Research Article

Android Malware Detection Using Machine Learning with SMOTE-Tomek Data Balancing

Maryam Sufiyanu Masari [1], Maiauduga Abdullahi Danladi [1], Ilori Loretta Onyinye [2], and Loreta Katok Tohomdet [2,*]

[1] Department of Cybersecurity, Faculty of Computing, Air Force Institute of Technology, Kaduna 800001, Kaduna State, Nigeria; e-mail: msufiyanu90@gmail.com; abdullahi.maiauduga@afit.edu.ng.
[2] Department of Information and Communication Technology, Faculty of Ground and Communication Engineering, Air Force Institute of Technology, Kaduna 800001, Kaduna State, Nigeria; e-mail: lorettaonyinye@gmail.com; loretatohomdet@gmail.com
* Corresponding Author: Loreta Katok Tohomdet

Abstract: Android malware poses an increasing threat to mobile ecosystems due to the rapid growth of malicious applications and the rising complexity of attack strategies. This study addresses the challenge of reliable malware detection under imbalanced data conditions, which remains a critical limitation of many existing detection approaches. The objective of this research is to empirically validate the effectiveness of Random Forest for Android malware detection, while using comparative evaluation as supporting evidence. A machine learning–based detection framework is developed using the preprocessed TUANDROMD dataset, which contains 4,465 applications represented by 241 static and dynamic features. To mitigate class imbalance, SMOTE-Tomek is applied to the training subset, and model performance is evaluated on an unseen test set using accuracy, precision, recall, F1-score, and ROC AUC. The experimental results show that Random Forest achieves consistently strong and balanced performance, particularly in maximizing recall and minimizing false negatives, while outperforming baseline classifiers such as Decision Tree, K-Nearest Neighbors, and Support Vector Machine. Comparison with previously published studies further supports the robustness and reliability of the proposed approach. These findings demonstrate that combining ensemble-based learning with explicit data balancing provides a practical and computationally efficient solution for Android malware detection. The study concludes that Random Forest, when appropriately balanced and evaluated, offers a reliable alternative to more complex deep learning–based models for real-world Android security applications.

Keywords: Android malware detection; Cybersecurity; Imbalanced dataset; Intrusion detection; Machine learning; Malicious detection; Malware classification; Random Forest.

Received: November, 14th 2025
Revised: January, 13th 2026
Accepted: January, 14th 2026
Published: January, 18th 2026

1. Introduction

Malware, short for malicious software, refers to intentionally designed programs that disrupt computer systems, compromise data integrity, and violate user privacy across a wide range of platforms, including personal computers, mobile devices, and embedded systems [1]. The primary objectives of malware attacks typically include system damage, unauthorized access to resources, and the theft of sensitive information [2]. Over time, malware has evolved into various forms, such as viruses, worms, Trojans, backdoors [3], ransomware [4], adware, and spyware [5]. As illustrated in Figure 1, the number of malware incidents worldwide has increased substantially in recent years, posing serious threats to modern digital infrastructures, including IoT devices, personal computing systems, and Android-based smart devices [3].

The history of malware dates back to 1986, with the emergence of Brain, the first known personal computer virus, which rapidly spread and infected millions of systems [6]. Since then, the proliferation of internet-connected devices and the widespread adoption of smart technologies have accelerated the development of increasingly sophisticated malware variants. Modern malware employs advanced evasion techniques, such as obfuscation, polymorphism, metamorphism, packers, and crypters [7], which significantly complicate detection by traditional antivirus solutions. The rapid expansion of Android-based ecosystems—encompassing mobile banking, e-healthcare, e-learning, and smart manufacturing—has further amplified the attack surface, enabling cybercriminals to exploit system vulnerabilities for financial gain or to corrupt critical system resources [8]. Malware commonly infiltrates systems through untrusted downloads, malicious URLs, phishing campaigns, or compromised software updates [9]–[12].

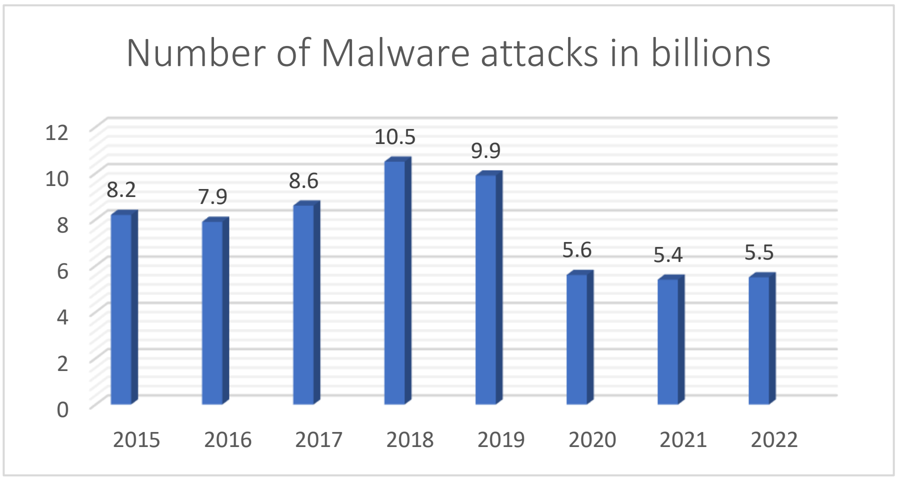


Figure 1. Annual representation of the quantity of malware incidents.

Conventional malware detection approaches primarily rely on static analysis techniques that compare suspicious files against known malware signatures. While effective against previously identified threats, these methods are inherently limited in detecting novel or zero-day malware due to the continuous evolution of malicious code and their reliance on signature databases [13]. To address these shortcomings, dynamic analysis techniques were introduced to monitor runtime behaviors in isolated environments, capturing system calls, network traffic, and memory usage to detect obfuscated or polymorphic malware [8]. Nevertheless, dynamic analysis may still fail when malware exhibits overlapping or concealed behavioral patterns. As a result, hybrid approaches combining static and dynamic analysis have been proposed to provide more comprehensive frameworks for malware assessment [14].

Despite these advances, the rapid growth in malware complexity necessitates more adaptive and scalable detection mechanisms. Consequently, recent studies have increasingly adopted machine learning (ML) and deep learning (DL) techniques to enable automated malware detection systems that can learn discriminative patterns from data [15], [16]. Traditional ML algorithms such as Decision Tree (DT), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Random Forest (RF) have been widely applied to classify benign and malicious applications based on extracted behavioral features [17]–[19]. In parallel, DL models have demonstrated strong performance by automatically learning feature representations from large-scale datasets [8], [20]. However, many DL-based approaches require substantial computational resources and remain insufficiently explored in scenarios involving highly imbalanced malware datasets.

Although several studies using benchmark datasets such as TUANDROMD report high detection accuracy [21]–[24], many do not explicitly examine the impact of class imbalance on model robustness and generalization. In malware detection, such an imbalance can bias classifiers toward the majority class, resulting in inflated accuracy while degrading detection reliability for minority-class samples. Motivated by this limitation, the present study does not aim to identify the best-performing classifier through direct algorithmic competition. Instead, it seeks to provide empirical evidence of RF suitability and robustness for Android malware detection under imbalanced data conditions.

Random Forest is adopted as the primary detection model due to its ability to handle high-dimensional feature spaces, mitigate overfitting through ensemble bagging, and maintain robustness against noisy and correlated static and dynamic features [10], [25], [26], such as permissions and API calls present in the TUANDROMD dataset. Furthermore, RF effectively captures complex nonlinear relationships in malware behavior and has consistently demonstrated stable, reliable performance in prior empirical studies of imbalanced classification problems. DT, KNN, and SVM are therefore employed strictly as baseline reference models to contextualize the empirical strengths of RF rather than to conduct a conventional comparative study.

In highly imbalanced malware datasets, conventional classifiers often become biased toward the majority class, leading to poor detection of minority-class samples. To mitigate this issue, data-level balancing techniques are commonly employed to improve learning fairness. In this study, SMOTE-Tomek is adopted as a hybrid data balancing strategy, where the Synthetic Minority Oversampling Technique (SMOTE) generates synthetic minority samples, while Tomek links remove overlapping and noisy instances near the decision boundary [10], [27]. This combination enables both class balancing and noise reduction, resulting in more reliable model training and evaluation.

Accordingly, the objective of this study is to empirically validate the effectiveness of RF for Android malware detection on the inherently imbalanced TUANDROMD dataset by employing appropriate data balancing strategies to ensure reliable learning and evaluation, while benchmarking its performance against baseline models and existing state-of-the-art approaches.

2. Related Work

The detection of malware on Android platforms has attracted significant research attention in recent years, driven by the rapid growth of malicious applications targeting mobile devices and the increasing sophistication of attack strategies. Numerous studies have explored both conventional ML and DL approaches to address this challenge, often relying on benchmark datasets such as TUANDROMD, Drebin, MalGenome, and CIC-2020.

Several studies have investigated traditional ML models using the TUANDROMD dataset to evaluate their suitability for Android malware classification. Iqubal et al. [21] conducted a comprehensive evaluation of multiple classifiers, including Logistic Regression, SVM, DT, KNN, Naïve Bayes, Neural Networks, RF, Gradient Boosting, AdaBoost, Bagging, and XGBoost, reporting near-perfect performance across various evaluation metrics. Similarly, Bhandari et al. [22] demonstrated the strong performance of RF, achieving high detection accuracy both with and without Principal Component Analysis (PCA). Other studies have explored the integration of static and dynamic features [23], the use of preprocessing pipelines and feature selection strategies [28], and alternative ML classifiers such as SMO and Simple Logistic [29], collectively highlighting the effectiveness of ML techniques for Android malware detection.

In parallel, advanced DL and hybrid frameworks have been proposed to enhance detection performance further. Wajahat et al. [24] introduced a Deep Neural Decision Forest model, while Ambekar et al. [30], [31] developed hybrid architectures that combine tabular learning with sequential modeling and explainability mechanisms. Transformer-based approaches, such as Trandroid [32], have also been explored, demonstrating the potential of DL models to capture complex behavioral patterns in Android applications. Despite their strong performance, these approaches often require substantial computational resources and may be less practical in resource-constrained environments.

However, a critical limitation in much of the existing literature is the insufficient consideration of class imbalance, which is inherent in malware datasets where benign applications typically outnumber malicious ones. This imbalance can bias learning algorithms toward the majority class, leading to inflated accuracy while degrading recall and detection reliability for malware [33]. Although some studies implicitly apply preprocessing techniques, the explicit impact of data-balancing strategies, such as SMOTE, on model robustness and minority-class detection performance remains underexplored.

Within this context, RF has emerged as a consistently stable and reliable classifier for Android malware detection. Its ensemble-based architecture enables effective handling of high-dimensional feature spaces, robustness to noisy and correlated features, and reduced overfitting through bagging. Empirical evidence from multiple studies [21], [22], [28] indicates that RF maintains strong and balanced performance across evaluation metrics, making it particularly suitable for imbalanced datasets when combined with appropriate resampling techniques.

Motivated by these observations, the present study aims to empirically validate the robustness of RF on the inherently imbalanced TUANDROMD dataset by explicitly incorporating class-balancing techniques such as SMOTE and Tomek. Rather than emphasizing algorithmic competition, DT, KNN, and SVM are employed strictly as baseline reference models to contextualize the stability and reliability of RF. A comprehensive evaluation framework based on precision, recall, F1-score, and ROC AUC is adopted to assess both overall classification performance and minority-class detection effectiveness, thereby providing a more reliable assessment of Android malware detection under imbalanced data conditions.

3. Proposed Method

This section presents the proposed analytical framework and methodology for Android malware detection using ML techniques. The overall workflow, illustrated in Figure 2, consists of data collection, preprocessing, class balancing, model development, and performance evaluation. Although multiple classifiers are implemented, RF is treated as the primary detection model, while DT, KNN, and SVMnare used strictly as baseline reference models.

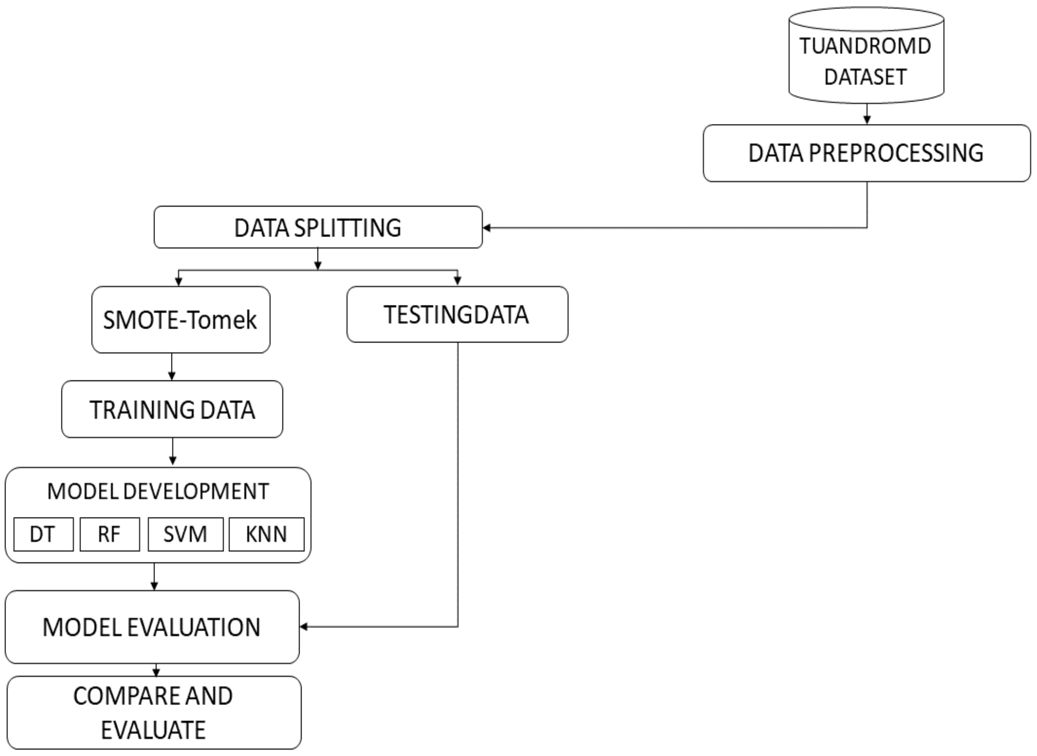


Figure 2. Proposed framework of the malware detection pipeline.

3.1. Data Collection

The dataset used in this study is the preprocessed version of the TUANDROMD dataset, obtained from the UCI Machine Learning Repository [34]. This benchmark dataset contains 4,465 Android application samples, each represented by 241 integer-valued features, and is widely used in Android malware detection research. The dataset is multivariate, belongs to the computer science domain, and is specifically designed for binary classification tasks. Each instance corresponds to either a malware or goodware application, with the class label defined as follows: 1 → Malware, 0 → Goodware.

The selection of TUANDROMD over other Android malware datasets is motivated by several factors. First, TUANDROMD provides comprehensive coverage of Android attack behaviors by including 71 malware families collected from multiple sources, enabling the development of a generalizable detection model rather than one limited to specific attack types. Second, the dataset contains rich static and dynamic features—such as permissions and API calls—which make it particularly suitable for evaluating ML models on high-dimensional data.

Compared to other commonly used datasets, such as Drebin and MalGenome, which contain samples collected prior to 2012, TUANDROMD offers a more recent and diverse representation of modern Android threats. Furthermore, the Contagio dataset lacks systematic categorization and sufficient diversity. As noted by Taheri et al. [35], reliance on outdated or limited datasets can reduce model robustness and expose detection systems to adversarial vulnerabilities. TUANDROMD addresses these limitations by providing a more representative dataset, thereby strengthening the validity of the proposed detection framework.

An initial statistical analysis of TUANDROMD revealed a pronounced class imbalance, with 3,565 malware samples (79.8%) and 899 goodware samples (20.2%). To address this imbalance and ensure fair evaluation, a data-level balancing strategy was applied to the training subset only. A summary of the dataset characteristics is provided in Table 1.

Table 1. Statistical overview of the TUANDROMD dataset.

Statistic	Value
Dataset shape	4,465 samples × 242 features
Malware instances	3,565 (79.8%)
Goodware instances	899 (20.2%)
Total missing values	242
Example features with high variance	KILL_BACKGROUND_PROCESSES, Ljavax/crypto/Cipher, Ljava/lang/Runtime

The dataset consists of 241 feature attributes and one binary class label. Although 242 missing values were identified, their proportion is negligible relative to the dataset size. Feature variance analysis indicates that permission-based and API-related features exhibit high variability, suggesting their strong discriminative potential. It is important to note that this analysis was performed only for exploratory data analysis (EDA); no feature selection or dimensionality reduction was applied, and all features were retained during model training. These observations further justify the use of robust ensemble-based classifiers and class balancing techniques.

3.2. Data Preprocessing

Data preprocessing is essential to ensure consistency, reliability, and fairness in model training. In this study, preprocessing consisted of missing value handling, feature scaling, dataset splitting, and class balancing. Since TUANDROMD is a feature-ready dataset, no additional feature extraction was performed.

3.2.1. Handling Missing Data

The dataset was examined for missing values using statistical summary checks and the isnull() function in Pandas. No systematic missing data patterns were detected across the 241 features, eliminating the need for imputation techniques such as mean or median replacement.

3.2.2. Feature Scaling

Because KNN and SVM rely on distance-based computations, feature scaling was applied to prevent attributes with larger numerical ranges from dominating the learning process. Min–Max normalization was used to scale all features into the range [0, 1], as defined in Equation (1):

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

where X = Original feature value, X_{min} = Minimum value of the feature, X_{max} = Maximum value of the feature and X' = Scaled feature value. This normalization ensures that no feature dominates due to its scale and maintains the interpretability of the data.

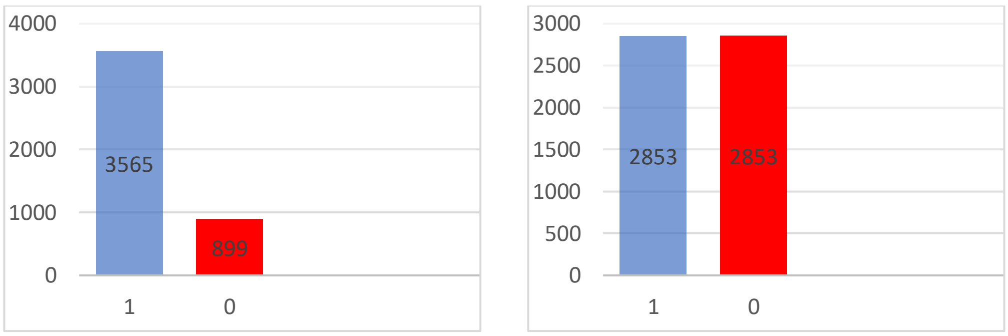
3.3. Train–Test Split and Cross-Validation

The dataset was divided into training and testing subsets using an 80:20 stratified split to preserve the original class distribution. The test set was kept completely unseen and used exclusively for final performance evaluation. Within the training set, stratified K-fold cross-validation was employed during model development and hyperparameter tuning. This strategy ensures that each fold maintains the original malware-to-goodware ratio, which is particularly important for imbalanced datasets. All resampling operations were performed only within the training folds to prevent data leakage and ensure unbiased evaluation.

3.3.1. Class Balancing with SMOTE-Tomek

To address the severe class imbalance in TUANDROMD, the training data was balanced using SMOTE-Tomek, a hybrid technique that combines oversampling and noise removal. SMOTE generates synthetic minority-class samples, while Tomek links remove overlapping and ambiguous instances near class boundaries, thereby improving class separability.

After the 80:20 split, the training set contained 2,852 malware samples and 719 goodware samples, while the test set contained 893 samples and remained untouched. SMOTE-Tomek was applied exclusively to the training set, resulting in a balanced distribution of malware and goodware samples, as shown in Figure 3.



(a) (b)

Figure 3. Class distribution before (**a**) and after applying SMOTE-Tomek (**b**).

3.4. Model Development

Four ML models were developed for malware detection: DT, KNN, SVM, and RF. RF is treated as the primary detection model, while the others serve as baseline references. The configuration of each classifier is summarized in Table 2.

Table 2. Model configuration

Classifier	Hyperparameters
Decision Tree	criterion = gini, max_depth optimized via CV
KNN	n_neighbors = 5 (grid search), distance = Euclidean
SVM	RBF kernel, C and γ optimized via grid search
Random Forest	n_estimators = 100, criterion = gini, max_features = sqrt

Table 2 summarizes the configuration of the classifiers used in this study. RF, which serves as the primary detection model, was configured with 100 decision trees using the Gini impurity criterion and square-root feature selection to enhance tree diversity and improve generalization performance. Its ensemble-based architecture enables robust learning from high-dimensional and correlated features, making it particularly suitable for Android malware detection.

For comparison purposes, DT, KNN, and SVM were implemented as baseline reference models. The DT classifier employed the Gini criterion, with the maximum tree depth optimized through cross-validation to reduce overfitting. The KNN model used Euclidean distance, with the number of neighbors optimized via grid search, while the SVM model utilized a radial basis function kernel with the regularization parameter C and kernel parameter γ optimized through grid search. For all models, training was performed on the SMOTE-Tomek–balanced training set to address class imbalance, while evaluation was conducted on the original, untouched test set to ensure unbiased performance assessment.

3.5. Model Evaluation

Model performance was assessed using Accuracy, Precision, Recall, F1-score, and ROC AUC, which together provide a comprehensive evaluation of binary classifiers under imbalanced conditions. Confusion matrices were used to quantify correct and incorrect predictions for both classes. While accuracy provides an overall performance summary, it can be misleading for imbalanced datasets such as TUANDROMD; therefore, greater emphasis is placed on recall, F1-score, and ROC AUC, which better reflect the reliability of malware detection.

4. Results and Discussion

This section presents the experimental results and discussion of the ML models developed for Android malware detection using the preprocessed and SMOTE-Tomek–balanced TUANDROMD dataset. The evaluation focuses on demonstrating the robustness of the proposed RF model while using DT, KNN, and SVM as baseline reference models.

4.1. Comparative Analysis of Classifiers

The comparative performance of the evaluated classifiers is summarized in Table 3, which reports Accuracy, Precision, Recall, F1-score, and ROC AUC for each model, both with and without SMOTE-Tomek-based balancing. This comparison highlights not only the relative performance of the classifiers but also the impact of data balancing on minority-class detection.

Table 3. Comparative performance of malware detection models.

Classifier	Without SMOTE-Tomek					With SMOTE-Tomek				
	Accuracy	Precision	Recall	F1-score	ROC-AUC	Accuracy	Precision	Recall	F1-score	ROC AUC
Random Forest	0.9913	0.9954	0.9937	0.9946	0.9990	**0.9933**	0.9917	**1.0000**	**0.9958**	**0.9998**
SVM	0.9812	0.9933	0.9832	0.9882	0.9970	0.9933	**0.9986**	0.9930	0.9958	0.9993
Decision Tree	0.9871	0.9902	0.9937	0.9920	0.9827	0.9888	0.9862	**1.0000**	0.9930	0.9805
K-Nearest Neighbors	0.9826	0.9906	0.9877	0.9891	0.9884	0.9765	0.9779	0.9930	0.9854	0.9578

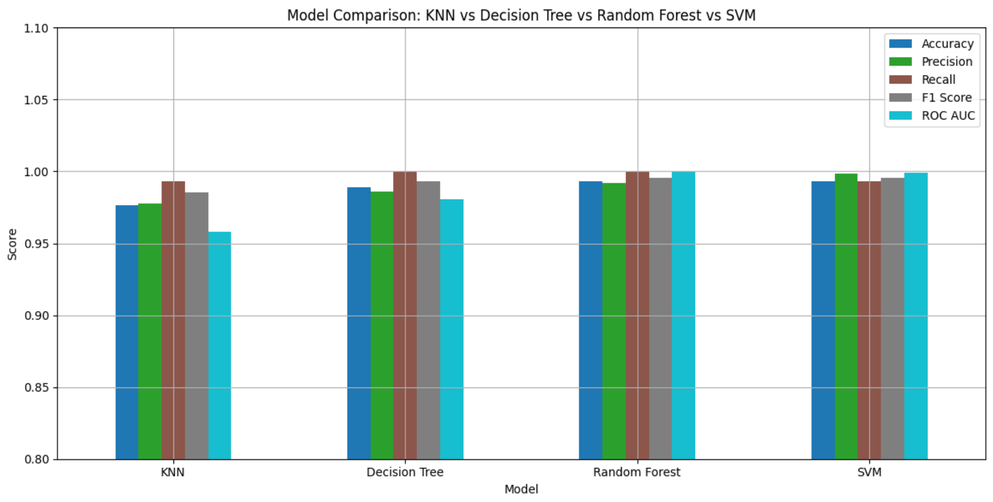


Figure 4. Performance comparison of DT, RF, KNN, and SVM classifiers.

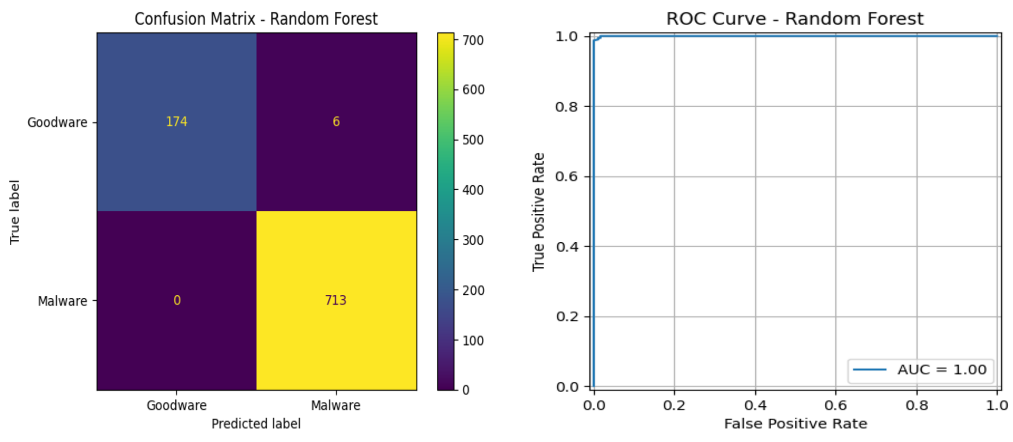
As shown in Table 3, all classifiers achieved high performance even without class balancing, indicating that the TUANDROMD dataset provides discriminative features for malware detection. However, performance improvements are consistently observed after applying SMOTE-Tomek. In particular, the RF model achieved perfect recall (1.0000) after balancing, eliminating all false negatives. This result confirms that performance gains are not solely attributable to the classifier design but are also strongly influenced by the data-balancing strategy.

A visual comparison of model performance is presented in Figure 4, which illustrates Accuracy, Precision, Recall, F1-score, and ROC AUC for all classifiers. From both Table 3 and Figure 4, RF emerges as the most reliable model, especially in terms of Recall and ROC AUC, which are critical for malware detection. While SVM achieves slightly higher precision, its recall is marginally lower than RF's. DT and KNN also demonstrate strong performance; however, their ROC AUC values indicate slightly lower generalization performance than ensemble-based models such as RF and SVM.

4.2. Confusion Matrix Analysis

To further analyze classification behavior, Figures 5–8 present the confusion matrices and ROC-AUC curves for all four classifiers evaluated on the test set. These visualizations provide insight into the distributions of false positives and false negatives, which are particularly critical in malware detection, where missed malware instances pose significant security risks.

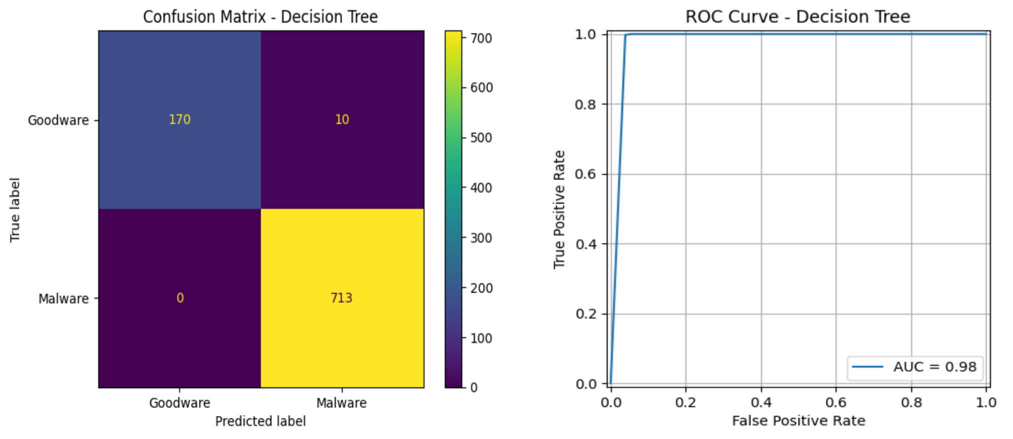
Figure 5 shows the confusion matrix and ROC-AUC curve of the RF model. RF correctly classified 174 goodware and 713 malware samples, misclassifying only 6 goodware instances as malware and failing to miss any malware samples. This result confirms perfect recall and demonstrates the high reliability of RF in detecting malicious applications.



(a) (b)

Figure 5. Confusion matrix and ROC-AUC of RF.

Figure 6 presents the results for the DT classifier, which correctly identified 170 goodware and 713 malware samples, misclassifying 10 goodware samples as malware and missing none. This explains its perfect recall and near-perfect precision.



(a) (b)

Figure 6. Confusion matrix and ROC-AUC of DT.

Figure 7 shows that KNN correctly classified 180 goodware and 708 malware samples, with no goodware samples misclassified as malware but 5 malware samples misclassified as goodware, resulting in slightly lower recall than RF and DT.

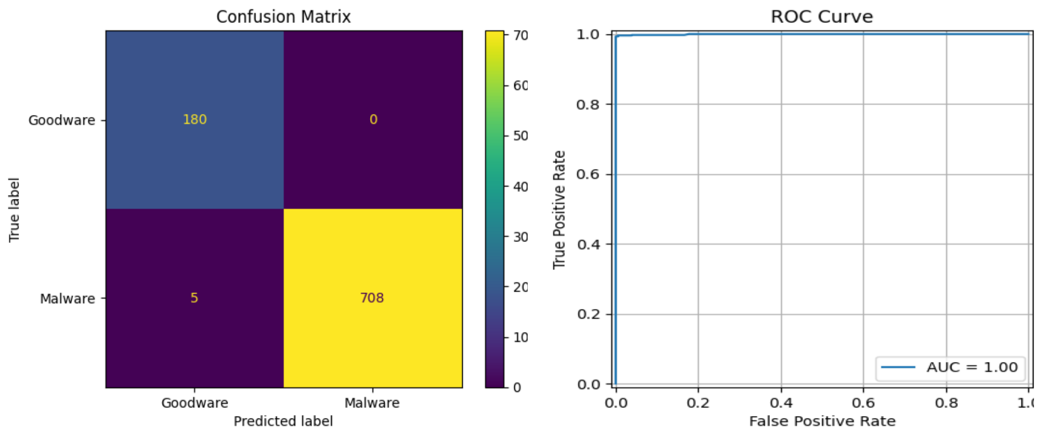


Figure 7. Confusion matrix and ROC-AUC of KNN.

Figure 8 presents the SVM results, where 179 goodware and 708 malware samples were correctly identified, with 1 goodware sample misclassified as malware and 5 malware samples misclassified as goodware. This resulted in very high precision, with minimal false positives, and a marginally lower recall due to a small number of missed malware instances.

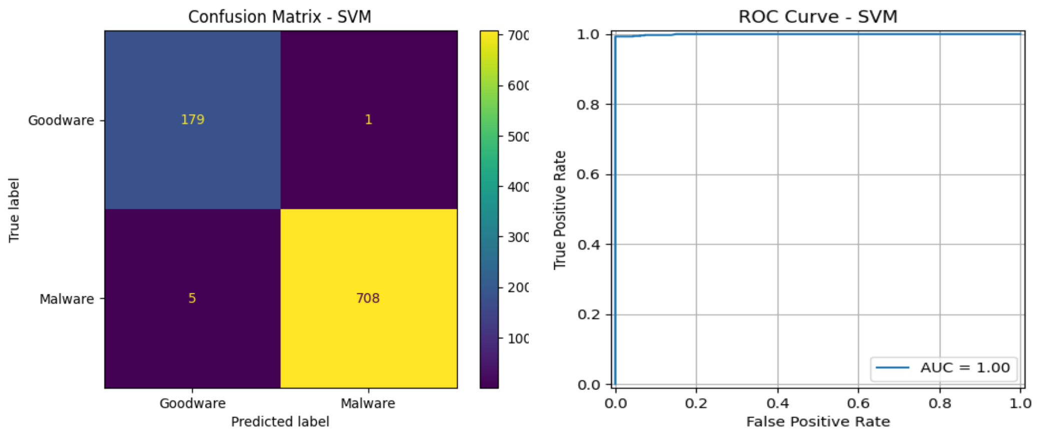


Figure 8. Confusion matrix and ROC-AUC of RF.

Overall, the confusion matrices and ROC-AUC curves indicate that RF offers the best balance between precision, recall, and generalization performance, making it particularly suitable for security-critical malware detection tasks.

4.3. Discussion and Comparison

The experimental results confirm the main hypothesis of this study: RF provides a robust and reliable solution for Android malware detection when trained on imbalanced data explicitly balanced using SMOTE-Tomek. Rather than relying on algorithmic competition, the findings demonstrate that the combination of an ensemble-based classifier and an appropriate data-balancing strategy leads to consistently high detection performance, particularly for security-critical malware samples.

Across all evaluated models, applying SMOTE-Tomek significantly improved learning fairness and minority-class detection, confirming the importance of explicitly addressing class imbalance in Android malware datasets. This observation aligns with prior research indicating that unbalanced data can bias classifiers toward the majority class and inflate accuracy without improving true detection reliability. The results in Table 3 and the confusion matrix analysis in Figures 5–8 collectively show that balancing the data is as critical as model selection for achieving reliable malware detection.

The superior performance of RF can be attributed to its ensemble architecture, which effectively handles high-dimensional, correlated features commonly found in Android malware datasets, such as permissions and API calls. Its ability to eliminate false negatives in the test set confirms its suitability for real-world malware detection scenarios, where missing a malicious application poses a higher risk than misclassifying benign software. The application of SMOTE-Tomek-based balancing proved critical across all models, significantly improving minority-class detection and ensuring fair evaluation. These findings reinforce the importance of addressing class imbalance explicitly in malware detection pipelines. To further contextualize the effectiveness of the proposed approach, Table 4 compares the RF model with existing state-of-the-art methods on the TUANDROMD dataset.

Table 4. Comparison of the proposed model with state-of-the-art methods on TUANDROMD

Reference	Accuracy	Precision	Recall	F1-score	ROC AUC
Wahajat et al. [24]	0.9860	0.9890	0.9940	–	–
Wahajat et al. [28]	0.9890	0.9700	1.0000	0.8420	–
Kacem and Tossou [32]	0.9925	0.9926	0.9925	0.9926	0.9876
Palabaş [33]	0.9700	0.8500	–	–	–
Akkaya and Altay [29]	0.9836	–	–	–	–
Random Forest (Ours)	0.9930	0.9920	1.0000	0.9960	0.9998

The comparative analysis reveals that the proposed RF model outperforms existing state-of-the-art models across key performance metrics. With an accuracy of 0.993, precision of 0.992, recall of 1.000, F1-score of 0.996, and an exceptionally high ROC AUC of 0.9998, the proposed model demonstrates superior predictive capability and robustness. In contrast, Wahajat et al. [24] and Palabaş [33] reported lower precision and overall accuracy, indicating reduced consistency and generalization. However, Kacem and Tossou [32] achieved competitive results with an accuracy of 0.9925 and F1-score of 0.9926, their ROC AUC of 0.9876 falls short of the proposed model's near-perfect classification performance. These results affirm that the proposed RF model provides more reliable and efficient predictions, marking a significant improvement over the state-of-the-art approaches.

5. Conclusions

This study investigated the effectiveness of RF as a robust Android malware detection model under imbalanced data conditions using the TUANDROMD dataset. Rather than focusing on algorithmic competition, the primary objective was to empirically validate the suitability of RF when combined with explicit data balancing, and to assess its reliability relative to commonly used baseline classifiers. The experimental results confirm that RF, when trained on SMOTE-Tomek–balanced data, provides stable and reliable malware detection, particularly in minimizing false negatives, which is critical for security-sensitive applications. The findings demonstrate that addressing class imbalance is a decisive factor in model performance and that combining ensemble learning with data-level balancing offers a practical and effective solution for Android malware detection. Baseline models such as DT, KNN, and SVM further contextualized this result, reinforcing the empirical robustness of RF without framing the study as a purely competitive evaluation.

Despite these contributions, this work has several limitations. The experiments were conducted using a single benchmark dataset, and although stratified cross-validation was applied during model development, generalization across datasets was not explored. In addition, the study focused on conventional ML models and did not include DL–based approaches, which may capture complementary behavioral patterns. Future research should extend this framework by evaluating the proposed approach across multiple, more diverse Android malware datasets, integrating hybrid or ensemble strategies that combine ML and DL models, and performing cross-dataset validation to assess generalization under real-world conditions. Incorporating adaptive or incremental learning mechanisms to handle evolving malware behaviors also represents a promising direction for further investigation.

Author Contributions: Conceptualization: M.S.M., M.A.D., I.L.O., L.K.T.; Methodology: L.K.T.; Software: L.K.T.; Validation: L.K.T., M.S.M., A.D.; Formal analysis: L.K.T.; Investigation: M.S.M., M.A.D., I.L.O.; Resources: M.S.M., M.A.D., I.L.O.; Data curation: L.K.T.; Writing—original draft preparation: M.S.M., M.A.D., I.L.O.; Writing—review and editing: I.L.O., L.K.T.; Visualization: L.K.T.; Supervision: M.S.M., A.D.; Project administration: M.S.M., M.A.D., I.L.O., L.K.T.; Funding acquisition: M.S.M., M.A.D., I.L.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The dataset is publicly available on the Kaggle platform and can be accessed via the link; https://archive.ics.uci.edu/dataset/855/tu-andromd+%28tezpur+university+android+malware+dataset%29.

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

References

[1] P. Kumar, G. P. Gupta, and R. Tripathi, "A Review on Intrusion Detection Systems and Cyber Threat Intelligence for Secure IoT-Enabled Networks," in *Big Data Analytics in Fog-Enabled IoT Networks*, Boca Raton: CRC Press, 2023, pp. 51–76. doi: 10.1201/9781003264545-3.

[2] A. K. Dey, G. P. Gupta, and S. P. Sahu, "A metaheuristic-based ensemble feature selection framework for cyber threat detection in IoT-enabled networks," *Decis. Anal. J.*, vol. 7, p. 100206, Jun. 2023, doi: 10.1016/j.dajour.2023.100206.

[3] P. Kumar, G. P. Gupta, and R. Tripathi, "Toward Design of an Intelligent Cyber Attack Detection System using Hybrid Feature Reduced Approach for IoT Networks," *Arab. J. Sci. Eng.*, vol. 46, no. 4, pp. 3749–3778, Apr. 2021, doi: 10.1007/s13369-020-05181-3.

[4] A. Aghamohammadi and F. Faghih, "Lightweight versus obfuscation-resilient malware detection in android applications," *J. Comput. Virol. Hacking Tech.*, vol. 16, no. 2, pp. 125–139, Jun. 2020, doi: 10.1007/s11416-019-00341-y.

[5] V. Sihag, M. Vardhan, and P. Singh, "A survey of android application and malware hardening," *Comput. Sci. Rev.*, vol. 39, p. 100365, Feb. 2021, doi: 10.1016/j.cosrev.2021.100365.

[6] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *J. Syst. Archit.*, vol. 112, p. 101861, Jan. 2021, doi: 10.1016/j.sysarc.2020.101861.

[7] M. Conti, V. P., and A. Vitella, "Obfuscation detection in Android applications using deep learning," *J. Inf. Secur. Appl.*, vol. 70, p. 103311, Nov. 2022, doi: 10.1016/j.jisa.2022.103311.

[8] S. K. Smmarwar, G. P. Gupta, and S. Kumar, "AI-empowered malware detection system for industrial internet of things," *Comput. Electr. Eng.*, vol. 108, p. 108731, May 2023, doi: 10.1016/j.compeleceng.2023.108731.

[9] P. H. Hussan and S. M. Mangj, "BERTPHIURL : A Teacher-Student Learning Approach Using DistilRoBERTa and RoBERTa for Detecting Phishing Cyber URLs," *J. Futur. Artif. Intell. Technol.*, vol. 1, no. 4, 2025, doi: 10.62411/faith.3048-3719-71.

[10] M. D. Okpor *et al.*, "Pilot Study on Enhanced Detection of Cues over Malicious Sites Using Data Balancing on the Random Forest Ensemble," *J. Futur. Artif. Intell. Technol.*, vol. 1, no. 2, pp. 109–123, Sep. 2024, doi: 10.62411/faith.2024-14.

[11] M. N. Musa and M. E. Irhebhude, "An Empirical Analysis of Injection Attack Vectors and Mitigation Strategies in Redis NoSQL Database," *J. Comput. Theor. Appl.*, vol. 2, no. 4, pp. 553–571, May 2025, doi: 10.62411/jcta.12640.

[12] C. Prakash, M. Lind, and E. De La Cruz, "Hybrid Real-time Framework for Detecting Adaptive Prompt Injection Attacks in Large Language Models," *J. Comput. Theor. Appl.*, vol. 3, no. 3, pp. 286–301, Jan. 2026, doi: 10.62411/jcta.15254.

[13] M. Alazab *et al.*, "A Hybrid Wrapper-Filter Approach for Malware Detection," *J. Networks*, vol. 9, no. 11, Dec. 1969, doi: 10.4304/jnw.9.11.2878-2891.

[14] T. Sharma and D. Rattan, "Malicious application detection in android — A systematic literature review," *Comput. Sci. Rev.*, vol. 40, p. 100373, May 2021, doi: 10.1016/j.cosrev.2021.100373.

[15] J. P. Ntayagabiri, Y. Bentaleb, J. Ndikumagenge, and H. El Makhtoum, "A Comparative Analysis of Supervised Machine Learning Algorithms for IoT Attack Detection and Classification," *J. Comput. Theor. Appl.*, vol. 2, no. 3, pp. 395–409, Feb. 2025, doi: 10.62411/jcta.11901.

[16] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *J. Netw. Comput. Appl.*, vol. 153, p. 102526, Mar. 2020, doi: 10.1016/j.jnca.2019.102526.

[17] S. Abijah Roseline and S. Geetha, "A comprehensive survey of tools and techniques mitigating computer and mobile malware attacks," *Comput. Electr. Eng.*, vol. 92, p. 107143, Jun. 2021, doi: 10.1016/j.compeleceng.2021.107143.

[18] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Comput. Secur.*, vol. 81, pp. 123–147, Mar. 2019, doi: 10.1016/j.cose.2018.11.001.

[19] S. Madan, S. Sofat, and D. Bansal, "Tools and Techniques for Collection and Analysis of Internet-of-Things malware: A systematic state-of-art review," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9867–9888, Nov. 2022, doi: 10.1016/j.jksuci.2021.12.016.

[20] S. I. Imtiaz, S. ur 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," *Futur. Gener. Comput. Syst.*, vol. 115, pp. 844–856, Feb. 2021, doi: 10.1016/j.future.2020.10.008.

[21] A. Iqubal, H. Happy, and S. K. Tiwari, "Android Malware Defense: Leveraging Machine Learning Models," in *2024 4th International Conference on Advancement in Electronics &amp; Communication Engineering (AECE)*, Nov. 2024, pp. 1356–1361. doi: 10.1109/AECE62803.2024.10911128.

[22] T. Bhandari, R. V. Romould, M. K. Gourisaria, V. Singh, R. Chatterjee, and D. K. Behera, "Unveiling Machine Learning Paradigms for Robust Malware Detection in Personal Data Security," in *2024 Sixth International Conference on Computational Intelligence and Communication Technologies (CCICT)*, Apr. 2024, pp. 226–231. doi: 10.1109/CCICT62777.2024.00045.
[23] H. Shah, V. Shah, N. Soni, V. Vadhavana, and K. Patel, "A Comparative Analysis for Android Malware Detection Using Machine Learning Models," in *2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, Jan. 2025, pp. 1040–1047. doi: 10.1109/ICMCSI64620.2025.10883385.
[24] A. Wajahat *et al.*, "An effective deep learning scheme for android malware detection leveraging performance metrics and computational resources," *Intell. Decis. Technol.*, vol. 18, no. 1, pp. 33–55, Feb. 2024, doi: 10.3233/IDT-230284.
[25] A. Çetin and S. Öztürk, "Comprehensive Exploration of Ensemble Machine Learning Techniques for IoT Cybersecurity Across Multi-Class and Binary Classification Tasks," *J. Futur. Artif. Intell. Technol.*, vol. 1, no. 4, pp. 371–384, Feb. 2025, doi: 10.62411/faith.3048-3719-51.
[26] B. Poudyal and M. Shakya, "Enhancing Earthquake Preparedness in Nepal through Machine Learning-Based Damage Prediction Models," *J. Futur. Artif. Intell. Technol.*, vol. 2, no. 3, pp. 476–492, Oct. 2025, doi: 10.62411/faith.3048-3719-109.
[27] D. R. I. M. Setiadi, K. Nugroho, A. R. Muslikh, S. W. Iriananda, and A. A. Ojugo, "Integrating SMOTE-Tomek and Fusion Learning with XGBoost Meta-Learner for Robust Diabetes Recognition," *J. Futur. Artif. Intell. Technol.*, vol. 1, no. 1, pp. 23–38, May 2024, doi: 10.62411/faith.2024-11.
[28] A. Wajahat *et al.*, "Outsmarting Android Malware with Cutting-Edge Feature Engineering and Machine Learning Techniques," *Comput. Mater. Contin.*, vol. 79, no. 1, pp. 651–673, 2024, doi: 10.32604/cmc.2024.047530.
[29] E. S. Akkaya and E. V. Altay, "Investigating the Performance of Machine Learning Methods for Malware Detection," in *EAI/Springer Innovations in Communication and Computing*, 2025, pp. 329–340. doi: 10.1007/978-3-031-88999-8_25.
[30] N. G. Ambekar, S. Thokchom, and S. Moulik, "TC-AMD: Android Malware Detection through Transfomer-CNN Hybrid Architecture," in *2024 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Dec. 2024, pp. 1–6. doi: 10.1109/ANTS63515.2024.10898633.
[31] N. G. Ambekar, N. N. Devi, S. Thokchom, and Yogita, "TabLSTMNet: enhancing android malware classification through integrated attention and explainable AI," *Microsyst. Technol.*, vol. 31, no. 3, pp. 695–713, Mar. 2025, doi: 10.1007/s00542-024-05615-0.
[32] T. Kacem and S. Tossou, "Trandroid: An Android Mobile Threat Detection System Using Transformer Neural Networks," *Electronics*, vol. 14, no. 6, p. 1230, Mar. 2025, doi: 10.3390/electronics14061230.
[33] T. Palabaş, "Android malware classification using basic machine learning methods," *Adıyaman Üniversitesi Mühendislik Bilim. Derg.*, vol. 11, no. 23, pp. 190–202, Aug. 2024, doi: 10.54365/adyumbd.1462488.
[34] P. Borah, D. Bhattacharyya, and J. Kalita, "Malware Dataset Generation and Evaluation," in *2020 IEEE 4th Conference on Information & Communication Technology (CICT)*, Dec. 2020, pp. 1–6. doi: 10.1109/CICT51604.2020.9312053.
[35] R. Taheri, M. Shojafar, F. Arabikhan, and A. Gegov, "Unveiling vulnerabilities in deep learning-based malware detection: Differential privacy driven adversarial attacks," *Comput. Secur.*, vol. 146, p. 104035, Nov. 2024, doi: 10.1016/j.cose.2024.104035.