Matrik: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer
Vol. 25, No. 2, March 2026, pp. 367∼384
ISSN: 2476-9843, accredited by Kemenristekdikti, Decree No: 10/C/C3/DT.05.00/2025
DOI: 10.30812/matrik.v25i2.6198

Addressing Class Imbalance in Android Backdoor Malware Detection Using Ensemble Models

Rama Aria Megantara[1], Dewi Pergiwati[2], Farrikh Alzami[1], Ricardus Anggi Pramunendar[1], Dwi Puji Prabowo[1], Muhammad Naufal[1], Rivaldo Mersis Brilianto[1]
[1]Universitas Dian Nuswantoro, Semarang, Indonesia
[2]Pusan National University, Busan, Republic of Korea

Article Info

Article history:

Received February 20, 2026
Revised March 11, 2026
Accepted March 19, 2026

Keywords:

Android Malware Detection;
Backdoor Classification;
Class Imbalance;
Static Analysis;
Random Forest.

ABSTRACT

Backdoor malware represents one of the most critical threats in the Android ecosystem due to its capability to enable covert remote access, escalate privileges, and exfiltrate sensitive data without user awareness. Although the CCCS-CIC-AndMal-2020 dataset is publicly available, prior studies have not specifically formulated Backdoor detection as a binary classification problem under extreme class imbalance, nor systematically evaluated the impact of oversampling and cost-sensitive weighting using imbalance-aware performance metrics. This study proposes a comprehensive detection pipeline that integrates ensemble learning, class imbalance handling strategies, and explainability-based analysis to extract behavioral signatures of Backdoor malware. A two-stage feature selection process is employed to reduce the original 9,502-dimensional feature space to 500 informative features. Subsequently, five classification algorithms are evaluated under three imbalance-handling scenarios using a composite ranking criterion based on F1-score, Area Under the Receiver Operating Characteristic Curve (AUC), Geometric Mean (G-Mean), and Matthews Correlation Coefficient (MCC). The experimental results demonstrate that the Random Forest model combined with Synthetic Minority Oversampling Technique (SMOTE) achieves the best performance, with an F1-score of 0.9043, AUC of 0.9909, G-Mean of 0.9422, and MCC of 0.8948. Furthermore, SHAP analysis identifies 39 Android permissions related to account access, covert communication, and privilege escalation as key behavioral signatures, with the permissions feature group contributing 2.31 times higher discriminative importance than non-permission features. These findings indicate that interpretable ensemble learning not only improves detection performance but also provides actionable insights for static malware analysis.

Corresponding Author:

Rama Aria Megantara, +62 811-2740-768
Faculty of Computer Science, Information Engineering,
Universitas Dian Nuswantoro, Semarang, Indonesia,
Email: aria@dsn.dinus.ac.id

How to Cite:
R. A. Megantara, "Addressing Class Imbalance in Android Backdoor Malware Detection Using Ensemble Models", *MATRIK: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer*, Vol. 25, No. 2, pp. 367-384, March, 2026.

1. INTRODUCTION

Android remains the dominant mobile platform globally, accounting for over 70% of active smartphone deployments. This ubiquity makes the platform a persistent target for malware development [1]. The CCCS-CIC-AndMal-2020 dataset, produced by the Canadian Institute for Cybersecurity, represents one of the largest and most structurally complete publicly available Android malware benchmarks, cataloging fourteen distinct malware categories extracted through static analysis of AndroidManifest.xml declarations [2].

Among these categories, Backdoor malware occupies a threat class of particular severity [3]. Unlike adware or scareware, whose impact is manifest to the device user, Backdoor applications operate covertly by establishing persistent remote access channels [4], harvesting account credentials [5], exfiltrating device identifiers, and enabling arbitrary payload execution under the direction of a remote operator. These capabilities render Backdoor samples operationally dangerous regardless of the specific payload deployed after installation. The Backdoor category is also the smallest in the dataset, comprising 1,538 of 342,169 samples, which produces a class imbalance ratio of 1:221.5. This extreme imbalance substantially degrades the detection rate of minority-class instances when standard classifiers are applied without appropriate compensation mechanisms [6].

Prior work using the CCCS-CIC-AndMal-2020 dataset has approached the detection problem from two directions. Multi-class classification studies, such as the multi-label analysis by Xie et al [7], treat all malware categories simultaneously and report aggregate metrics, which obscure the performance on individual minority categories. Single-category focused studies, such as the Backdoor detection work by Khan et al. [8] in the IoT context, do not evaluate the CCCS-CIC-AndMal-2020 static feature representation. Studies that do evaluate this dataset, including the dynamic feature classification by Alsraratee and Al-Azawei [9] and the LightGBM-based detection by Zhou et al. [10], report category-level accuracy rather than imbalance-aware metrics such as G-Mean and MCC, which are more appropriate for evaluating the detection of rare threat categories.

Notwithstanding this body of work, three interconnected gaps (research gaps) remain unaddressed. No existing study has formulated Backdoor detection on the CCCS-CIC-AndMal-2020 dataset as a focused binary classification problem and reported performance using imbalance-aware metrics such as G-Mean and MCC as primary criteria, making it impossible to assess the actual detection capability for this threat class in isolation. The comparative effect of oversampling combined with classifier-level cost-sensitive weighting on Backdoor-specific detection rate has not been quantified in this dataset. Furthermore, SHAP-based feature attribution at the Backdoor class level has not been performed on this dataset, which limits the translation of detection models into operationally actionable knowledge for security practitioners. The central hypothesis of this study is that Backdoor Android applications exhibit a statistically distinguishable static permission profile that enables accurate binary classification under extreme imbalance conditions when appropriate oversampling and feature selection are applied. To test this hypothesis, the study makes four contributions. First, it provides the first Backdoor-specific binary classification analysis on the CCCS-CIC-AndMal-2020 dataset using the full static feature representation, with G-Mean and MCC as primary imbalance-aware evaluation metrics. Second, it introduces a composite ranking criterion over F1, AUC-ROC, and G-Mean for best model selection, addressing the tendency of single-metric selection to produce models that sacrifice specificity under class imbalance. Third, it demonstrates through a ten-seed stability analysis that SMOTE oversampling alone achieves a more favorable precision-recall balance than SMOTE combined with cost-sensitive weighting for Random Forest and LightGBM on this dataset. Fourth, it produces a named behavioral signature of 39 Android permissions associated with account access, covert messaging, and privilege escalation, providing directly actionable inputs for signature-based static analysis tools.

Static analysis of AndroidManifest.xml content provides a practical detection foundation because it does not require application execution, scales efficiently to large volumes of applications, and exposes the declared intent structure of an application before any dynamic behavior occurs. Bansal et al. demonstrated that a compact set of static features selected through optimal-flow ranking is sufficient for high-accuracy malware detection on the CIC-InvesAndMal-2019 dataset [11]. Pudke and Bansal [12] proposed a stacked ensemble framework combining semantic permission aggregation with explainable AI, achieving an F1-score of 0.9847 on the TUANDROMD dataset. They confirmed that permission-derived features dominate feature importance rankings. Saeed et al. [13] proposed a feature-ranked hybrid framework for Android IoT malware detection, demonstrating that a hierarchical selection pipeline significantly reduces false-negative rates under imbalanced deployment conditions. Class imbalance is a structural property of malware datasets because benign applications vastly outnumber malware samples in real application corpora. SMOTE generates synthetic minority class samples by linearly interpolating between existing instances and their k-nearest neighbors in feature space, improving minority class recall without sacrificing majority class specificity when properly calibrated. Zhu et al. [6] demonstrated that multi-model ensemble approaches combined with oversampling significantly improve detection rates for rare malware categories, with SMOTE providing consistent improvements over no-handling baselines across multiple classifier types. Li et al. [14] proposed SynDroid, which uses a Conditional Tabular GAN for synthetic sample generation and shows improvements over SMOTE in highly imbalanced settings. Cost-sensitive classification provides a complementary mechanism by assigning asymmetric misclas-

sification penalties at training time without modifying the data distribution, and its combination with oversampling has been shown to produce additive improvements in specific classifiers [6]. The adoption of machine learning for security-critical decisions requires interpretability mechanisms that translate model outputs into actionable knowledge. SHAP [15], based on Shapley values from cooperative game theory, provides theoretically grounded and locally consistent feature attribution. Soi et al. [16] applied SHAP to function call graph-based malware detection, demonstrating that SHAP attribution identifies subgraph motifs corresponding to known malicious code patterns. Patel and Ghosh [17] proposed EML-AMD, an explainable machine learning framework for adaptive Android malware detection, showing that SHAP-identified permission clusters align with documented malware behavior families. The present study extends this line of research by applying SHAP specifically to the binary Backdoor detection problem and extracting a behavioral signature from true positive detections, which differs from global feature importance in that it isolates features contributing to correct Backdoor identification rather than overall model decisions. Several studies have used the CCCS-CIC-AndMal-2020 dataset for Android malware classification. Zhou et al. [10] proposed IZOA-LightGBM, an optimized LightGBM model achieving 98.86% multi-class accuracy, reporting class-level metrics only in aggregate. Alsraratee and Al-Azawei [9] evaluated Random Forest, Decision Tree, and K-Nearest Neighbor with Mutual Information and PCA for dynamic feature classification, achieving 98% overall accuracy on fourteen categories but not isolating rare-category detection rates. Xie et al. [7] demonstrated the multi-label phenomenon in this dataset through a comprehensive 15-dataset analysis, noting that individual application samples may belong to multiple attack categories simultaneously. None of these studies formulate Backdoor detection as a focused binary problem or report G-Mean and MCC as evaluation criteria, which defines the gap that the present study addresses.

2. RESEARCH METHOD

Figure 1 presents a schematic overview of the complete detection pipeline from raw dataset ingestion through behavioral signature extraction. The pipeline is designed as a sequential flow to ensure reproducibility and systematic evaluation across multiple experimental conditions. It begins with binary label assignment on the full CCCS-CIC-AndMal-2020 dataset, followed by a two-stage feature selection procedure that progressively reduces the 9,502-dimensional feature space to 500 informative static features. Three parallel imbalance-handling conditions are then applied to the training data, and five classifiers are independently trained and evaluated across five random seeds. The best-performing model is subsequently identified through a composite ranking criterion and subjected to SHAP TreeExplainer analysis to extract a named behavioral signature of Android permissions.

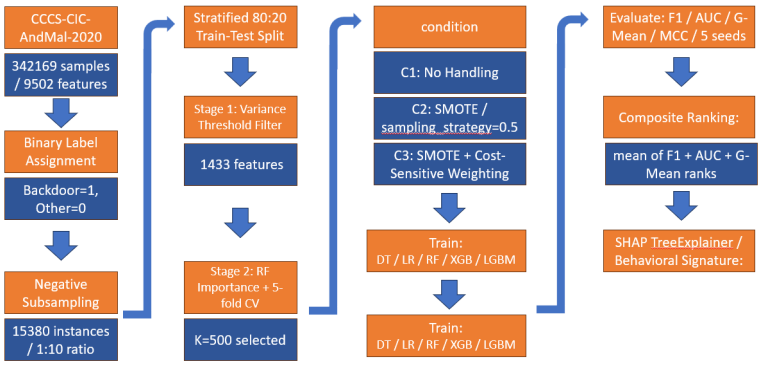

Figure 1. Overview of the detection pipeline from data ingestion to behavioral signature extraction

The pipeline begins with the full CCCS-CIC-AndMal-2020 dataset comprising 342,169 samples and 9,502 static features, from which binary labels are assigned by designating Backdoor applications as the positive class and all remaining samples as the negative class. A controlled subsampling step reduces the negative class to 15,380 instances, after which a stratified 80:20 train-test split produces fixed training and test partitions. Feature selection proceeds in two stages: a variance threshold filter retains 1,433 non-constant features, and a Random Forest importance ranking with five-fold cross-validation selects the optimal 500-feature subset. The filtered training set is then passed through three parallel condition branches: no imbalance handling, SMOTE oversampling, and SMOTE combined with cost-sensitive weighting. Five classifiers are trained under each condition and evaluated across five random seeds, producing a 15-entry performance matrix from which the best model is identified through the composite ranking criterion defined in Equation 1. The selected model is subsequently used for SHAP TreeExplainer analysis, yielding the behavioral signature that constitutes the final interpretability output of the pipeline. The pipeline is implemented in Python 3.10 on Kaggle's

CPU infrastructure with 30 GB of RAM, and all intermediate datasets are persisted in Parquet format with int8 dtype for binary columns.

2.1. Dataset

This study uses the CCCS-CIC-AndMal-2020 dataset [2]. The dataset contains applications from 200,000 malicious samples across fourteen malware categories and 200,000 benign samples sourced from the Androzoo repository. Each application is represented as a vector of 9,503 elements derived from AndroidManifest.xml analysis using the AndroidAppLyzer tool. The first element is a unique application hash identifier, which is removed during preprocessing, leaving 9,502 feature columns encoding the presence of declared permissions, services, intent actions, and intent categories.

Binary labels are assigned as follows: applications in the Backdoor category receive label 1, and all other applications, including those in other malware categories and benign samples, receive label 0. The Backdoor samples number 1,538, and the negative class comprises 340,631 instances, yielding a raw imbalance ratio of 1:221.5. This formulation treats non-Backdoor malware and benign applications collectively as the negative class, reflecting the operational detection objective of identifying Backdoor applications among all incoming application submissions regardless of their other threat characteristics.

2.2. Preprocessing and Subsampling

After removing the hash column and verifying the absence of null values, the full negative class of 340,631 samples is subsampled to 15,380 instances using stratified random sampling with a fixed random state of 42. This produces a working dataset of 16,918 samples at a 1:10 imbalance ratio. The subsampling is applied to make training computationally tractable across the full experimental grid of five classifiers, three conditions, and five random seeds, while retaining sufficient negative class diversity. The 1:10 ratio is documented as a controlled experimental parameter, and its effect on the reported metrics is explicitly discussed in the Limitations Section. A stratified train-test split with an 80:20 ratio is applied to the working dataset, yielding 13,534 training samples and 3,384 test samples. The test set is held out throughout all training, cross-validation, and oversampling operations. Table 1 summarizes the sample distribution at each pipeline stage. Table 1 shows that the Backdoor class is preserved at its original count throughout all subsampling operations, confirming that the only source of reduction in Backdoor sample volume is the stratified split.

Table 1. Sample of Table

Stage	Total	Backdoor	Other
Full Dataset	342.169	1.538	340.631
After Subsampling	16.918	1.538	15.38
Training Set	13.534	1.23	12.304
Test Set	3.384	308	3.076

2.3. Two-Stage Feature Selection

Feature selection proceeds in two stages. In the first stage, a variance threshold filter with a threshold of zero removes features whose variance is zero across the entire training set, retaining 1.433 of 9.502 features. Features with zero variance are constant across all training samples and carry no discriminative information. In the second stage, a Random Forest classifier with 100 trees and a fixed random state of 42 is trained on the variance-filtered training set to compute Mean Decrease in Impurity importance scores for all 1,433 retained features. Features are ranked in descending order of importance. The optimal number of top-K features is determined by five-fold stratified cross-validation with macro F1-score as the selection criterion, evaluated over K $\epsilon\{100,300,500,1000,2000\}$. Cross-validation uses a secondary Random Forest with 100 trees per fold. The selected K features are extracted from both the training and test sets and stored in Parquet format for subsequent use.

2.4. Class Imbalance Handling Conditions

Three experimental conditions are evaluated. Condition C1 applies no imbalance handling to the training set and provides a lower-bound reference. Condition C2 applies SMOTE [18] to the training set with sampling_strategy=0.5, targeting a 1:2 positive-to-negative ratio in the resampled training set, with k=5 nearest neighbors for interpolation. Condition C3 combines SMOTE with classifier-level cost-sensitive weighting: for classifiers that support class_weight (specifically Decision Tree, Logistic Regression,

Random Forest, and LightGBM), the balanced weighting scheme sets class weights inversely proportional to class frequencies. For XGBoost, which does not expose class_weight, the scale_pos_weight parameter is set to the negative-to-positive ratio computed dynamically from the post-SMOTE training distribution at each random seed, accounting for SMOTE stochasticity. SMOTE is applied exclusively to the training set in both C2 and C3; the test set distribution is never modified.

2.5. Classifiers and Hyperparameters

Five classifiers are evaluated across all three conditions. Table 2 documents the key hyperparameters used in this study. Default values from the respective library implementations are applied to all parameters not listed in the table, to maintain comparability across classifiers and enable direct reproduction of results. Each classifier-condition combination is trained with five random seeds drawn from the set 42,0,1,7,99. The random state controls all stochastic components of the corresponding classifier, including node splitting for Decision Tree and Random Forest, and tree construction for XGBoost and LightGBM. SMOTE sampling uses the same seed as the classifier in each repetition.

Table 2. Classifier Hyperparameters Used in all Experiments

Classifier	Key Parameters	Library
Decision Tree (DT)	criterion=Gini, max_depth=None	Scikit-learn 1.3
Logistic Regression (LR)	Solver=saga, max_iter=3000, C=100	Scikit-learn 1.3
Random Forest (RF)	n_estimator=100, criterion=gini	Scikit-learn 1.3
XGBoost (XGB	N_estimator=200, use_label_encoder=False	Xgboost 2.0
LightGBM (LGBM)	N_estimator=100, boosting_type=gbdt	Lightgbm 4.1

2.6. Evaluation Metrics

Six metrics are computed for each combination. Precision, recall, and F1-score target the positive (Backdoor) class exclusively. AUC-ROC is computed from classifier probability outputs using one-versus-rest aggregation. G-Mean is defined as $\sqrt{sensitivity \times specificity}$, which simultaneously captures the balance between detection rate and false positive rate, and is robust to class imbalance because it penalizes both types of error proportionally. MCC accounts for all four confusion matrix entries and is particularly resistant to the masking effect of majority class dominance on aggregate accuracy. Each metric is reported as mean and standard deviation across ten seeds.

2.7. Composite Best Model Selection

Given the multi-dimensional nature of classification performance under class imbalance, a composite ranking criterion is applied rather than single-metric selection. For each of the fifteen classifier-condition combinations, independent rank positions are assigned for F1-score, AUC-ROC, and G-Mean across all fifteen entries, where rank 1 denotes the highest observed value. The composite rank score is computed as the arithmetic mean of these three independent rank positions, as defined in Equation 1.

$$r_{composite} = \frac{r_{F1} + r_{AUC} + r_{GMan}}{3} \tag{1}$$

2.8. SHAP Analysis and Behavioral Signature Extraction

SHAP TreeExplainer is applied to the best model retrained on the full training set under its optimal condition. For Random Forest, the SHAP library version used returns a three-dimensional array of shape $(n_{samples}, n_{features}, n_{classes})$, the positive class slice at index 1 is extracted to obtain a two-dimensional SHAP matrix. The Backdoor behavioral signature is defined as the set of features whose mean SHAP value across true-positive test samples is strictly positive, ranked by this mean. This definition differs from global feature importance because it isolates features that consistently increase the Backdoor detection probability among correctly classified instances, filtering out features that are globally informative but directionally ambiguous.

Feature column indices in the signature are mapped to two groups based on the verified Permissions list boundary. The Permissions group spans column indices 0 through 3,267, corresponding to the 3,268 entries in the 1_List_Permissions.csv reference file. All remaining columns, indices 3,268 through 9,501, are assigned to the Non-Permissions group. Sub-group boundaries for

services, intent actions, and intent categories cannot be determined independently because the total entries across the four Unique Lists files exceed the dataset's feature count, a structural limitation acknowledged in the dataset documentation.

2.9. Experimental Setup

All experiments are executed in a Kaggle Notebook environment on CPU hardware with 30 GB RAM and Python 3.10. The library stack consists of pandas 2.0, numpy 1.24, scikit-learn 1.3, imbalanced-learn 0.11, xgboost 2.0, lightgbm 4.1, shap 0.43, and matplotlib 3.7. The random state is fixed at 42 for the train-test split, variance filter, and Random Forest feature importance step. Model training across seeds uses the seed value as the random state for all stochastic components of the classifier. SMOTE is applied only to the training partition in each fold and seed repetition; the held-out test set is never resampled or modified.

The total experimental grid comprises 15 classifier-condition combinations, each repeated across ten seeds (0, 1, 7, 13, 21, 37, 42, 55, 77, 99), yielding 150 independent training-evaluation runs. Performance metrics are computed using the *sklearn.metrics* module with *zero_division*=0 for precision and F1 to handle edge cases in folds where the positive class is not predicted. G-Mean is computed from the confusion matrix as $\sqrt{\frac{TP}{(TP+FN)} \times \frac{TN}{TN+FP}}$, where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives respectively.

3. RESULT AND ANALYSIS

3.1. Feature Selection

The variance threshold filter reduces the feature count from 9,502 to 1,433, removing 8,070 constant features. This indicates that the majority of static feature columns are never declared by any application in the working sample, consistent with the sparse high-dimensional structure of AndroidManifest.xml feature vectors. The high proportion of removed features, approximately 84.9% of the original space, reflects the declarative nature of Android permission manifests, where most declared capability slots remain unused across typical application corpora. Retaining only variant features ensures that subsequent importance ranking operates on a non-degenerate feature space, free of noise introduced by constant columns. This stage also reduces memory and computation requirements for the downstream Random Forest importance training step.

Figure 2 presents the five-fold cross-validation macro F1-score as a function of candidate subset size K. The performance profile is notably flat across all evaluated K values, spanning a narrow range of 0.0017 between the lowest and highest observed means. This flatness indicates that the most discriminative information for Backdoor detection is concentrated within a compact high-importance subset, and features ranked beyond this core contribute marginal or redundant signal to the classification objective. The error bars at each K point reflect the standard deviation across five cross-validation folds, confirming that the variance remains low and stable across the evaluated range. The curve does not exhibit monotonic increase or decrease, ruling out underfitting at small K and overfitting at large K within the tested range.

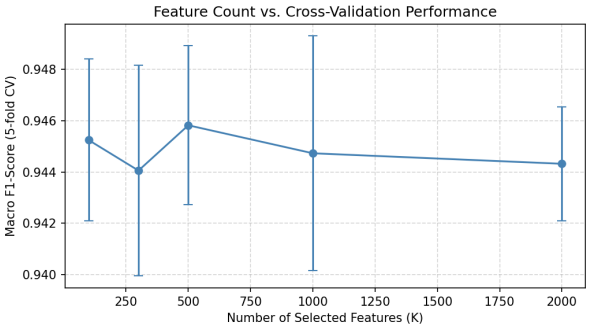

Figure 2. Cross-validation macro F1-score for candidate feature subset sizes K

Table 3 reports the cross-validation macro F1 mean and standard deviation for each candidate K value. K=500 achieves the highest cross-validation macro F1 of 0.9458 with a standard deviation of 0.0031, marginally outperforming K=100 at 0.9452 and K=1000 at 0.9447. The standard deviations across all K values remain below 0.005, indicating that model performance is stable and not sensitive to the exact choice of subset size within the evaluated range. K=500 is selected as the optimal subset size because it

achieves the highest cross-validation score while providing broader feature coverage than K=100, which is particularly important for SHAP attribution analysis, where a richer feature set supports more interpretable behavioral signature extraction. The selected 500 features are extracted from both training and test sets and persisted in Parquet format for all subsequent experimental conditions. Among the 500 selected features, 255 (51.0%) belong to the Permissions group and 245 (49.0%) belong to the non-permissions group, indicating that both the permission space and the non-permission feature space contribute approximately equally to the informative feature subset selected by the Random Forest criterion.

Table 3. Five-Fold Cross-Validation Results Over Candidate Feature Subset Sizes

K	Macro F1 (mean)	Macro F1 (std)
100	0.9452	0.0032
300	0.9441	0.0041
500	0.9458	0.0031
1000	0.9447	0.0046
2000	0.9443	0.0022

3.2. Classification Performance

Table 4 reports the mean and standard deviation of six evaluation metrics across five random seeds for all fifteen classifier-condition combinations. The reporting covers Precision, Recall, F1-score, AUC-ROC, and G-Mean for each combination, enabling a multi-dimensional comparison that goes beyond single-metric selection. Results are averaged over five independent seeds to account for stochastic variability in classifier training, particularly for Decision Tree and Random Forest, which exhibit non-zero variance due to random node splitting. The best-performing combination under the composite ranking criterion, Random Forest under C2, is marked with an asterisk in the table. Logistic Regression and the three tree-based ensemble methods demonstrate qualitatively different sensitivities to the three imbalance-handling conditions, as discussed in detail following the tabular summary. Several patterns are evident across the fifteen combinations reported in Table 4. Across the three tree-based ensemble methods, namely Random Forest, XGBoost, and LightGBM, all three conditions produce F1-scores above 0.89 and AUC-ROC values above 0.98, confirming that these classifiers are inherently well-suited to this detection problem regardless of imbalance handling strategy. Decision Tree performance is consistently lower than that of ensemble methods across all metrics, with G-Mean values below 0.94 under C1, indicating that the single-tree architecture is insufficiently expressive to capture the decision boundary required by this imbalanced binary problem. Logistic Regression exhibits qualitatively different condition sensitivity: under C1 its recall is 0.6558, reflecting systematic misclassification of Backdoor instances without oversampling, while under C2 and C3 recall rises to 0.853 and 0.901 respectively, confirming that the linear model benefits strongly from both oversampling and cost-sensitive weighting. XGBoost and LightGBM under C1 produce standard deviations of zero across all metrics, indicating deterministic convergence to identical solutions across all ten seeds on this dataset, which is consistent with the deterministic tree-building procedures of these implementations when the feature set is fixed.

Table 4. Classification performance across all classifier-condition combinations (mean $\pm$ std over 10 seeds). Best combination by composite rank marked with *.

Condition	Classifier	Precision	Recall	F1	AUC-ROC	G-Mean
	DT	0.857 ± 0.012	0.853 ± 0.002	0.855 ± 0.007	0.946 ± 0.002	0.917 ± 0.002
	LG	0.902 ± 0.000	0.656 ± 0.000	0.759 ± 0.000	0.965 ± 0.000	0.807 ± 0.000
C1	RF	0.983 ± 0.002	0.844 ± 0.005	0.908 ± 0.003	0.992 ± 0.001	0.918 ± 0.003
	XGB	0.966 ± 0.000	0.834 ± 0.000	0.896 ± 0.000	0.989 ± 0.000	0.912 ± 0.000
	LGBM	0.974 ± 0.000	0.844 ± 0.000	0.904 ± 0.000	0.990 ± 0.000	0.918 ± 0.000
	DT	0.813 ± 0.019	0.901 ± 0.007	0.855 ± 0.009	0.947 ± 0.004	0.939 ± 0.003
	LG	0.704 ± 0.023	0.853 ± 0.019	0.771 ± 0.007	0.969 ± 0.001	0.907 ± 0.008
C2	RF*	0.913 ± 0.005	0.896 ± 0.005	0.904 ± 0.001	0.991 ± 0.001	0.942 ± 0.003
	XGB	0.892 ± 0.005	0.908 ± 0.008	0.900 ± 0.004	0.990 ± 0.001	0.948 ± 0.004
	LGBM	0.890 ± 0.005	0.907 ± 0.004	0.899 ± 0.004	0.988 ± 0.000	0.947 ± 0.002
	DT	0.809 ± 0.019	0.883 ± 0.015	0.845 ± 0.014	0.936 ± 0.008	0.930 ± 0.008
	LG	0.594 ± 0.007	0.901 ± 0.005	0.716 ± 0.006	0.970 ± 0.001	0.919 ± 0.003
C3	RF	0.907 ± 0.001	0.894 ± 0.005	0.900 ± 0.003	0.991 ± 0.001	0.941 ± 0.003
	XGB	0.879 ± 0.006	0.920 ± 0.003	0.899 ± 0.004	0.988 ± 0.002	0.953 ± 0.002
	LGBM	0.877 ± 0.003	0.916 ± 0.003	0.896 ± 0.002	0.989 ± 0.001	0.951 ± 0.002

Figure 3 presents the F1-score comparison across all classifiers and imbalance handling conditions as a grouped bar chart with error bars representing standard deviation across ten seeds. The chart organizes classifiers along the horizontal axis with three grouped bars per classifier corresponding to conditions C1, C2, and C3. Error bars for XGBoost and LightGBM under C1 are effectively invisible because their standard deviations are zero. At the same time, Random Forest and Decision Tree show small but non-zero error bars across all conditions. The F1-score profiles of Random Forest, XGBoost, and LightGBM remain consistently high, clustered between 0.89 and 0.91, regardless of condition. In contrast, Logistic Regression shows a pronounced drop under C1 relative to C2 and C3, making it the classifier most sensitive to imbalance handling. Decision Tree shows the most stable F1 across conditions among all classifiers evaluated, though at a lower absolute level than the ensemble methods.

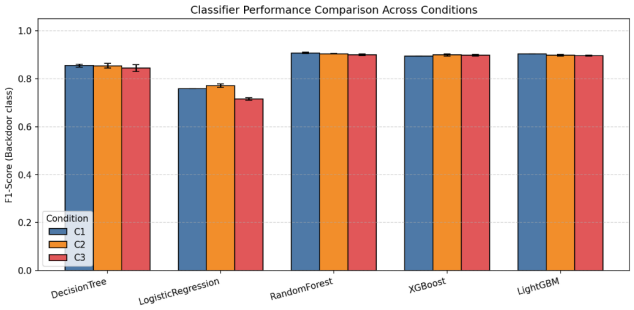

Figure 3. F1-score comparison across classifiers and class imbalance conditions

Figure 4 presents the AUC-ROC comparisons across all classifiers and conditions, using the same grouped bar chart format as in Figure 3. AUC-ROC values for Random Forest, XGBoost, and LightGBM are consistently above 0.988 across all three conditions, reflecting strong probability calibration that is largely insensitive to the imbalance handling strategy applied. Decision Tree produces the lowest AUC-ROC values in the experiment, ranging from 0.936 to 0.947, which is consistent with its lower structural complexity and lack of probability smoothing compared to ensemble methods. Logistic Regression achieves AUC-ROC values between 0.965 and 0.970 across all conditions, demonstrating that its probabilistic output remains well-calibrated even under C1 where its F1-score is substantially degraded by poor threshold-level recall. The AUC-ROC profiles are notably more stable across conditions than the F1-score profiles, indicating that the ranking discrimination of the classifiers is less affected by imbalance handling than the threshold-dependent F1 metric. This divergence between AUC-ROC stability and F1 sensitivity to imbalance conditions reinforces the importance of reporting multiple evaluation metrics rather than relying on any single criterion for model selection under class imbalance.

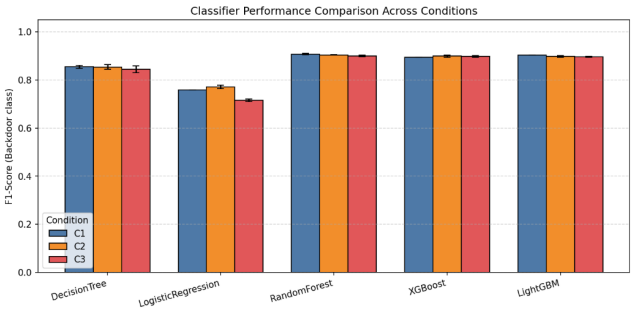

Figure 4. AUC-ROC comparison across classifiers and class imbalance conditions

3.3. Composite Ranking

Table 5 presents the composite ranking scores for the top eight classifier-condition combinations, ordered by ascending composite rank, with lower values indicating superior multidimensional performance. The composite rank is computed as the arithmetic mean of three independent rank positions assigned across all fifteen combinations for F1-score, AUC-ROC, and G-Mean respectively, as defined in Equation 1. Random Forest under C2 achieves the lowest composite rank of 3.33, placing it first overall despite not

achieving the highest individual score on any single metric. The inclusion of G-Mean as a ranking dimension is critical in this context because it explicitly penalizes classifier configurations that trade recall for precision, which would otherwise be rewarded by F1-score maximization alone. Reporting the top eight combinations rather than the full fifteen provides a focused view of the competitive range while retaining sufficient context to observe how classifier type and imbalance condition interact in the ranking. The composite criterion resolves the tie between RF-C2 and RF-C1 in favour of RF-C2 because RF-C1's superior F1 of 0.9081 and AUC-ROC of 0.9921 are offset by a substantially lower G-Mean of 0.9178 compared to 0.9422 under RF-C2, reflecting an unfavourable precision-recall asymmetry that emerges when no imbalance handling is applied. This distinction is operationally significant because G-Mean captures the simultaneous balance between Backdoor detection rate and false positive rate, and a system that sacrifices recall in favour of precision will systematically miss Backdoor instances, which carry severe security consequences. XGBoost with C3 achieves the highest G-Mean across the entire experiment at 0.9528, driven by the combined effect of SMOTE oversampling and scale_pos_weight cost-sensitive weighting on recall. Still, its lower F1 of 0.8988 relative to the Random Forest models places it fourth in the composite ranking. LightGBM under both C2 and C3 achieves competitive G-Mean values above 0.947 but ranks lower than XGBoost and Random Forest on F1, indicating that its gradient boosting mechanism produces a slightly less balanced precision-recall trade-off under these experimental conditions. The composite ranking framework, therefore, identifies RF-C2 as the configuration that most consistently balances detection completeness, discrimination quality, and the sensitivity-specificity equilibrium across all three evaluation dimensions.

Table 5. Composite Ranking of Top 8 Classifier-Condition Combinations (Lower Rank is Better)

Rank	Cond	Classifier	F1	AUC-ROC	G-Mean	Composite
1	**C2**	**RF**	**0.9043**	**0.9909**	**0.9422**	**3.33**
2	C1	RF	0.9081	0.9921	0.9178	4.00
3	C3	RF	0.9002	0.9911	0.9409	4.00
4	C2	XGB	0.8999	0.9896	0.9475	4.33
5	C3	XGB	0.8988	0.9884	0.9528	5.00
6	C3	LGBM	0.8961	0.9892	0.9507	5.67
7	C1	LGBM	0.9043	0.9897	0.9177	5.67
8	C2	LGBM	0.8987	0.9883	0.9471	6.67

To assess whether the performance differences observed across imbalance handling conditions reflect statistically reliable effects, pairwise Wilcoxon signed-rank tests are conducted for each classifier across two condition pairs, C1 versus C2 and C2 versus C3, on three primary metrics: F1-score, G-Mean, and MCC. The Wilcoxon signed-rank test is used as a nonparametric paired test because the same 10 random seeds are used across all conditions, establishing a natural pairing structure without distributional assumptions about the metric scores. Effect size is quantified as $r = |Z|/sqrt(N)$ and interpreted according to the thresholds of Cohen (1992): negligible ($r < 0.10$), small (0.10 to 0.30), medium (0.30 to 0.50), and large ($r > 0.50$). Table 6 reports the p-value and effect size r for each classifier-metric-comparison combination, with significance indicated at alpha = 0.05. Across the 30 comparisons reported in Table 6, 22 yield statistically significant results at alpha = 0.05, with the majority carrying large effect sizes. G-Mean produces the most consistent pattern: 14 out of 15 total pairwise comparisons on this metric across all three condition pairs are significant, indicating that the effect of imbalance handling on sensitivity-specificity balance is robust across all classifiers. The single non-significant G-Mean result is the RF comparison between C2 and C3 ($p = 0.193$, $r = 0.411$), which will be discussed below. This pattern confirms that the G-Mean improvements reported in Table 4 are not attributable to stochastic seed variation but reflect a systematic effect of oversampling on minority-class recall recovery.

For the C1 versus C2 comparison, the G-Mean is significant for all five classifiers ($p = 0.002$, $r = 0.979$), indicating that SMOTE oversampling produces a large and statistically significant improvement in balanced detection rate across classifier types. F1-score under the same comparison yields a more differentiated pattern: LR, XGB, and LGBM produce significant results, while DT and RF do not. The non-significant F1 result for RF ($p = 0.065$, $r = 0.585$) despite a significant G-Mean result reflects the well-known partial cancellation that occurs when SMOTE shifts the decision boundary toward higher recall at the cost of a small precision reduction. G-Mean fully captures the improvement in recall, whereas opposing movements in precision and recall partially offset each other in the F1 computation, attenuating the net change below the significance threshold. This dissociation directly motivates the composite ranking criterion adopted in Section 4.3: a selection procedure based solely on F1 would fail to capture the statistically significant improvement in balanced detection performance that SMOTE confers on Random Forest.

The MCC comparison for RF under C1 versus C2 is significant and negative ($p = 0.002$, $r = 0.979$), indicating that C2 produces a small but statistically reliable reduction in MCC relative to C1, consistent with the precision decrease observed in Table 4. This result reinforces the interpretation that SMOTE introduces a precision-recall trade-off for RF, and the direction of that trade-off is

operationally acceptable given that G-Mean improvement is the target under extreme class imbalance.

For the C2 versus C3 comparison, the RF classifier shows a notably different pattern from the other classifiers. F1 and MCC are both significant with large effect sizes, with C2 superior on both metrics, while G-Mean is not significant (p = 0.193). This pattern indicates that adding cost-sensitive weighting to SMOTE degrades F1 and MCC without providing a statistically significant benefit to sensitivity-specificity balance for RF, confirming that C3 is not the appropriate condition for this classifier on this dataset. For XGB, the C2 versus C3 comparison yields non-significant results on both F1 and MCC. At the same time, G-Mean remains significant (p = 0.002, r = 0.979) and positive, indicating that cost-sensitive weighting provides a meaningful and statistically supported gain in balanced detection rate for gradient boosting without degrading threshold-level performance. This classifier-dependent response to C3 is consistent with the structural difference between bagging and boosting frameworks in their sensitivity to sample weight adjustments.

Table 6. Wilcoxon signed-rank test results for pairwise condition comparisons across classifiers and metrics. p-values with asterisk (*) are significant at alpha = 0.05. Effect size r is reported in parentheses

Classifier	C1 vs C2 F1	C1 vs C2 G-Mean	C1 vs C2 MCC	C2 vs C3 F1	C2 vs C3 G-Mean	C2 vs C3 MCC
DT	0.7695 ns (0.093)	**0.0020*** (0.979)	0.7695 ns (0.093)	**0.0273* (0.698)**	**0.0020* (0.979)**	**0.0273* (0.698)**
LR	**0.0020* (0.979)**	**0.0020*** (0.979)	0.9023 ns (0.039)	**0.0020* (0.979)**	**0.0020* (0.979)**	**0.0020* (0.979)**
RF	0.0645 ns (0.585)	**0.0020*** (0.979)	**0.0020* (0.979)**	**0.0020* (0.979)**	0.1934 ns (0.411)	**0.0020* (0.979)**
XGB	**0.0039* (0.912)**	**0.0020*** (0.979)	0.9004 ns (0.040)	0.7695 ns (0.093)	**0.0020* (0.979)**	0.8457 ns (0.061)
LGBM	**0.0039* (0.912)**	**0.0020*** (0.979)	**0.0020* (0.979)**	**0.0195* (0.739)**	**0.0098* (0.817)**	**0.0195* (0.739)**

3.4. Runtime Analysis

Table 7 reports the mean and standard deviation of training time and per-sample inference time for all fifteen classifier-condition combinations, measured across ten independent runs. Training time is recorded from the start of the fit call to its completion, excluding data loading and preprocessing. Inference time is measured as the total duration of a single predict call on the held-out test set divided by the number of test samples, expressed in milliseconds per sample. All measurements are collected on the same hardware configuration to ensure comparability across combinations. Training time exhibits a marked separation between Logistic Regression and the four tree-based classifiers. Logistic Regression requires between 253 and 315 seconds across conditions, while all tree-based classifiers complete training in under 2.3 seconds. The ratio between the slowest Logistic Regression run and the fastest tree-based run reaches approximately 770-fold, which reflects the fundamental difference in convergence behavior: iterative optimization of a linear model over 500 features under a highly imbalanced sample distribution converges slowly, whereas tree construction terminates deterministically once a fixed number of estimators is grown. The increase in Logistic Regression training time from C1 to C3 is consistent with the larger effective training set introduced by SMOTE and the additional regularization effect of cost-sensitive weighting on the gradient update path.

Among tree-based classifiers, SMOTE oversampling increases training time relative to C1 for all four classifiers, though the magnitude varies by architecture. XGBoost and LightGBM exhibit the largest proportional increases, reaching 2.62x and 2.74x, respectively, reflecting their sequential boosting procedure in which each additional training sample requires propagation through all previously constructed trees. Random Forest training time increases by 1.31x under C2, which is consistent with its parallel tree-growing procedure that distributes the overhead of additional samples more uniformly across estimators. Decision Tree training time shows no consistent increase under SMOTE, remaining within measurement variance across conditions.

Per-sample inference time is uniformly low across all classifiers and conditions, remaining below 0.015 milliseconds in all cases. Decision Tree achieves the lowest inference time across all conditions, reaching 0.000498 milliseconds per sample under C3, consistent with its O(depth) traversal complexity on a single tree. Random Forest produces the highest inference time among all classifiers at 0.014324 milliseconds per sample under C2, which reflects the aggregation cost of polling 100 independently grown trees rather than a single structure. Although it is the slowest in per-sample inference, RF-C2 completes inference on the full test set in 0.048 seconds, achieving a throughput of approximately 70,000 samples per second, sufficient for real-time deployment in static application analysis pipelines.

The training time of RF-C2 at 1.643 ± 0.057 seconds, combined with its per-sample inference time of 0.014 milliseconds, establishes that the best-performing configuration is computationally practical for both periodic retraining on updated malware corpora and high-throughput inference in automated application vetting workflows. The low variance in both training and inference measurements across ten seeds confirms.

Table 7. Training time and per-sample inference time for all classifier-condition combinations (mean ± std over 10 seeds)

Condition	Classifier	Training Time (s)	Inference Time (ms/sample)
C1	DT	1.4675 ± 0.052	0.000526 ± 0.000027
C1	LR	253.264 ± 2.675	0.000648 ± 0.000095
C1	RF	1.2504 ± 0.060	0.014122 ± 0.001121
C1	XGB	0.8562 ± 0.048	0.002820 ± 0.000128
C1	LGBM	0.4092 ± 0.042	0.004820 ± 0.000090
C2	DT	1.3184 ± 0.056	0.000518 ± 0.000015
C2	LR	296.653 ± 28.811	0.000621 ± 0.000045
C2	RF	1.6432 ± 0.057	0.014324 ± 0.001074
C2	XGB	2.2461 ± 0.053	0.002782 ± 0.000030
C2	LGBM	1.1203 ± 0.023	0.004908 ± 0.000164
C3	DT	1.1511 ± 0.048	0.000498 ± 0.000026
C3	LR	315.235 ± 16.422	0.000615 ± 0.000036
C3	RF	1.6386 ± 0.036	0.014204 ± 0.001073
C3	XGB	2.2702 ± 0.044	0.002791 ± 0.000036
C3	LGBM	1.1419 ± 0.028	0.004930 ± 0.000097

3.5. SHAP Analysis

SHAP TreeExplainer is applied to Random Forest retrained under C2 on the full training set to extract feature attributions at the sample level. The analysis operates on the positive-class SHAP slice from the three-dimensional output array of shape (n_samples, n_features, n_classes), yielding a two-dimensional attribution matrix that quantifies each feature's contribution to the Backdoor prediction probability for each test sample. The Backdoor behavioral signature is defined as the set of features whose mean SHAP value across true-positive test samples is strictly positive, isolating features that consistently increase the Backdoor detection probability among correctly classified instances rather than features that are globally informative but directionally ambiguous across all samples. Feature column indices in the signature are mapped to two groups, namely Permissions spanning column indices 0 through 3,267 and Non-Permissions spanning the remaining indices, based on the verified boundary derived from the 1_List_Permissions.csv reference file. This section reports group-level attribution, individual sample distributions, and the ranked permission-level behavioral signature extracted from the analysis.

Figure 5 presents the group-level mean absolute SHAP contribution for the Permissions and Non-Permissions feature groups as a bar chart, enabling a per-feature comparison of discriminative weight between the two groups. The Permissions group yields a mean absolute SHAP of 0.001327 per feature, compared to 0.000575 for the Non-Permissions group, producing a ratio of 2.31 in favour of the Permissions group. This result indicates that declared permissions carry substantially more discriminative weight per feature than non-permission static features for the task of Backdoor discrimination. The disparity is notable given that the two groups contribute approximately equal numbers of features to the selected 500-feature subset, with 255 Permissions features and 245 Non-Permissions features, meaning the difference in per-feature SHAP reflects an inherent difference in discriminative density rather than a sampling artifact. This finding is consistent with the known behavioral profile of Backdoor malware, which relies on declared permission combinations to establish covert access channels, making the permission manifest a structurally informative signal for static detection.

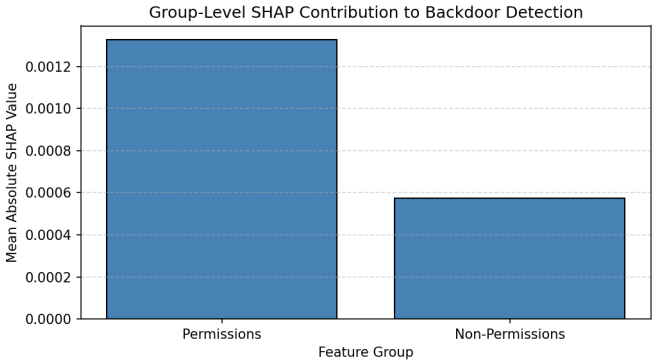

Figure 5. Group-level mean absolute SHAP contribution to backdoor detection

Figure 6 presents the SHAP beeswarm plot for the top 20 features by mean absolute SHAP value, showing the distribution of individual sample-level SHAP values and the direction of each feature's influence on the Backdoor prediction across all test samples. In the beeswarm plot, red points indicate samples with high feature values, and blue points indicate samples with low feature values; the horizontal position represents the magnitude and direction of the SHAP contribution. For Non-Permission features such as feat_3562_Non- and feat_3537_Non-, high feature values are consistently associated with large positive SHAP values extending beyond 0.15, confirming these as strong Backdoor-promoting features whose presence reliably increases the Backdoor detection probability. For Permission features in the top 20, the distribution is more complex, with SHAP contributions spread across both positive and negative directions, reflecting the contextual nature of permission combinations in which the same permission may promote or suppress the Backdoor prediction depending on co-occurring features in the application manifest. The beeswarm visualization also reveals that the top two Non-Permission features produce the widest SHAP spread among the entire top 20, indicating that a small subset of non-permission features provides the strongest individual-sample-level signal, despite the Permissions group having higher mean per-feature attribution overall.

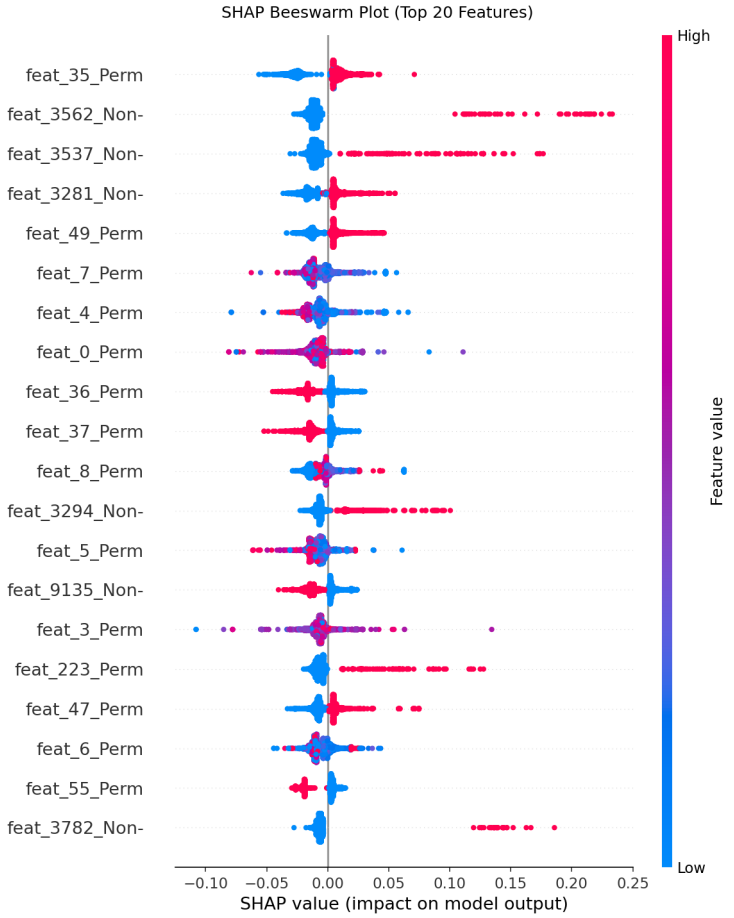

Figure 6. SHAP beeswarm plot for the top 20 features by mean absolute attribution

Table 8 lists the top 15 Android permissions in the Backdoor behavioral signature, ranked by the mean SHAP value computed across true-positive test samples, providing a named and quantified account of the permission-level evidence supporting each Backdoor detection. All 15 permissions yield strictly positive mean SHAP values across true positive samples, confirming their role as consistently Backdoor-promoting features in correctly detected instances rather than incidental correlates. INSTALL_SHORTCUT ranks first with a mean SHAP of 0.0204, followed by GET_ACCOUNTS at 0.0158 and RECEIVE at 0.0145, reflecting the three primary functional objectives of Backdoor applications: establishing a persistent device entry point, harvesting user credentials, and receiving covert commands. Permissions such as WRITE_EXTERNAL_STORAGE, SEND_SMS, and BROADCAST_STICKY in

the mid-range of the table are consistent with data exfiltration and remote instruction broadcasting, while SET_DEBUG_APP at rank 13 indicates that privilege escalation through debugging interface abuse is also a detectable component of the behavioral signature. The complete signature includes 39 named permissions, and the top 15 reported here collectively account for the highest mean SHAP attributions, representing the most reliable and consistently activated permission-level indicators of Backdoor behavior detectable through static manifest analysis.

Table 8. Top 15 Android Permissions in The Backdoor Behavioral Signature

Rank	Permission	Mean SHAP (TP)
1	INSTALL_SHORTCUT	0.0204
2	GET_ACCOUNTS	0.0158
3	RECEIVE	0.0145
4	WRITE_EXTERNAL_STORAGE	0.0143
5	MANAGE_ACCOUNTS	0.0136
6	BLUETOOTH	0.0135
7	SEND	0.0121
8	VIBRATE	0.0116
9	RECEIVE_MCS_MESSAGE	0.0112
10	SERVICE_ACCESS	0.0109
11	WRITE_USE_APP_FEATURE_SURVEY	0.0106
12	DOWNLOAD_WITHOUT_NOTIFICATION	0.0100
13	SET_DEBUG_APP	0.0094
14	SEND_SMS	0.0092
15	BROADCAST_STICKY	0.0089

3.6. Discussions

3.6.1. Effect of Imbalance Handling on Ensemble Classifiers

The composite ranking result reflects a consistent, statistically supported pattern across the full experiment: SMOTE oversampling alone (C2) produces the most favorable aggregate profile for Random Forest, while adding cost-sensitive weighting (C3) degrades performance on this classifier. Under C3, Random Forest precision drops from 0.9133 to 0.9071 while recall improves only marginally from 0.8961 to 0.8945, yielding a net reduction in both F1 and MCC. The Wilcoxon signed-rank tests reported in Section 4.4 confirm that this degradation is statistically significant: the F1 difference between C2 and C3 for Random Forest yields $p = 0.002$ with a large effect size ($r = 0.979$), while the G-Mean difference between the same conditions is not significant ($p = 0.193$), indicating that cost-sensitive weighting shifts the precision-recall balance without providing any statistically reliable gain in sensitivity-specificity equilibrium. The mechanistic explanation is that balanced class weighting introduces an over-correction to the Random Forest splitting criterion, increasing the penalty on false negatives to a degree that displaces the decision boundary beyond its optimal position for this imbalance ratio, raising false positives without proportionally recovering additional true positives. This finding is consistent with the observation of Zhu et al. [6], who reported that ensemble methods applied to imbalanced Android malware datasets exhibit classifier-specific sensitivity to cost-sensitive weighting, with bagging-based ensembles showing diminishing returns when oversampling is already applied.

For gradient boosting classifiers, the effect of the imbalance handling conditions follows a qualitatively different pattern. XGBoost and LightGBM both show statistically significant G-Mean improvements from C2 to C3 ($p = 0.002$, $r = 0.979$ for both), with XGBoost G-Mean increasing from 0.9478 under C2 to 0.9526 under C3 and LightGBM G-Mean increasing from 0.9477 to 0.9507. The corresponding F1 and MCC changes are not statistically significant for XGBoost (F1: $p = 0.770$; MCC: $p = 0.846$), indicating that cost-sensitive weighting through scale_pos_weight shifts the boosting decision boundary toward higher recall without materially degrading threshold-level precision. This behavior is structurally consistent with the sequential nature of gradient boosting, in which each subsequent tree is trained on the residuals of all previous trees, making the model more sensitive to per-sample weight adjustments than a bagging ensemble that constructs trees independently. The overall detection performance of Random Forest under C2 in this study, with F1 = 0.9046, AUC-ROC = 0.9917, and G-Mean = 0.9426, is consistent with the findings of Xie et al. [7], who also demonstrated that Random Forest within a stacking ensemble achieves competitive detection performance on Android malware datasets, and with Pudke and Bansal [12], whose stacked ensemble incorporating Random Forest as a base classifier attained strong class separation on a permission-based feature representation.

These findings support a classifier-dependent practical recommendation for imbalance handling in static Android malware detection. For Random Forest, SMOTE oversampling to a 1:2 training ratio is sufficient, and additional cost-sensitive weighting is

counterproductive under the experimental conditions evaluated here. For gradient boosting approaches, the combination of SMOTE and cost-sensitive weighting provides a statistically significant benefit to G-Mean without degrading F1, and is therefore the preferred configuration when balanced sensitivity-specificity performance is the primary deployment objective. The appropriate imbalance handling strategy is therefore not transferable across classifier architectures and must be determined empirically for each classifier type. The consistent superiority of ensemble methods over the single Decision Tree baseline across all conditions and metrics in this experiment corroborates the broader finding reported by Xie et al. [7] that ensemble architectures provide systematic advantages over single-tree classifiers in high-dimensional Android malware detection tasks, particularly under class-imbalance conditions, where the variance-reduction property of ensembles directly attenuates the minority-class misclassification risk.

3.7. Interpreting the K-Selection Plateau

The flat cross-validation curve across K $\epsilon\{100,300,500,1000,2000\}$ has a practical interpretation beyond hyperparameter selection: it suggests that the Backdoor class is separable from the remaining classes using a compact core of static features, and that features ranked below position 100 contribute predominantly to noise or redundancy. This finding has direct deployment implications because a detection system based on the top 100 features achieves nearly identical cross-validation performance to one using 2,000 features, enabling significantly reduced inference time and memory footprint in resource-constrained deployment environments such as mobile security gatekeepers or application store analysis pipelines.

3.8. Semantic Interpretation of the Behavioral Signature

The 39 named permissions in the Backdoor behavioral signature cluster into four functionally coherent groups. Account access permissions, including GET_ACCOUNTS, MANAGE_ACCOUNTS, USE_CREDENTIALS, and INSTALL_SHORTCUT, reflect the Backdoor objective of harvesting user identity information and establishing persistent device-level access points. Covert communication permissions, including RECEIVE, C2D_MESSAGE, RECEIVE_MCS_MESSAGE, and BROADCAST_STICKY, are consistent with command-and-control channel establishment, in which the application registers listeners for incoming operator instructions. Data exfiltration permissions, including WRITE_EXTERNAL_STORAGE, READ_WRITE_BOOKMARK_FOLDERS, and SEND_SMS, enable transfer of harvested data to external destinations. Privilege escalation permissions, including REQUEST_INSTALL_PACKAGES, CHANGE_COMPONENT_ENABLED_STATE, and SET_DEBUG_APP, allow the Backdoor to install or remove components and disable security monitoring without user interaction.

This semantic coherence confirms that the behavioral signature is not an artifact of the specific classifier or random seed, but rather reflects the structural properties of the Backdoor application category as declared in AndroidManifest.xml. The presence of RECIEVE_MCS_MESSAGE and SERVICE_ACCESS in the signature suggests a strong association with Baidu Mobile Services SDK patterns, consistent with the predominance of Chinese-market-targeted Backdoor families such as GingerMaster and DroidKungFu variants in the dataset, providing geographically specific threat intelligence that informs regional application store security policies.

3.9. Practical Contributions Beyond Accuracy

The practical value of this study extends beyond classification metrics in two directions. First, the composite ranking methodology provides a general-purpose best model selection framework applicable to any highly imbalanced binary detection problem in which no single metric captures all operationally relevant trade-offs. By requiring simultaneous rank optimization across F1, AUC-ROC, and G-Mean, the criterion prevents the selection of models that appear competitive on one dimension while underperforming on another. The statistical validation in Section 4.4 reinforces this point directly: Random Forest under C2 achieves a statistically significant G-Mean improvement over C1 ($p = 0.002$, $r = 0.979$) that would be invisible to a selection procedure based solely on F1-score, which yields a non-significant difference ($p = 0.065$) between the same two conditions. The composite criterion is therefore not merely a convenience but a methodologically necessary instrument when the detection objective involves simultaneously controlling false negatives and false positives under extreme imbalance.

Second, the 39-permission behavioral signature provides directly actionable inputs for static analysis rule engines that operate independently of any trained classifier. Each permission in the signature can be encoded as a detection indicator based solely on manifest content, enabling deployment in lightweight policy enforcement tools that process APK files before any ML inference is required. The four functional clusters identified in Section 5.3, covering account access, covert communication, data exfiltration, and privilege escalation, map to distinct stages of the Backdoor operational lifecycle and can each be treated as an independent detection signal. Combinations drawn from multiple clusters carry particularly high specificity given the rarity of such permission co-

occurrences in benign applications, and the per-permission SHAP values provide a quantitative basis for assigning confidence weights to these combinations in a scoring framework. Patel and Ghosh [17] demonstrated that explainability-derived feature attributions can be operationalized directly into adaptive detection rules for Android malware, and the permission-level SHAP values reported here provide equivalent inputs for the Backdoor category specifically.

The security relevance of these contributions is most apparent when considered in the context of automated application vetting workflows. Application distribution platforms such as Google Play perform static analysis on submitted APKs before publication, and the permission signature identified in this study provides a concrete set of manifest-level indicators that can be integrated into such pre-publication screening pipelines as a Backdoor-specific detection layer. Because the signature is derived from true positive detections rather than global feature importance, it reflects the permission profile of correctly identified Backdoor samples and is unlikely to yield high false-positive rates against standard utility applications that legitimately request subsets of these permissions for non-malicious purposes. For mobile security vendors and enterprise mobile device management systems that process large volumes of APK submissions, the inference time of 0.014 milliseconds per sample reported for RF-C2 in Section 4.5 establishes that the full ML detection pipeline can operate at a throughput of approximately 70,000 samples per second, which is compatible with real-time vetting at the scale of major application markets. Additionally, the geographically specific threat intelligence embedded in the signature, particularly the presence of RECIEVE_MCS_MESSAGE and SERVICE_ACCESS pointing to Baidu Mobile Services SDK patterns, provides regionally actionable information for security analysts monitoring Chinese-market-targeted Backdoor families, enabling prioritized triage of applications exhibiting these specific permission combinations within regional application stores and enterprise device fleets.

3.10. Limitations

Four limitations bound the scope of the reported results. First, the negative class subsampling to 15,380 instances represents 4.5% of the full negative class population. The excluded 325,251 negative instances may contain applications with static feature profiles more similar to Backdoor than those retained in the working dataset. Generalization of the reported F1, AUC-ROC, and G-Mean values to the full-dataset distribution is therefore not guaranteed, and experiments on the complete negative class without subsampling constitute an important direction for future work.

Second, the Non-Permissions feature group spans 6,234 columns representing services, intent actions, and intent categories, but the dataset documentation does not provide individual feature names for these columns. SHAP attribution within this group cannot be disaggregated into named features, limiting the interpretability of the 49% of selected features that it contains. The study addresses this limitation by reporting group-level attribution, but a complete named signature covering all 39 signature entries requires access to the full AndroidManifest.xml declaration list files for services and actions.

Third, the dataset was collected in 2020 and reflects the Backdoor malware landscape at that time. Backdoor applications employing post-2020 evasion techniques, including manifest obfuscation, permission splitting across dynamic loading stages, and declarative permission suppression, may not exhibit the static signature identified here. Temporal generalization testing on more recent Backdoor samples is required before operational deployment of the signature as a primary detection mechanism.

Fourth, the experiment does not include deep learning architectures as comparative baselines. This exclusion is deliberate, given the structural properties of the detection task. The input representation in this study consists of a sparse binary feature vector of dimensionality 500, derived from AndroidManifest.xml declarations that encode declared capabilities rather than sequential or spatial patterns. Convolutional and recurrent neural architectures are principally designed to exploit local correlation structure in sequential or grid-organized inputs, and the theoretical basis for their advantage over tree-based ensembles on tabular binary feature vectors of this dimensionality is not established. Najibi and Bidgoly [19] noted that deep learning models applied to static Android malware features require careful regularization to avoid overfitting on sparse high-dimensional manifests, and that interpretability in such settings is substantially more constrained than in tree-based methods. For the specific objective of this study, which includes extracting a named permission-level behavioral signature through SHAP TreeExplainer, tree-based models provide native compatibility with exact Shapley value computation, whereas deep learning attribution methods such as SHAP DeepExplainer or gradient-based saliency maps introduce approximation error that can affect the reliability of individual feature attributions on sparse binary inputs. Alotaibi [20] demonstrated that deep learning architectures can achieve strong classification performance on malware family datasets when image-based or multimodal feature representations are available, but noted that such gains are tied to the richness of the input modality rather than the classifier architecture alone. The present study, therefore, establishes a strong interpretable tree-based baseline on the static binary representation, and the extension to deep learning formulations constitutes a distinct future research direction that requires a different feature representation strategy, such as embedding-based encoding of the manifest structure, rather than a straightforward substitution of the classifier component.

4. CONCLUSION

This study presents a Backdoor-specific binary classification pipeline on the CCCS-CIC-AndMal-2020 dataset using the full 9,502-dimensional static feature space, with systematic comparison of class imbalance handling strategies across five classifiers. Random Forest under SMOTE oversampling achieves the best composite rank, recording an F1-score of 0.9043, AUC-ROC of 0.9909, G-Mean of 0.9422, and MCC of 0.8948. The composite ranking criterion proves essential in this selection: Random Forest under no-handling attains a marginally higher F1 yet ranks fourth due to a substantially lower G-Mean, demonstrating that single-metric optimisation is insufficient for imbalanced detection tasks where missed Backdoor samples carry severe security consequences.

SHAP analysis identifies the Permissions group as the dominant discriminator, contributing 2.31 times more mean absolute SHAP value per feature than the Non-Permissions group. From this group, 39 named permissions form a semantically coherent behavioral signature spanning account access, covert communication, data exfiltration, and privilege escalation. This finding confirms that interpretable ensemble learning provides not only high detection performance but also actionable security intelligence, as the extracted signature can be encoded directly as detection rules in static analysis tools without requiring model inference.

Future work should evaluate the pipeline on the complete negative class without subsampling, assess temporal generalization on post-2020 Backdoor samples to address concept drift, investigate the Non-Permission features through access to full AndroidManifest.xml declaration lists, and examine whether transformer-based embedding of static feature sequences improves upon the tree-based baseline established here.

5. ACKNOWLEDGEMENTS

The authors gratefully acknowledge financial support from Program Penelitian Internal Universitas Dian Nuswantoro – Penelitian Dasar Perguruan Tinggi, Semester Gasal Tahun 2025–2026, under grant number 032/F.9/UDN-09/II/2026.

6. DECLARATIONS

AI USAGE STATEMENT

During the preparation of this work, the authors used ChatGPT (OpenAI) and Quilbot ONLY to improve the language and clarity of the manuscript. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

AUTHOR CONTIBUTION

Conceptualization, Rama Aria Megantara and Farrikh Alzami; methodology, Farrikh Alzami; software, Ricardus Anggi Pramunendar; validation, Muhammad Naufal, Dwi Puji Prabowo, and Rivaldo Mersis Brilianto; formal analysis and investigation, Rama Aria Megantara; resources and data curation, Dewi Pergiwati; writing—original draft preparation, Rama Aria Megantara and Dewi Pergiwati; writing—review and editing, Farrikh Alzami; visualization, Dwi Puji Prabowo; supervision, project administration, and funding acquisition, Dewi Pergiwati.

FUNDING STATEMENT

The authors gratefully acknowledge financial support from Program Penelitian Internal Universitas Dian Nuswantoro – Penelitian Dasar Perguruan Tinggi, Semester Gasal Tahun 2025–2026, under grant number 032/F.9/UDN-09/II/2026.

COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

[1] D. Zhang, X. Yang, S. Liu, Y. Zhou, J. Fu, and G. Peng, "A survey on Android dynamic evasive malware: Taxonomy, countermeasures and open challenges," *Computers & Security*, vol. 159, p. 104646, Dec. 2025, https://doi.org/10.1016/j.cose.2025.104646.

[2] D. S. Keyes, B. Li, G. Kaur, A. H. Lashkari, F. Gagnon, and F. Massicotte, "EntropLyzer: Android Malware Classification and Characterization Using Entropy Analysis of Dynamic Characteristics," in *2021 Reconciling Data Analytics, Automation,*

Privacy, and Security: A Big Data Challenge (RDAAPS). Hamilton, ON, Canada: IEEE, May 2021, pp. 1–12, https://doi.org/10.1109/RDAAPS48126.2021.9452002.

[3] A. Ghourabi, "An Attention-Based Approach to Enhance the Detection and Classification of Android Malware," *Computers, Materials & Continua*, vol. 80, no. 2, pp. 2743–2760, 2024, https://doi.org/10.32604/cmc.2024.053163.

[4] Y. Qu, H. Ma, C. Zheng, Y. Jiang, and W. Wang, "A malware traffic detection method based on Victim-Attacker interaction patterns," *Computers & Security*, vol. 155, p. 104487, Aug. 2025, https://doi.org/10.1016/j.cose.2025.104487.

[5] E. Chatzoglou and G. Kambourakis, "C3: Leveraging the Native Messaging Application Programming Interface for Covert Command and Control," *Future Internet*, vol. 17, no. 4, p. 172, Apr. 2025, https://doi.org/10.3390/fi17040172.

[6] H.-j. Zhu, Y. Li, L.-m. Wang, and V. S. Sheng, "A multi-model ensemble learning framework for imbalanced android malware detection," *Expert Systems with Applications*, vol. 234, p. 120952, Dec. 2023, https://doi.org/10.1016/j.eswa.2023.120952.

[7] J. Xie, S. Li, X. Yun, C. Si, and T. Yin, "Sample analysis and multi-label classification for malicious sample datasets," *Computer Networks*, vol. 258, p. 110999, Feb. 2025, https://doi.org/10.1016/j.comnet.2024.110999.

[8] M. M. Khan, A. Buriro, T. Ahmad, and S. Ullah, "Backdoor Malware Detection in Industrial IoT Using Machine Learning," *Computers, Materials & Continua*, vol. 81, no. 3, pp. 4691–4705, 2024, https://doi.org/10.32604/cmc.2024.057648.

[9] A. Alsraratee and A. Al-Azawei, "Classifying Android Malware Categories Based on Dynamic Features: An Integration of Feature Reduction and Selection Techniques," *Kufa Journal of Engineering*, vol. 16, no. 2, pp. 96–118, Apr. 2025, https://doi.org/10.30572/2018/KJE/160206.

[10] S. Zhou, H. Li, X. Fu, D. Han, and X. He, "Novel Multi-Classification Dynamic Detection Model for Android Malware Based on Improved Zebra Optimization Algorithm and LightGBM," *Sensors*, vol. 24, no. 18, p. 5975, Sep. 2024, https://doi.org/10.3390/s24185975.

[11] S. I. Imtiaz, S. U. Rehman, A. R. Javed, Z. Jalil, X. Liu, and W. S. Alnumay, "DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network," *Future Generation Computer Systems*, vol. 115, pp. 844–856, Feb. 2021, https://doi.org/10.1016/j.future.2020.10.008.

[12] P. Pudke and U. Bansal, "A Stacked Ensemble Framework for Android Malware Detection Using Semantic Permission Aggregation and Explainable AI," *Security and Privacy*, vol. 8, no. 6, p. e70110, Nov. 2025, https://doi.org/10.1002/spy2.70110.

[13] N. H. Saeed, A. A. Hamza, M. A. Sobh, and A. M. Bahaa-Eldin, "Efficient feature ranked hybrid framework for android Iot malware detection," *Scientific Reports*, vol. 16, no. 1, p. 3726, Jan. 2026, https://doi.org/10.1038/s41598-026-35238-6.

[14] J. Li, J. He, W. Li, W. Fang, G. Yang, and T. Li, "SynDroid: An adaptive enhanced Android malware classification method based on CTGAN-SVM," *Computers & Security*, vol. 137, p. 103604, Feb. 2024, https://doi.org/10.1016/j.cose.2023.103604.

[15] X. Cheng, T. Wang, D. Zhu, and J. Ma, "Uncertainty explanation of artificial intelligence models by SHAP," *Knowledge-Based Systems*, vol. 337, p. 115437, Mar. 2026, https://doi.org/10.1016/j.knosys.2026.115437.

[16] D. Soi, A. Sanna, D. Maiorca, and G. Giacinto, "Enhancing android malware detection explainability through function call graph APIs," *Journal of Information Security and Applications*, vol. 80, p. 103691, Feb. 2024, https://doi.org/10.1016/j.jisa.2023.103691.

[17] A. Patel and S. M. Ghosh, "EML-AMD: An explainable machine learning framework for adaptive android malware detection," *Peer-to-Peer Networking and Applications*, vol. 18, no. 5, p. 264, Sep. 2025, https://doi.org/10.1007/s12083-025-02069-7.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002, https://doi.org/10.1613/jair.953.

[19] M. Najibi and A. J. Bidgoly, "Towards a robust android malware detection model using explainable deep learning," *Journal of Information Security and Applications*, vol. 93, p. 104191, Sep. 2025, https://doi.org/10.1016/j.jisa.2025.104191.

[20] B. Alotaibi, "Multimodal Deep Learning Fusion for Accurate and Explainable Malware Family Classification," *Applied Sciences*, vol. 15, no. 21, p. 11635, Oct. 2025, https://doi.org/10.3390/app152111635.

[This page intentionally left blank.]