," *Sensors*, vol. 20, no. 19, p. 5707, Oct. 2020, doi: 10.3390/s20195707.
- [47] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [48] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850v5*, Jun. 2014, doi: 10.48550/arXiv.1308.0850.
- [49] S. Minaee, E. Amini, and A. A. Abdolrashidi, "Deep-sentiment: Sentiment analysis using ensemble of CNN and Bi-LSTM models," *arXiv preprint arXiv:1904.04206v1*, Apr. 2019, doi: 10.48550/arXiv.1904.04206.
- [50] W. Fang, Y. Chen, and Q. Xue, "Survey on Research of RNN-Based Spatio-Temporal Sequence Prediction Algorithms," *Journal on Big Data*, vol. 3, no. 3, pp. 97–110, 2021, doi: 10.32604/jbd.2021.016993.
- [51] D. Gaur and S. Kumar Dubey, "Develompent of Activity Recognition Model using LSTM-RNN Deep Learning Algorithm," *Journal of information and organizational sciences*, vol. 46, no. 2, pp. 277–291, Dec. 2022, doi: 10.31341/jios.46.2.1.