BAREKENG: Journal of Mathematics and Its Applications
June 2026 Volume 20 Issue 2 Page 1675–1692
P-ISSN: 1978-7227 E-ISSN: 2615-3017
https://doi.org/10.30598/barekengvol20iss2pp1675-1692

A COMPARATIVE STUDY OF PIPELINE-VALIDATED MACHINE LEARNING CLASSIFIERS FOR PERMISSION-BASED ANDROID MALWARE DETECTION

Arif Ridho Lubis[1*], Dewi Wulandari[2], Lilis Tiara Adha[3], Tika Ariyani[4], Yuyun Yusnida Lase[5], Fahdi Saidi Lubis[6]

[1,2,3,4,5]*Departement Computer Engineering and Informatics, Politeknik Negeri Medan*
Jln. Almamater no 1 Kampus USU, Medan, 20155, Indonesia

[6]*Departement of Information Technology, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia*
Jln. Gombak, 53100, Malaysia

*Corresponding author's e-mail: * **arifridho@polmed.ac.id***

Article Info

Article History:

Received: 23rd July 2025
Revised: 12th September 2025
Accepted: 3rd November 2025
Available Online:26th January 2026

Keywords:

Android Malware;
Classification;
Gradient Boosting Machine;
Logistic Regression;
Permission-Based Detection;
Random Forest.

ABSTRACT

The growing prevalence of Android malware distributed through third-party APK sideloading poses a significant security threat to users and developers. This study aims to evaluate the effectiveness of three machine learning algorithms—Logistic Regression (LR), Random Forests (RF), and Gradient Boosting Machine (GBM)—for static Android malware detection based on permission features. The experiment employs the publicly available Android Malware Prediction Dataset (Kaggle, accessed 2025), containing 4,464 application samples with 328 binary permission attributes. A leakage-free CRISP-DM workflow was implemented, integrating data cleaning, automated feature selection via SelectKBest (Mutual Information), and hyperparameter optimisation using GridSearchCV with stratified 5-fold cross-validation. Results on the unseen hold-out test set show that GBM achieved the best performance, with 96.05% accuracy and 0.9924 ROC-AUC, outperforming LR and RF. In addition, GBM exhibited superior probability calibration (Brier Score = 0.0344) and interpretability, as confirmed through SHAP analysis. The ablation study further validated that optimal model performance saturates at 30–40 selected features. This research contributes a reproducible and pipeline-validated comparative framework for static Android malware detection, addressing prior studies' limitations regarding feature selection bias and data leakage. Nevertheless, the study is limited by its reliance on static permission features and the absence of dynamic behavioural data, which may restrict generalisation to evolving malware families.

How to cite this article:

A. R. Lubis, D. Wulandari, L. T. Adha, T. Ariyani, Y. Y. Lase and F. S. Lubis., "A COMPARATIVE STUDY OF PIPELINE-VALIDATED MACHINE LEARNING CLASSIFIERS FOR PERMISSION-BASED ANDROID MALWARE DETECTION, *BAREKENG: J. Math. & App.*, vol. 20, no. 2, pp. 1675-1692, Jun, 2026.

Journal homepage: *https://ojs3.unpatti.ac.id/index.php/barekeng/*
Journal e-mail: *barekeng.math@yahoo.com; barekeng.journal@mail.unpatti.ac.id*

Research Article · Open Access

1. INTRODUCTION

In today's hyperconnected digital environment, the boundaries between communication, work, and entertainment have become increasingly blurred. The Android operating system dominates the mobile market, powering billions of devices worldwide and serving as the backbone of mobile applications and online services [1][2]. Its open-source framework provides flexibility and accessibility for developers, yet it also introduces major cybersecurity vulnerabilities [3]. Cybercriminals are increasingly using social engineering tactics to trick users into downloading malicious Android application packages (APKs) that appear legitimate, often disguised as entertainment, productivity tools, or system updates [4][5]. These applications are typically distributed outside the Google Play Store, a process known as sideloading, which bypasses built-in security verification and exposes users to elevated risk [6]. Once installed, such malware can access sensitive resources through overprivileged permissions, enabling activities such as surveillance, credential theft, and financial fraud [7][8]. Research comparing sideloaded apps to those from the Play Store found that sideloaded applications are more likely to lack privacy policies, transmit sensitive data unencrypted, and even contain stalkerware or other malicious functionalities, making them especially dangerous for users [9]. Dynamic code loading, a technique often used in sideloaded apps, can stealthily introduce malware or vulnerabilities that evade detection by traditional antivirus tools, further increasing the risk of privacy breaches and unauthorised access to device resources [10].

To mitigate these growing threats, machine learning (ML) has emerged as a central approach in Android malware detection. ML-based systems can learn complex behavioural patterns from static features such as permissions, API calls, or code metadata, providing more scalable detection than traditional signature-based methods [11]. Among these, permission-based static analysis is particularly attractive because it does not require executing the application and can thus be deployed in resource-constrained environments such as mobile devices or app stores [12][13]. Each Android app declares its required permissions in the manifest file, and patterns in these requests have been shown to strongly correlate with malicious intent [14]. Various ML algorithms, including Random Forests, XGBoost, Support Vector Machines, and ensemble methods, have demonstrated strong performance in malware detection, with some models achieving accuracy above 90% and low false-positive rates [15]. However, challenges remain, such as handling imbalanced datasets, inconsistent evaluation protocols, limited cross-validation, or non-calibrated probability outputs, which undermine reproducibility and limit real-world reliability [16].

The purpose of this study is to analyse and classify malware disseminated through Android applications using a publicly available Kaggle dataset containing thousands of pre-labelled benign and malicious samples, each represented by binary permission features [17]. The dataset was cleaned, deduplicated, and standardised to ensure integrity and reproducibility. The research employs a structured data-mining framework consistent with CRISP-DM (CRoss-Industry Standard Process for Data Mining) [18] principles, integrating mutual information–based feature selection, cross-validated hyperparameter tuning, and probability calibration via Brier scores and reliability curves Three supervised learning algorithms were also selected based on their complementary characteristics, best results of the previous studies, and relevance to binary classification tasks in malware detection. Logistic Regression was chosen for its interpretability and computational efficiency, enabling a clear understanding of how individual permissions contribute to the prediction of malicious behaviour [19]. Random Forests were included as a robust ensemble method that mitigates overfitting and effectively handles large sets of binary permission features [20]. Meanwhile, the Gradient Boosting Machine (GBM) was employed to capture complex non-linear relationships through iterative error correction, often achieving superior predictive performance on structured datasets [21]. The models will be trained to determine whether an application is malicious by learning the permissions requested by APK files. The combination of these models enables a comparative analysis that balances interpretability, generalisation, and accuracy, offering insights into the trade-offs between lightweight and high-performance malware classifiers.

Several studies have addressed malware detection, yet methodological differences remain. The first research, conducted by Nasri et al. [22], employs five algorithms, namely Random Forests, Naive Bayes, J48, DecisionTable, and MLP, for classifying and identifying malware. Their results showed that Random Forests achieved the highest accuracy of 89.36%, validating ensemble methods as effective models for malware detection. The dataset used by Nasri et al. was sourced from the AndroZoo project, containing APKs collected from various repositories, including Google Play. While their approach demonstrated competitive accuracy, their manual decompilation of applications to extract permissions limited scalability and reproducibility. The present study differs by leveraging automated preprocessing and reproducible open-source datasets,

integrated into a fully encapsulated pipeline that supports robust hyperparameter tuning and model comparison.

The study conducted by Ahmed et al. [23] compared the performance of six machine learning algorithms for classifying Android malware: Decision Trees, Support Vector Machines, Naive Bayes, Random Forests, K-Nearest Neighbours, and Ensemble Methods / Extra-Tree Classifier. The study achieved its best accuracy of 95% using Random Forests with feature selection techniques such as Correlation, Chi-Square, and Information Gain. Although the findings confirmed Random Forest's strength for malware detection, the evaluation pipeline selected features before cross-validation, introducing a feature-leakage problem. Moreover, their analysis lacked emphasis on model calibration or interpretability. The present research improves upon this by embedding feature selection and scaling directly within the cross-validation process and evaluating probability calibration using Brier scores.

The third research by Droos et al. [24] utilised four machine learning algorithms for the detection of Android malware: Random Forests, Naive Bayes (NB), J48, and IBK (K-Nearest Neighbors). Their evaluation employed metrics such as Recall, Precision, and Accuracy, achieving the highest accuracy of 98.6% with Random Forests. The dataset, sourced from the University of New Brunswick repository, contained multiple malware types, including adware, banking trojans, and SMS malware. Although their results were impressive, their study did not explore calibration or discuss model efficiency for deployment in resource-limited devices. In contrast, this research evaluates three models of varying complexity, which are Logistic Regression, Random Forests, and GBM, under identical conditions to identify the optimal balance between interpretability, computational efficiency, and predictive performance.

The fourth study, Kapoor et al. [25], examined six classical algorithms—Logistic Regression, Linear Discriminant Analysis, k-Nearest Neighbours, Decision Tree, Naïve Bayes, and SVM—for permission-based Android malware detection. Logistic Regression achieved the highest accuracy (99.34%), confirming its strong suitability for binary permission-based data. However, their dataset was self-constructed, lacked validation against external sources, and did not control for feature overlap or data leakage. The current research builds upon Kapoor et al. by employing a public, reproducible dataset, applying mutual information-based feature selection, and using calibrated probabilistic outputs, thereby addressing reproducibility and overfitting concerns absent in earlier studies.

The purpose of this comparative analysis is to determine the most effective model for identifying permission-based Android malware by using Logistic Regression, Random Forests, and Gradient Boosting Machines. This research introduces several key innovations: (1) a reproducible, leakage-free pipeline integrating mutual information–based feature selection; (2) cross-validated hyperparameter optimisation; and (3) calibrated probabilistic evaluation. Additionally, an ablation study was conducted to analyse the influence of feature dimensionality and selection criteria (Mutual Information vs. Correlation), confirming the robustness and stability of the proposed framework. By addressing long-standing issues of data leakage, reproducibility, and model interpretability, this research aims to contribute toward the development of trustworthy, resource-efficient, and scientifically transparent malware detection systems.

2. RESEARCH METHODS

2.1 Workflow

This study follows a structured analytical workflow inspired by the Cross-Industry Standard Process for Data Mining (CRISP-DM). CRISP-DM is a domain- and technology-independent process model that structures data mining and machine learning projects into six iterative phases: business understanding, data understanding, data preparation, modelling, evaluation, and deployment [26]. The stages include Dataset Understanding, Exploratory Data Analysis (EDA), Feature Selection Strategy Design, Data Preprocessing, Modelling and Evaluation, Implementation and Study, as shown in Fig. 1.

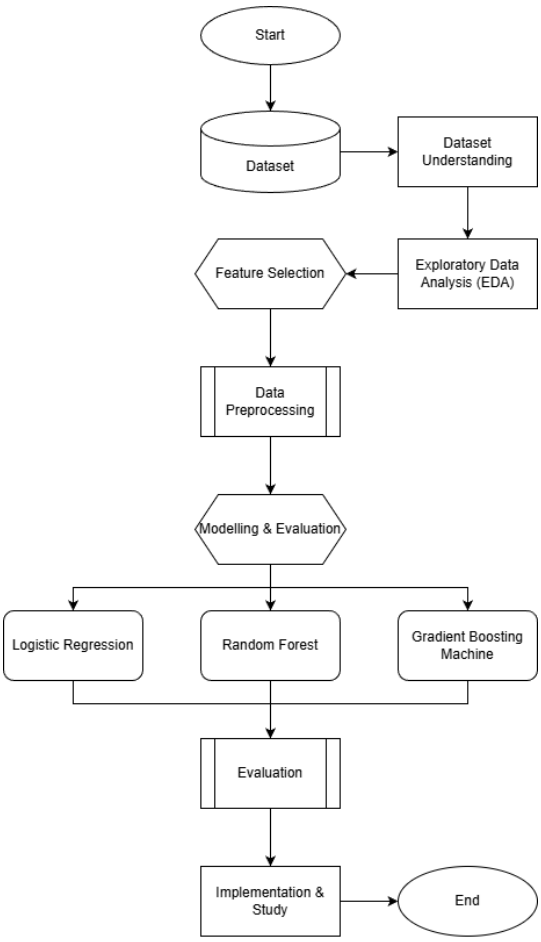

Figure 1. Workflow Diagram

The process begins with dataset acquisition and understanding, where the Android permissions dataset from Kaggle [16] is examined for completeness, consistency, and suitability for classification. This stage lays the foundation for subsequent analysis by identifying the data's structure and characteristics. Following data understanding, exploratory data analysis (EDA) is performed to visualise and summarise the dataset's main features, identify general patterns, highlight key differences between benign and malicious applications, and see how the columns correlate with each other. EDA insights guide the design of the feature selection strategy, which determines the most relevant permission attributes using K-Fold Select. The data is then preprocessed through cleaning, normalisation, and transformation to ensure that it is ready for training and evaluation.

The prepared data proceeds to the modelling and evaluation stage, where three supervised learning algorithms—Logistic Regression, Random Forests, and Gradient Boosting Machine—are implemented and compared. Each model is trained using a standardised pipeline with cross-validation and grid search optimisation. The results are evaluated using multiple performance metrics, and the workflow concludes with implementation and study, where the models are instructed to solve high-risk, medium-risk, low-risk, and no-risk cases, and the models are analysed for calibration, interpretability, and robustness through additional validation and ablation experiments.

2.2 Dataset

The dataset utilised in this research is a permission-based Android malware dataset, the "Android Malware Detection Dataset" by Danny Revaldo, obtained from Kaggle [16], which contains static binary features representing permissions requested by Android application packages (APKs). Each row in the dataset corresponds to an individual application, while each column represents a specific permission or API-level feature extracted from the application's manifest file. The dataset comprises 4,464 records and 328 binary attributes, along with a target column labelled benign or malware that serves as the ground truth for this study. Each feature takes a binary value of 1 if the corresponding permission is declared by the application, and 0 otherwise. After duplicate removal, 3,292 unique samples remain, with 1,745 benign applications and 1,547 malware applications in the cleaned dataset.

The permissions in this dataset span a wide range of Android system resources, covering device identifiers, storage access, communication services, location data, and system-level execution capabilities. These permissions are typically used by legitimate applications but are frequently abused by malicious ones to exfiltrate data, track user activity, or execute unauthorised actions. The dataset also includes several API call indicators that correspond to potentially sensitive operations, such as dynamic code loading or direct network communication, which can reveal malicious payload distribution or command-and-control behaviour. The features in this dataset capture these behavioural patterns, making it particularly suitable for static analysis and machine learning classification. Table 1 lists the permissions represented in the dataset.

Table 1. Permission Examples in the Dataset

No.	Column Name	Description
1.	android.permission.READ_PHONE_STATE	Allows access to device identifiers, IMEI, network information, and call status.
2.	android.permission.ACCESS_NETWORK_STATE	Allows an application to access the device's location services to determine the user's precise location.
3.	android.permission.BLUETOOTH	Enables an application to use the device's Bluetooth capabilities, including connecting to other devices.
4.	android.permission.CAMERA	Allows access to the device's camera, including taking photos and recording videos.
5.	android.permission.CHANGE_WIFI_STATE	Grants an application the ability to change the device's Wi-Fi settings.
6.	android.permission.INTERNET	Allows an application to access the internet, including sending and receiving data over the network
7.	android.permission.PROCESS_OUTGOING_CALLS	Enables an application to monitor and manage outgoing phone calls.
8.	android.permission.READ_CONTACTS	Grants an application access to the device's contact list.
9.	android.permission.READ_INTERNAL_STORAGE	Allows an application to write data to internal storage devices.
10.	android.permission.WAKE_LOCK	Enables an application to prevent the device from going into sleep mode or locking the screen, allowing it to continue running in the background.
11	android.permission.READ_SMS	Grants an application access to the device's SMS messages, including reading and managing SMS messages, including receiving OTP (One-Time Passwords).
12	android.permission.SEND_SMS	Allows an application to send SMS messages, including sending and managing SMS messages.
13	android.permission.PROCESS_OUTGOING_CALLS	Enables an application to monitor and manage outgoing phone calls, including intercepting calls, managing call logs, and controlling call settings

2.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial step in the workflow, performed to examine the structural characteristics and statistical patterns within a dataset before model development. EDA involves using visualisations and statistical summaries to uncover underlying patterns, detect outliers, identify errors or corrupt data, and reveal relationships between variables, all of which inform subsequent modelling decisions and help validate initial assumptions about the data [27]. All EDA procedures were conducted using the pandas, seaborn, and matplotlib libraries in Python, ensuring purely descriptive exploration without exposing test data to the modelling process.

2.3.1 Distribution of Active Permissions per Application

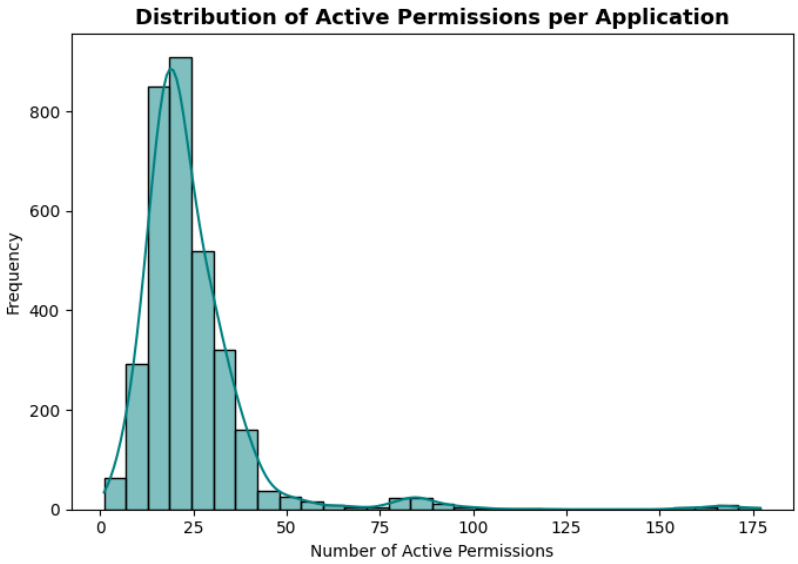

Figure 2. Distribution of Active Permissions per Application

A statistical summary of the total active permissions requested per application (referred to as permission_count) shows a mean of 24.57, a standard deviation of 16.99, a minimum of 1, and a maximum of 177. The histogram in Fig. 2 shows a right-skewed distribution, with most applications requesting between 15 and 30 permissions, while a few outliers exhibit exceptionally high counts. This long-tail pattern suggests that while most apps use moderate permissions for functional purposes, a minority of highly permission-intensive applications may pose greater security risks. Such overprivileged behaviour often correlates with malicious intent, as excessive access rights enable unauthorised operations on device resources.

2.3.2 Comparison of Permission Counts

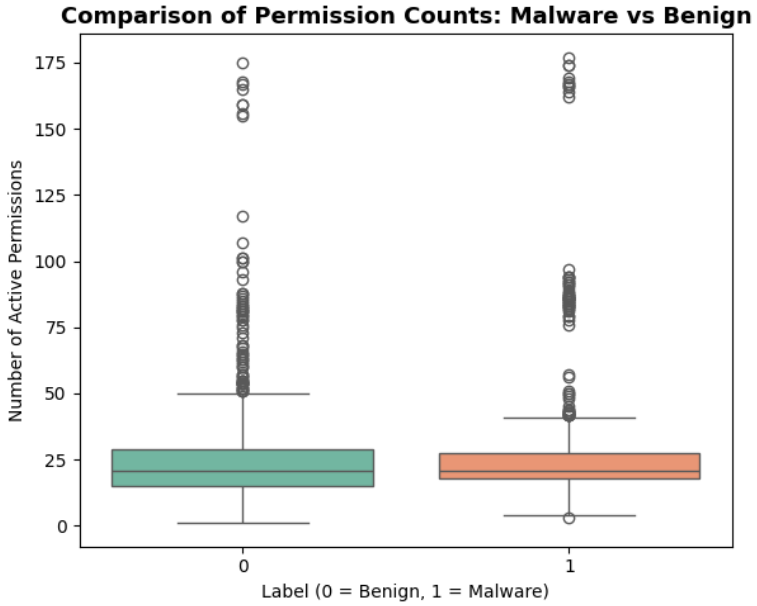

Figure 3. Comparison of Permission Counts

The boxplot in Fig. 3 visualises the difference in permission intensity between benign and malicious applications. Malware samples tend to have higher median permission counts and wider interquartile ranges, reflecting greater variation in the number of permissions requested. Several outliers are visible, representing applications that demand an unusually large number of permissions, a characteristic often associated with privilege abuse or spyware-like activity. The benign group shows a more concentrated distribution, indicating that legitimate applications typically operate within narrower permission scopes.

2.3.3 Feature Correlation Heatmap

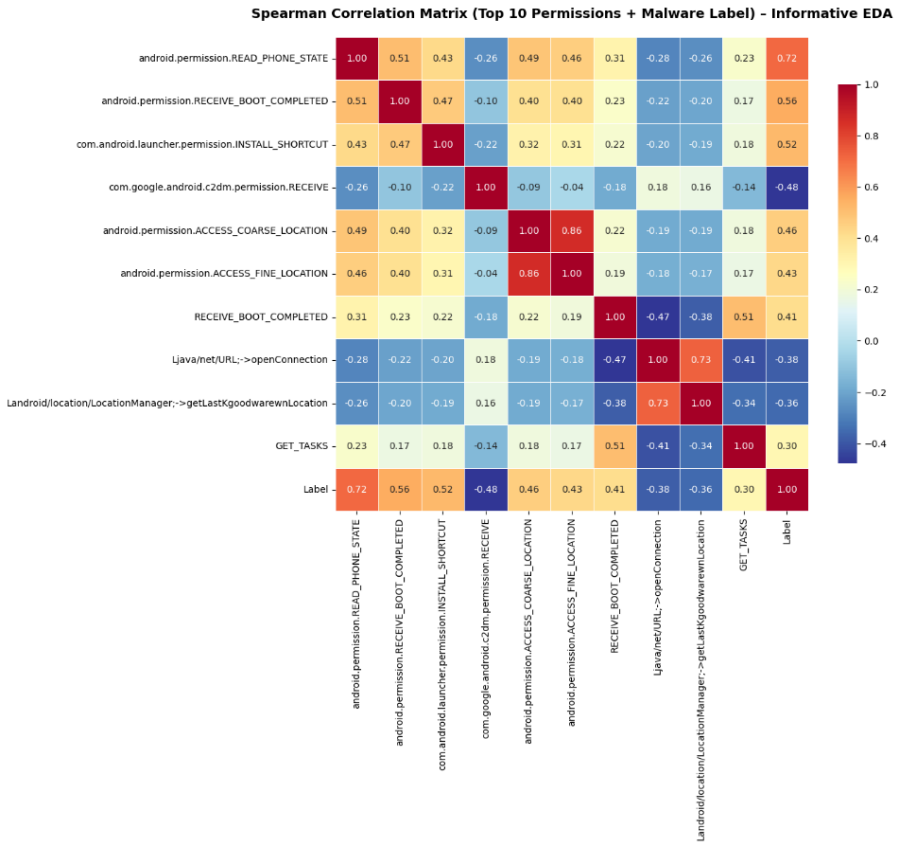

Figure 4. Correlation Heatmap

Fig. 4 visualises the Spearman correlation coefficients among the ten permission features most strongly correlated with the malware label. The bottom row and rightmost column show each feature's relationship with the target class, indicating that several permissions have strong positive correlations with malicious behaviour. The top-ranked features include read phone state ($\rho = 0.72$), receive boot completed notifications ($\rho = 0.56$), and installing shortcuts ($\rho = 0.52$), all of which are linked to high-risk operations such as device identification, persistence after reboot, and silent shortcut installation. Additional correlations are observed for access coarse location ($\rho = 0.46$), access fine location ($\rho = 0.43$), and network-related APIs such as Ljava/net/URL;->openConnection ($\rho = 0.38$), which are commonly associated with user tracking and unauthorised communication. To prevent bias and data leakage, the actual feature selection in this study is performed independently in the modelling phase (Section 2.4) using SelectKBest with mutual information scoring within a cross-validation pipeline.

2.4 Feature Selection

Feature selection is a critical step in machine learning to reduce dimensionality, improve model performance, and prevent overfitting [28]. In this research, we compared two feature selection approaches to determine the most effective method for malware classification:

2.4.1 Manual Spearman Correlation

The first approach involved manually selecting the top 20 features based on Spearman correlation coefficients with the target variable (malware status). The Spearman rank correlation coefficient (ρ) measures the strength of a monotonic relationship between two ranked variables [29] and is defined as:

$$p = 1 - \frac{6 \sum d_i^2}{n\,(n^2 - 1)}, \tag{1}$$

where d_i represents the difference between the ranks of paired values and n is the number of observations.

While intuitive and computationally simple, this baseline method presents several limitations:

1. Data leakage risk: Computing correlations on the entire dataset (including the test set) before splitting can lead to overly optimistic performance estimates.
2. Static selection: The same set of features is used across all models, ignoring differences in algorithmic sensitivity or inductive bias.
3. Limited to linear relationships: Correlation captures only monotonic linear relationships and may fail to detect non-linear dependencies between features and the target variable.

Consequently, although this method serves as a meaningful interpretive baseline, it is not used in the final modeling pipeline due to concerns about its reproducibility and bias.

2.4.2 Automated Select KBest with Mutual Information

To address the limitations of manual selection, we implemented an automated feature selection method using SelectKBest with mutual information classification as the scoring function. This approach quantifies both linear and non-linear dependencies between features and the target variable, making it more robust for heterogeneous binary data such as Android permissions [30]. The mutual information between a feature $X$and target $Y$is defined as:

$$I(X,Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\, log - \frac{p(x,y)}{p(x)p(y)}. \quad (2)$$

This measure evaluates how much knowing X reduces uncertainty about Y, providing a model-agnostic criterion for feature relevance. In this research, the mutual information selector was integrated directly within a scikit-learn Pipeline, ensuring that feature selection occurs only on the training folds during Stratified K-Fold cross-validation. This design provides three major benefits:

1. Leakage prevention: Feature selection is confined to training subsets within each fold, ensuring the test data remain unseen throughout optimisation.
2. Joint optimisation: The number of selected features (columns) (k) is tuned in conjunction with model hyperparameters via GridSearchCV, aligning the feature subset with each classifier's structure. It means we don't have to use all 328 columns to train the models.
3. Non-linear dependency capture: Mutual information can detect complex statistical relationships beyond linear correlation, allowing the model to exploit diverse patterns in permission usage.

2.5 Data Preprocessing

The preprocessing phase was conducted to ensure the dataset was free of errors, duplicates, and inconsistencies before model training. The dataset initially contained 4,464 records, of which 1,172 duplicates were identified and removed using the duplicated() function in pandas. After deduplication, the dataset consisted of 3,292 unique samples. The Label column, which indicated whether an application was benign or malware, was standardised into binary values—benign as 0 and malware as 1. This standardisation was done in-place using the map() function, and the distribution of labels was verified with value_counts(). This ensures that the dataset accurately represents the two target categories.

Following deduplication and label standardisation, the dataset was split into training/validation (80%) and test (20%) subsets using Stratified K-Fold cross-validation to preserve the proportions of both classes in each fold. The training set was used for feature selection and hyperparameter tuning, while the test set was kept separate to avoid data leakage. StandardScaler was applied only to the Logistic Regression model within the pipeline because it is sensitive to feature magnitudes and relies on linear combinations of input variables. Tree-based models (Random Forest and Gradient Boosting Machine) were not scaled, as they inherently partition the feature space based on threshold values and are thus invariant to feature scaling [31].

2.6 Modelling and Evaluation

After splitting the data, the models were trained using three distinct machine learning algorithms: Logistic Regression, Random Forests, and Gradient Boosting Machine. The models represent distinct learning paradigms: linear (LR), bagging ensemble (RF), and boosting ensemble (GBM), allowing comprehensive assessment of performance trade-offs. These three algorithms consistently demonstrate strong

performance in permission-based malware detection [21][22][23][24], providing a robust baseline for comparison.

2.6.1 Logistic Regression

The algorithm's name, logistic regression, has led to the assumption that it is a regression tool, but the method is actually for classification [32]. Logistic Regression is a machine learning method for classifying a binary problem, where the outcome is either a binary yes/no or a binary positive/negative [33]. It is also one of the simplest Machine Learning algorithms due to its interpretable and accurate results. Logistic regression uses the following formula:

$$P(Y-1) = \frac{1}{1 + e - (\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n}). \quad (3)$$

This formula is known as the sigmoid function, which maps the linear combination of the independent variables to a probability value between 0 and 1. The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad (4)$$

where z is the input to the function. This function has a unique property: it maps all real numbers to a value between 0 and 1, making it suitable for modelling binary outcomes.

2.6.2 Random Forests

Random forests are an efficient non-parametric prediction method for classification (categorical outcome) and regression (continuous outcome) applications **[34]**. Random Forests aim to improve upon the limitations of single decision trees, such as overfitting, through techniques like bagging (bootstrap aggregating) and feature randomness when building each tree, which helps to ensure that the trees are uncorrelated and that the ensemble model has lower variance **[35]**. When performing classification on Random Forests, the following formula is used:

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2. \quad (5)$$

In which the formula uses probability to determine the Gini of every branch, in this case, it determines the probability of an app to be 0 or 1 based on the permission needed.

2.6.3 Gradient Boosting Machine

Gradient Boosting Machine (GBM) is an ensemble learning algorithm in machine learning that utilises multiple weak models, typically decision trees, to produce a strong predictive model [36]. The method works by sequentially adding predictors to an ensemble, each correcting its predecessor by focusing on errors from previous models. GBM optimises a loss function over iterations using gradient descent, making it adaptable to a variety of predictive modelling problems [37]. Gradient boosting itself uses an iterative update of the model that could be represented as:

$$F_m(x) = F_{m-1}(x) + n \,.\, h_m(x) \,. \quad (6)$$

After the initial development and application of each model, GridSearchCV will be utilised to fine-tune the hyperparameters. This step is crucial to prevent overfitting, which can negatively impact the model's accuracy. The process involves defining a parameter grid for each model, specifying the range of values to test for each hyperparameter. GridSearchCV will then systematically explore various combinations of these parameters to identify the set that yields the best performance. Once the optimal parameters are determined, the models will be retrained using these settings to enhance their predictive capabilities.

2.6.4 Evaluation Metrics

Each model was evaluated on the unseen test set using multiple complementary metrics to assess classification effectiveness:

1. Accuracy measures the overall correctness of the predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (7)$$

2. Precision quantifies how many predicted malware instances were correct:

$$Precision = \frac{TP}{TP + FP}. \quad (8)$$

3. Recall (or Sensitivity) measures how many actual malware instances were correctly identified:

$$Recall = \frac{TP}{TP + FN}. \quad (9)$$

4. F1-Score is the harmonic mean of precision and recall:

$$F1 = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall}. \quad (10)$$

These metrics collectively ensure that both the detection capability and reliability of positive predictions are considered.

In addition, two threshold-independent metrics were calculated:

1. ROC-AUC (Receiver Operating Characteristic – Area Under Curve) measures the model's ability to distinguish between malware and benign apps across thresholds.
2. PR-AUC (Precision–Recall Area Under Curve) focuses on precision-recall trade-offs, particularly valuable when the positive class is less frequent.

Both are computed as the integral (area) under the respective curves of true positive rate versus false positive rate (ROC), and precision versus recall (PR).

2.6.5 Calibration Analysis

To ensure that probabilistic outputs from the models were interpretable and reliable, a calibration analysis was performed. Calibration curves (reliability diagrams) were generated to compare the predicted probabilities with the actual fraction of positive outcomes [38]. Models close to the diagonal line indicate well-calibrated confidence estimates. Calibration quality was quantified using the Brier Score, defined as:

$$BS = \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2, \quad (11)$$

where p_i is the predicted probability and y_i is the true label. A lower Brier Score indicates better calibration.

2.7 Study

Following the primary model evaluation, additional analyses were conducted to validate the interpretability and robustness of the proposed framework. These studies provide deeper insights into model behaviour, feature relevance, and performance stability under varying conditions.

2.7.1 Model Explainability (SHAP Analysis)

To enhance interpretability, SHAP (SHapley Additive exPlanations) was applied to quantify the contribution of each permission feature to individual predictions. SHAP values are grounded in cooperative game theory, where each feature's contribution to the model's output is assessed relative to a baseline [39].

2.7.2 Ablation Study

An ablation study was performed to investigate how the number of selected features (k) influences model performance [40], and to compare two feature selection strategies: Mutual Information (MI) and Spearman Correlation (Corr). Using Gradient Boosting Machine (GBM) as the reference classifier, the number of selected features was varied across $k = \{10,20,30,40,60,80\}$. Each configuration was evaluated using five-fold cross-validation on the training data and further validated on the hold-out test set.

3. RESULTS AND DISCUSSION

3.1 Model Performance Comparison

Following the methodology outlined in Section 2, three machine learning algorithms—Logistic Regression (LR), Random Forest (RF), and Gradient Boosting Machine (GBM)—were trained and evaluated using a leakage-free pipeline integrating SelectKBest feature selection, GridSearchCV hyperparameter optimisation, and stratified 5-fold cross-validation. All models were evaluated on an unseen hold-out test set (659 samples, 20% of the cleaned dataset) to assess their generalisation capability.

3.1.1 Hyperparameter Optimisation Results

GridSearchCV systematically explored combinations of feature count (k) and model-specific hyperparameters to identify the optimal configuration for each classifier. Table 2 summarises the best hyperparameters discovered through cross-validated grid search.

Table 2. Optimal Hyperparameters from GridSearchCV

Model	Feature Count (k)	Key Hyperparameters	CV ROC-AUC
Logistic Regression	40	C=1.0, solver='lbfgs', class_weight='balanced'	0.9877
Random Forests	40	max_depth=10, n_estimators=400, class_weight='balanced'	0.9899
Gradient Boosting Machine	40	learning_rate=0.1, max_depth=3, n_estimators=200	0.9924

Key observations:

1. All three models converged on k=40 features as optimal, suggesting this feature dimensionality balances information gain with overfitting prevention
2. GBM achieved the highest cross-validation ROC-AUC (0.9924), indicating superior discrimination capability during training
3. The class_weight='balanced' parameter for LR and RF addressed the slight class imbalance (1,745 benign vs. 1,547 malware).

3.1.2 Metric Evaluation Results

Table 3 presents comprehensive performance metrics on the hold-out test set.

Table 3. Optimal Hyperparameters from GridSearchCV

Model	Accuracy	Precision (Malware)	Recall (Malware)	F1 (Malware)	Precision (Benign)	Recall (Benign)	F1 (Benign)	ROC-AUC	PR-AUC
Logistic Regression	94.34%	0.9513	0.9452	0.9482	0.9516	0.9570	0.9543	0.9880	0.9872
Random Forests	95.45%	0.9516	0.9516	0.9516	0.9570	0.9570	0.9570	0.9900	0.9901
Gradient Boosting Machine	96.05%	0.9671	0.9484	0.9577	0.9549	0.9713	0.9631	0.9924	0.9918

Performance analysis:

1. Gradient Boosting Machine emerged as the top performer, achieving:
   a. Highest test accuracy (96.05%).
   b. Best ROC-AUC (0.9924), indicating superior ability to distinguish malware across all classification thresholds.
   c. Best F1-score for both classes (Malware: 0.9577, Benign: 0.9631).
2. Random Forest demonstrated balanced performance:
   a. Strong accuracy (95.45%) with equal precision and recall for both classes (0.9516/0.9570).
   b. Slightly lower ROC-AUC than GBM (0.9899 vs. 0.9924).

c. Well-suited for scenarios requiring symmetric error costs.

3. Logistic Regression, despite being the simplest model, achieved competitive results (94.34% accuracy), confirming that permission-based malware detection exhibits substantial linear separability.

3.1.3 Confusion Matrix Analysis

Fig. 5 (a), Fig.5 (b), and Fig. 5 (c) present confusion matrices for Logistic Regression, Random Forests, and Gradient Boosting Machine after testing the hold-out data, visualising the distribution of correct and incorrect predictions.

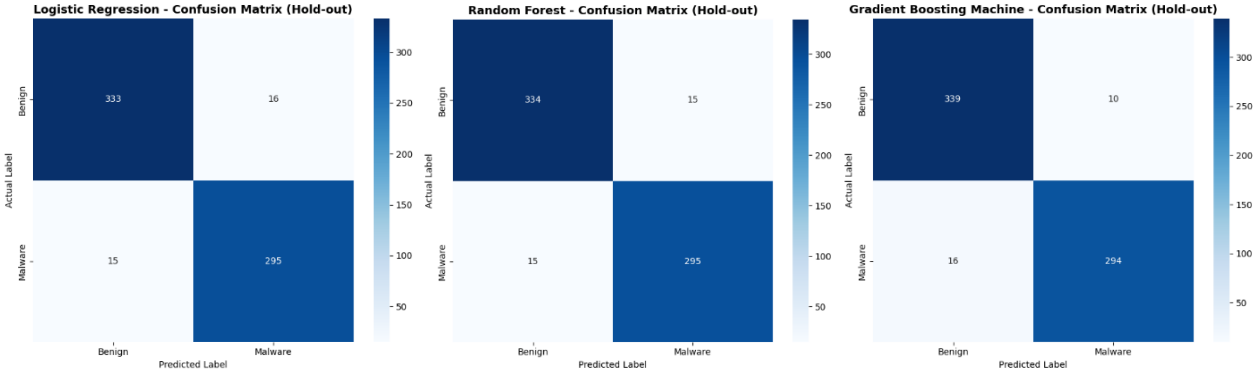

Figure 5. Confusion Matrix for (a) Logistic Regression, (b) Random Forests, (c) Gradient Boosting Machine

Insights about the confusion matrices:

1. GBM achieved the lowest false positive rate (10 benign apps misclassified as malware), crucial for user trust in production systems.
2. All models exhibited balanced error distribution between false positives and false negatives, indicating no systematic bias toward either class.
3. The low false negative count (14–16 missed malware) confirms that the models effectively identify malicious applications.

3.2 Feature Selection Strategy Comparison

To validate the effectiveness of the automated SelectKBest (Mutual Information) approach against the traditional manual Spearman–correlation–based feature selection, a comparative analysis was performed. This experiment addresses a key methodological gap in Android malware studies, where feature selection methods are frequently adopted without assessing their impact on generalisation and data leakage risk [22][23][24][25]. Both strategies were evaluated using identical data splits and classifiers—Logistic Regression (LR), Random Forest (RF), and Gradient Boosting Machine (GBM)—to isolate the effect of the selection method itself.

In the correlation-based baseline, the top 20 features were selected manually according to their absolute Spearman correlation coefficients with the malware label, computed on the entire dataset before train–test partitioning. This approach, while intuitive and interpretable, inherently risks data leakage because test samples influence feature rankings. In contrast, the automated pipeline integrated SelectKBest(mutual_info_classif) directly within a 5-fold Stratified K-Fold cross-validation using GridSearchCV, ensuring that feature relevance was recalculated exclusively on training folds. This design eliminates leakage, supports model-specific feature optimisation, and captures both linear and non-linear dependencies between permissions and malware status. Table 4 shows the most influential features in both methods.

Table 4. 10 Most Influential Features in Spearman vs Mutual Information

Rank (Spearman)	Feature Name	Spearman Correlation	MI Importance	Rank (GBM)
1	android.permission.READ_PHONE_STATE	0.7166	0.5779	1

Rank (Spearman)	Feature Name	Spearman Correlation	MI Importance	Rank (GBM)
2	android. permission.RECEIVE_BOOT_CO MPLETED	0.5556	0.0477	3
3	com.android.launcher.permission.IN STALL_SHORTCUT	0.5215	0.0464	4
4	com.google.android.c2dm.permissio n.RECEIVE	0.4790	0.1961	2
5	android.permission.ACCESS_COA RSE_LOCATION	0.4573	0.0029	14
6	android.permission.ACCESS_FINE _LOCATION	0.4316	0.0090	10
7	RECEIVE_BOOT_COMPLETED	0.4123	0.0333	5
8	Ljava/net/URL;->openConnection	0.3825	0.0117	8
9	Landroid/location/LocationManager; ->getLastKgoodwarewnLocation	0.3574	0.0136	7
10	GET_TASKS	0.2996	0.00098	18

Both approaches identified overlapping high-risk permissions, indicating feature-level semantic consistency between monotonic (Spearman) and entropy-based (Mutual Information) ranking. However, their relative order of importance and contribution magnitudes differ significantly. For instance, while READ_PHONE_STATE and C2DM.RECEIVE consistently dominates in both rankings; their relative weights shift under the MI criterion — the latter increasing sharply (from ρ=0.4790 → MI weight=0.1961) due to the capture of a non-linear dependency between network-related permissions and malware probability. Conversely, location permissions (ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION) drop in importance when evaluated with MI, indicating that their predictive contribution is largely redundant or confounded by correlated features ($\rho \approx 0.43–0.45$ but MI<0.01).

3.3 Calibration Analysis

Calibration analysis assesses whether a model's confidence scores correspond to actual outcome rates. Models close to the diagonal line indicate well-calibrated predictions. The calibration results are then evaluated using the Brier score, as shown in Table 5.

Table 5. Calibration Quality Metrics

Model	Brier Score	Mean Predicted Probability	Actual Positive Rate	Calibration Error
Logistic Regression	0.0399	0.472	0.470	0.0325
Random Forests	0.0401	0.475	0.470	0.1069
Gradient Boosting Machine	0.0344	0.468	0.470	0.0955

Interpretation:

1. Gradient Boosting Machine achieved the lowest Brier Score (0.0344), indicating the most accurate probability estimates among the three models.
2. All models demonstrate good calibration (Brier < 0.1), which is considered well-calibrated for binary classification tasks. This indicates the models are suitable for production systems that require reliable confidence estimates.
3. Mean Predicted Probability closely matches Actual Positive Rate across all models, with GBM showing the smallest deviation (0.468 predicted vs. 0.470 actual, $\Delta = 0.002$), confirming minimal systematic bias.
4. Lower calibration error and reliability scores for GBM indicate superior probabilistic predictions, making it more trustworthy for scenarios where confidence scores influence decision-making.

3.4 Ablation Study

An ablation study was conducted to investigate how feature count (k) influences model performance and to assess the robustness of the SelectKBest mutual information approach. Using the Gradient Boosting Machine as the reference classifier, the number of selected features was varied across $k \in \{10, 20, 30, 40, 60,$

80}, with each configuration evaluated using 5-fold cross-validation and validated on the hold-out test set. Additionally, a comparison was made between two feature selection strategies: Table 6 presents the comprehensive results of the ablation study.

Table 6. Ablation Study Results

k	Method	CV ROC-AUC	CV Std	Test Accuracy	Test F1-Macro	Test ROC-AUC
10	MI	0.9785	0.0051	94.69%	0.9467	0.9881
10	Correlation	0.9778	0.0060	94.23%	0.9422	0.9838
20	MI	0.9855	0.0042	94.84%	0.9482	0.9883
20	Correlation	0.9855	0.0023	95.45%	0.9543	0.9886
30	MI	0.9866	0.0033	95.30%	0.9528	0.9909
30	Correlation	0.9881	0.0021	96.05%	0.9604	0.9912
40	MI	0.9856	0.0031	95.30%	0.9528	0.9905
40	Correlation	0.9889	0.0014	95.90%	0.9588	0.9927
60	MI	0.9879	0.0014	95.14%	0.9513	0.9931
60	Correlation	0.9895	0.0023	95.90%	0.9588	0.9935
80	MI	0.9886	0.0021	95.60%	0.9558	0.9930
80	Correlation	0.9898	0.0019	96.21%	0.9619	0.9935

Key findings:

1. Both feature selection methods show performance saturation beyond $k = 30 - 40$ features. The test accuracy improvements become marginal as k increases from 40 to 80 ($\Delta < 0.5\%$), suggesting that the most informative permissions are captured within the first 30-40 features.
2. Interestingly, Spearman correlation-based selection achieved slightly higher test accuracy than mutual information across most k values. At $k = 80$, correlation-based selection reached 96.21% compared to MI's 95.60%. This suggests that, for this dataset, the linear monotonic relationships captured by Spearman's correlation are sufficient for effective malware detection.
3. Cross-validation standard deviation remains consistently low (< 0.006) across all configurations, indicating stable and reproducible performance. Correlation-based selection generally exhibits lower CV variance, particularly at $k = 40$ ($std = 0.0014$ vs. 0.0031 for MI).
4. Based on the ablation study, $k = 30 - 40$ features represent the optimal range, balancing performance gains with computational efficiency. Beyond $k = 40$, additional features contribute minimal accuracy improvements while increasing model complexity.

3.5 SHAP Analysis

To enhance model interpretability and validate feature relevance beyond global importance scores, this study employed SHAP on the GBM model using the hold-out test set as shown in Fig. 6.

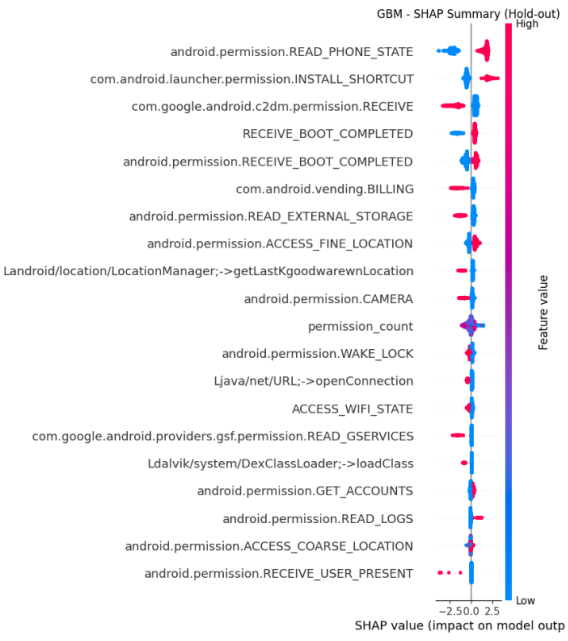

Figure 6. SHAP Summary Plot

Fig. 6 presents the SHAP summary plot, which illustrates each feature's impact on the model's output via its SHAP value distribution. Features positioned higher on the plot exert greater influence on malware classification decisions. The analysis shows that READ_PHONE_STATE, INSTALL_SHORTCUT, and RECEIVE are the three most influential features, consistently producing strong positive SHAP values when active (indicated by red points). This implies that granting these permissions significantly increases the model's predicted probability of an application being malware. These features are directly associated with sensitive operations such as device identification, silent shortcut creation, and background communication; behaviours commonly exploited by spyware and trojans.

In contrast, features located at the bottom of the plot exhibit near-zero SHAP distributions, indicating minimal contribution to the model's decision boundary. The symmetric concentration of blue points around zero for these permissions suggests limited discriminative power or contextual dependency, meaning that their presence or absence does not substantially alter the likelihood of malware.

3.6 Sample Prediction Analysis

To validate model behaviour and demonstrate practical applicability, four test scenarios were designed representing different risk profiles based on feature importance rankings. This analysis addresses the practical question: "How do models respond to applications with varying permission combinations?"

Features were categorised into risk levels based on GBM importance quartiles:

1. HIGH-RISK: Top 25% importance (10 features, led by READ_PHONE_STATE).
2. MEDIUM-RISK: Middle 50% importance (20 features).
3. LOW-RISK: Bottom 25% importance (10 features).

Table 7. Sample Prediction Summary

Test Case	Active Permissions (Examples)	Logistic Regression	Random Forest	Gradient Boosting Machine
High-Risk Only (7 features)	READ_PHONE_STATE, c2dm.RECEIVE, RECEIVE_BOOT_COMPLETED, INSTALL_SHORTCUT, READ_EXTERNAL_STORAGE	Malware (97.7%)	Malware (77.0%)	Malware (98.3%)
Mixed (3 High + 3 Medium)	READ_PHONE_STATE, c2dm.RECEIVE, RECEIVE_BOOT_COMPLETED, CAMERA, permission_count, WAKE_LOCK	Benign (94.9%)	Benign (88.9%)	Benign (85.3%)
Low-Risk Only (6 features)	loadLibrary, getSimOperator, RECEIVE_MMS, UPDATE_DEVICE_STATS, System.load, READ_USER_DICTIONARY	Benign (97.2%)	Benign (93.9%)	Benign (95.4%)
No Permissions	None	Benign (98.3%)	Benign (91.4%)	Benign (95.7%)

Key observations:

1. GBM demonstrates the highest confidence and aligns closely with test performance results.
2. RF provides more conservative estimates, indicating potential robustness in real-world deployment.
3. All models show consistent agreement on extreme cases; high-risk combinations yield malware predictions, while low/no permissions are benign.
4. Interestingly, the mixed-risk case resulted in benign predictions across all models with high confidence (85-95%). This suggests that permission combinations, rather than isolated high-risk features, drive malware classification. The absence of critical permission clusters (e.g., READ_PHONE_STATE + INSTALL_SHORTCUT + INTERNET simultaneously) may explain benign predictions despite individual high-risk permissions being present. This behaviour demonstrates model robustness against false positives, crucial for production deployment.

4. CONCLUSION

This study presented a structured, leakage-free machine learning framework for permission-based Android malware detection, comparing Logistic Regression, Random Forest, and Gradient Boosting Machine models within the CRISP-DM process model. Using a Kaggle-sourced dataset of 4,464 applications and 328 permissions, all models demonstrated strong classification capability, with GBM consistently outperforming the others (96.05% accuracy, 0.9924 ROC-AUC, and the lowest Brier Score of 0.0344). Feature importance and SHAP analyses confirmed that permissions such as READ_PHONE_STATE, RECEIVE_BOOT_COMPLETED, and INSTALL_SHORTCUT were key indicators of malicious behaviour. The comparative evaluation of manual Spearman correlation versus automated Mutual Information selection revealed that automated selection is more robust and prevents data leakage. From a theoretical perspective, the study provides empirical validation that ensemble boosting methods with entropy-based feature selection deliver the best balance between predictive accuracy and interpretability for static malware detection. In practice, the pipeline design ensures reproducibility and reliability for real-world applications such as mobile antivirus systems and app store vetting. However, this study is limited to static permission analysis and does not incorporate dynamic runtime behaviours or multi-source datasets. Future work should integrate hybrid static–dynamic features, cross-dataset validation, and longitudinal updates to address model drift and enhance detection resilience against adaptive malware.

Author Contributions

Arif Ridho Lubis: Conceptualisation, Methodology, Validation, Writing—Original Draft, Supervision. Dewi Wulandari: Data Curation, Software, Investigation, Writing—Improvement upon the draft and editing. Lilis Tiara Adha: Formal Analysis, Visualisation, Writing—Review and Editing. Tika Ariyani: Resources, Software, Validation. Yuyun Yusnida Lase: Project Administration, Data Curation, Writing—Review and Editing. Fahdi Saidi Lubis: Supervision, Methodology, Writing—Review and Editing. All authors reviewed, discussed, and agreed on the final version of the manuscript and contributed equally to the development and completion of this research.

Funding Statement

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

Acknowledgment

The authors would like to express their sincere gratitude to Politeknik Negeri Medan for its institutional support and for providing the necessary facilities and laboratory infrastructure throughout the research process, especially for data processing and analysis. We also extend our appreciation to colleagues who provided technical guidance, reviewers who offered constructive feedback to improve the manuscript, and everyone who contributed directly or indirectly to the completion of this studY.

Declarations

The authors declare that they have no conflicts of interest to report.

Declaration of Generative AI and AI-assisted technologies

The authors used generative AI (ChatGPT) only to assist with language polishing and formatting consistency (e.g., improving wording and ensuring uniform terminology). No AI was used to generate research content, perform analyses, or create/modify figures and tables. The authors reviewed the manuscript in full and remain responsible for its content.

REFERENCES

[1] I. Kandel and M. Castella, "HOW DEEPLY TO FINE-TUNE A CONVOLUTIONAL NEURAL NETWORK: A CASE STUDY USING A HISTOPATHOLOGY DATASET," *Comput. Secur.*, vol. 81, no. 5, p. ii, 2022, [Online]. Available: https://doi.org/10.1016/j.cose.2022.102785%0Ahttps://doi.org/10.1016/j.jksuci.2022.02.026%0Ahttps://doi.org/10.1016/j.i

jepes.2022.108733%0Ahttps://doi.org/10.1016/j.cmpb.2022.107141%0Ahttps://doi.org/10.1016/j.chemolab.2022.104534%0Ahttps://doi.org/10.101

[2] C. Easttom, "ANDROID OPERATING SYSTEM," *An In-Depth Guid. to Mob. Device Forensics*, 2021, doi: https://doi.org/10.1201/9781003118718-4.

[3] J. Lee, H. Jang, S. Ha, and Y. Yoon, "ANDROID MALWARE DETECTION USING MACHINE LEARNING WITH FEATURE SELECTION BASED ON THE GENETIC ALGORITHM," *Mathematics*, vol. 9, no. 21, pp. 1–20, 2021, doi: https://doi.org/10.3390/math9212813.

[4] R. Satrio Hadikusuma, L. Lukas, and E. M. Rizaludin, "METHODS OF STEALING PERSONAL DATA ON ANDROID USING A REMOTE ADMINISTRATION TOOL WITH SOCIAL ENGINEERING TECHNIQUES," *Ultim. J. Tek. Inform.*, vol. 15, no. 1, pp. 44–49, 2023, doi: https://doi.org/10.31937/ti.v15i1.3122.

[5] H. A. S. Alsharya, "LEVERAGING SOCIAL ENGINEERING TECHNIQUES FOR ETHICAL PURPOSES: AN APPROACH TO DEVELOP FAKE ANDROID APP FOR COLLECTING VALUABLE DATA DISCREETLY," *Wasit J. Comput. Math. Sci.*, vol. 3, no. 3, pp. 45–59, 2024, doi: https://doi.org/10.31185/wjcms.268.

[6] G. M. Naidoo and A. Reddy Moonasamy, "WHATSAPP AS A TOOL FOR TEACHING AND LEARNING DURING THE COVID-19 LOCKDOWN," *Univers. J. Educ. Res.*, vol. 10, no. 10, pp. 570–580, 2022, doi: https://doi.org/10.13189/ujer.2022.101003.

[7] A. O. Japinye, D. O. Ukeagu, and E. C. Ejianya, "ENHANCING MOBILE SECURITY THROUGH HAPTIC FEEDBACK: A MULTI-PARTICIPANT INVESTIGATION INTO MITIGATING SOCIAL ENGINEERING ATTACKS ON ANDROID DEVICES," *Eur. J. Comput. Sci. Inf. Technol.*, vol. 13, no. 33, pp. 1–15, 2025, doi: https://doi.org/10.37745/ejcsit.2013/vol13n33115.

[8] B. Urooj, M. A. Shah, C. Maple, M. K. Abbasi, and S. Riasat, "MALWARE DETECTION: A FRAMEWORK FOR REVERSE ENGINEERED ANDROID APPLICATIONS THROUGH MACHINE LEARNING ALGORITHMS," *IEEE Access*, vol. 10, no. August, pp. 89031–89050, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3149053.

[9] E.-M. Maier, L. M. Tanczer, and L. D. Klausner, *SURVEILLANCE DISGUISED AS PROTECTION: A COMPARATIVE ANALYSIS OF SIDELOADED AND IN-STORE PARENTAL CONTROL APPS*, vol. 2025, no. 2. Association for Computing Machinery, 2025. doi: https://doi.org/10.56553/popets-2025-0052.

[10] Z. Fang, W. Han, and Y. Li, "PERMISSION BASED ANDROID SECURITY: ISSUES AND COUNTERMEASURES," *Comput. Secur.*, vol. 43, no. 0, pp. 205–218, 2024, doi: https://doi.org/10.1016/j.cose.2014.02.007.

[11] F. Akbar, M. Hussain, R. Mumtaz, Q. Riaz, A. W. A. Wahab, and K. H. Jung, "PERMISSIONS-BASED DETECTION OF ANDROID MALWARE USING MACHINE LEARNING," *Symmetry (Basel).*, vol. 14, no. 4, 2022, doi: https://doi.org/10.3390/sym14040718.

[12] A. Muzaffar, H. Ragab Hassen, M. A. Lones, and H. Zantout, "AN IN-DEPTH REVIEW OF MACHINE LEARNING BASED ANDROID MALWARE DETECTION," *Comput. Secur.*, vol. 121, p. 102833, 2022, doi: https://doi.org/10.1016/j.cose.2022.102833.

[13] A. Iqubal and A. Payal, "MALWARE DETECTION TECHNIQUE FOR ANDROID DEVICES USING MACHINE LEARNING ALGORITHMS," *2024 Int. Conf. Comput. Sci. Commun. ICCSC 2024*, no. 9, pp. 0–3, 2024, doi: https://doi.org/10.1109/ICCSC62048.2024.10830310.

[14] P. Singh, P. Tiwari, and S. Singh, "ANALYSIS OF MALICIOUS BEHAVIOR OF ANDROID APPS," *Procedia Comput. Sci.*, vol. 79, pp. 215–220, 2019, doi: https://doi.org/10.1016/j.procs.2016.03.028.

[15] W. Xie and X. Zhang, "THE APPLICATION OF MACHINE LEARNING IN ANDROID MALWARE DETECTION," *2024 4th Int. Conf. Neural Networks, Inf. Commun. Eng. NNICE 2024*, pp. 1–4, 2024, doi: https://doi.org/10.1109/NNICE61279.2024.10498936.

[16] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A REVIEW OF ANDROID MALWARE DETECTION APPROACHES BASED ON MACHINE LEARNING," *IEEE Access*, vol. 8, pp. 124579–124607, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3006143.

[17] D. Revaldo, "ANDROID MALWARE DETECTION DATASET," Kaggle. Accessed: Mar. 15, 2024. [Online]. Available: https://www.kaggle.com/datasets/dannyrevaldo/android-malware-detection-dataset

[18] F. Martinez-Plumed *et al.*, "CRISP-DM TWENTY YEARS LATER: FROM DATA MINING PROCESSES TO DATA SCIENCE TRAJECTORIES," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 8, pp. 3048–3061, 2021, doi: https://doi.org/10.1109/TKDE.2019.2962680.

[19] R. Surendran, T. Thomas, and S. Emmanuel, "A TAN BASED HYBRID MODEL FOR ANDROID MALWARE DETECTION," *J. Inf. Secur. Appl.*, vol. 54, 2020, doi: https://doi.org/10.1016/j.jisa.2020.102483.

[20] Z. Sun, G. Wang, P. Li, H. Wang, M. Zhang, and X. Liang, "AN IMPROVED RANDOM FOREST BASED ON THE CLASSIFICATION ACCURACY AND CORRELATION MEASUREMENT OF DECISION TREES," *Expert Syst. Appl.*, vol. 237, no. PB, p. 121549, 2024, doi: https://doi.org/10.1016/j.eswa.2023.121549.

[21] Abdullah-All-Tanvir, I. Ali Khandokar, A. K. M. Muzahidul Islam, S. Islam, and S. Shatabda, "A GRADIENT BOOSTING CLASSIFIER FOR PURCHASE INTENTION PREDICTION OF ONLINE SHOPPERS," *Heliyon*, vol. 9, no. 4, p. e15163, 2023, doi: https://doi.org/10.1016/j.heliyon.2023.e15163.

[22] N. N. M. Nasri, M. F. A. Razak, R. R. Saedudin, S. M. Azmara, and A. Firdaus, "ANDROID MALWARE DETECTION USING MACHINE LEARNING," *Proc. - 2020 Innov. Intell. Syst. Appl. Conf. ASYU 2020*, vol. 9, no. 1, pp. 327–333, 2020, doi: https://doi.org/10.1109/ASYU50717.2020.9259834.

[23] K. A. Ahmed, K. Boopalan, K. Lokeshwaran, R. Sugumar, and C. Kotteeswaran, "ANALYSIS OF ANDROID MALWARE DETECTION USING MACHINE LEARNING TECHNIQUES," *AIP Conf. Proc.*, vol. 2935, no. 1, pp. 85–108, 2024, doi: https://doi.org/10.1063/5.0199036.

[24] A. Droos, A. Al-Mahadeen, T. Al-Harasis, R. Al-Attar, and M. Ababneh, "ANDROID MALWARE DETECTION USING MACHINE LEARNING," *2022 13th Int. Conf. Inf. Commun. Syst. ICICS 2022*, pp. 36–41, 2022, doi: https://doi.org/10.1109/ICICS55353.2022.9811130.

[25] A. Kapoor, H. Kushwaha, and E. Gandotra, "PERMISSION BASED ANDROID MALICIOUS APPLICATION DETECTION USING MACHINE LEARNING," *2019 Int. Conf. Signal Process. Commun. ICSC 2019*, pp. 103–108, 2019, doi: https://doi.org/10.1109/ICSC45622.2019.8938236.

[26] J. Brzozowska, J. Pizoń, G. Baytikenova, A. Gola, A. Zakimova, and K. Piotrowska, "DATA ENGINEERING IN CRISP-DM PROCESS PRODUCTION DATA – CASE STUDY," *Appl. Comput. Sci.*, vol. 19, no. 3, pp. 83–95, 2023, doi: https://doi.org/10.35784/acs-2023-26.

[27] K. M. Arsyad, A. Yunita, H. M. Krismartopo, A. S. Dimar, K. Dewi, and I. Madrinovella, "REVEALING INSIGHTS THROUGH EXPLORATORY DATA ANALYSIS ON EARTHQUAKE DATASET," *J. Sci. Informatics Soc.*, vol. 1, no. 1, pp. 1–6, 2023, doi: https://doi.org/10.57102/jsis.v1i1.18.

[28] D. Theng and K. K. Bhoyar, "FEATURE SELECTION TECHNIQUES FOR MACHINE LEARNING: A SURVEY OF MORE THAN TWO DECADES OF RESEARCH," *Knowl. Inf. Syst.*, vol. 66, no. 3, pp. 1575–1637, 2024, doi: https://doi.org/10.1007/s10115-023-02010-5.

[29] British Medical Journal, "ERRATUM: SPEARMAN'S RANK CORRELATION COEFFICIENT," *BMJ*, vol. 349, no. December, p. 7528, 2014, doi: https://doi.org/10.1136/bmj.g7528.

[30] J. Gonzalez-Lopez, S. Ventura, and A. Cano, "DISTRIBUTED SELECTION OF CONTINUOUS FEATURES IN MULTILABEL CLASSIFICATION USING MUTUAL INFORMATION," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 7, pp. 2280–2293, 2020, doi: https://doi.org/10.1109/TNNLS.2019.2944298.

[31] E. Dumitrescu, S. Hué, C. Hurlin, and S. Tokpavi, "MACHINE LEARNING FOR CREDIT SCORING: IMPROVING LOGISTIC REGRESSION WITH NON-LINEAR DECISION-TREE EFFECTS," *Eur. J. Oper. Res.*, vol. 297, no. 3, pp. 1178–1192, 2022, doi: https://doi.org/10.1016/j.ejor.2021.06.053.

[32] N. R. Panda, J. K. Pati, J. N. Mohanty, and R. Bhuyan, "A REVIEW ON LOGISTIC REGRESSION IN MEDICAL RESEARCH," *Natl. J. Community Med.*, vol. 13, no. 4, pp. 265–270, 2022, doi: https://doi.org/10.55489/njcm.134202222.

[33] Jajang, N. Nurhayati, and S. J. Mufida, "ORDINAL LOGISTIC REGRESSION MODEL AND CLASSIFICATION TREE ON ORDINAL RESPONSE DATA," *Barekeng*, vol. 16, no. 1, pp. 75–82, 2022, doi: https://doi.org/10.30598/barekengvol16iss1pp075-082.

[34] A. Devaux, C. Proust-Lima, and R. Genuer, "RANDOM FORESTS FOR TIME-FIXED AND TIME-DEPENDENT PREDICTORS: THE DYNFOREST R PACKAGE," 2023, [Online]. Available: http://arxiv.org/abs/2302.02670

[35] M. Denuit, D. Hainaut, and J. Trufin, "BAGGING TREES AND RANDOM FORESTS," in *Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions*, Cham: Springer International Publishing, 2020, pp. 107–130. doi: https://doi.org/10.1007/978-3-030-57556-4_4.

[36] R. M. Syafei and D. A. Efrilianda, "MACHINE LEARNING MODEL USING EXTREME GRADIENT BOOSTING (XGBOOST) FEATURE IMPORTANCE AND LIGHT GRADIENT BOOSTING MACHINE (LIGHTGBM) TO IMPROVE ACCURATE PREDICTION OF BANKRUPTCY," *Recursive J. Informatics*, vol. 1, no. 2, pp. 64–72, 2023, doi: https://doi.org/10.15294/rji.v1i2.71229.

[37] R. Auti, A. Bhatt, and S. Tidake, "COMPARATIVE ANALYSIS OF MACHINE LEARNING ALGORITHMS FOR GENOMIC DATA," *2023 1st DMIHER Int. Conf. Artif. Intell. Educ. Ind. 4.0, IDICAIEI 2023*, vol. 13, no. 1, pp. 217–223, 2023, doi: https://doi.org/10.1109/IDICAIEI58380.2023.10406455.

[38] B. Li *et al.*, "PREDICTING OUTCOMES FOLLOWING ENDOVASCULAR ABDOMINAL AORTIC ANEURYSM REPAIR USING MACHINE LEARNING," *Ann. Surg.*, vol. 279, no. 3, 2024, [Online]. Available: https://journals.lww.com/annalsofsurgery/fulltext/2024/03000/predicting_outcomes_following_endovascular.23.aspx

[39] Y. Nohara, K. Matsumoto, H. Soejima, and N. Nakashima, "EXPLANATION OF MACHINE LEARNING MODELS USING SHAPLEY ADDITIVE EXPLANATION AND APPLICATION FOR REAL DATA IN HOSPITAL," *Comput. Methods Programs Biomed.*, vol. 214, no. February, pp. 1–7, 2022, doi: https://doi.org/10.1016/j.cmpb.2021.106584.

[40] Y. Xue, X. Cai, and F. Neri, "A MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM WITH INTERVAL BASED INITIALIZATION AND SELF-ADAPTIVE CROSSOVER OPERATOR FOR LARGE-SCALE FEATURE SELECTION IN CLASSIFICATION," *Appl. Soft Comput.*, vol. 127, p. 109420, 2022, doi: https://doi.org/10.1016/j.asoc.2022.109420.