# A Hybrid Feature Model for Android Malware Detection

**Hadeel Khalaf Alharbi**
College of Computer Science and Engineering, Taibah University, Madinah, Saudi Arabia
hadeel_al7rbi@hotmail.com (corresponding author)

**Rashiq Rafiq Marie**
College of Computer Science and Engineering, Taibah University, Madinah, Saudi Arabia
rmarie@taibahu.edu.sa

*Received: 6 February 2026 | Revised: 19 April 2026, 22 April 2026, 7 May 2026, and 23 May 2026 and | Accepted: 1 June 2026*

 *DOI: https://doi.org/10.48084/etasr.17995*

## ABSTRACT

**Android malware has become a significant cybersecurity threat due to the open nature of the Android platform. To address this issue, the present study proposes a hybrid malware detection model that combines static and dynamic features with feature selection and ensemble learning. Three feature selection methods, including L1, L2, and ElasticNet, were applied and evaluated using stratified 10-fold cross-validation. In addition, an ensemble model combining Support Vector Machine (SVM) and Random Forest (RF) was used to improve classification performance. Experiments were conducted on a hybrid dataset with almost 3,239 Android applications and 675 combined static and dynamic features. The experimental results demonstrate that hybrid features outperform both static and dynamic features. Among the tested feature selection methods, L2 demonstrated comparatively consistent performance and stable results, obtaining the highest accuracy of 0.9880 ± 0.0059, an F1-score of 0.9875 ± 0.0062, and an MCC value of 0.9759 ± 0.0119. However, Wilcoxon signed-rank testing indicated that the differences between the feature selection methods were not statistically significant ($p > 0.05$). The ensemble model demonstrated slightly more stable predictions. These findings indicate that the use of hybrid feature representation can be beneficial for Android malware detection. Future research should use more diverse datasets and explore deep learning techniques.**

***Keywords-Android malware detection; static and dynamic analysis; hybrid feature model; machine learning; feature selection; ensemble model***

## I. INTRODUCTION

Android operating systems are an open-source platform that offers users a high level of flexibility in managing and customizing their devices. While this openness supports wide adoption, it also introduces security risks that can be exploited by attackers. Malware attacks often begin when users install deceptive applications or fake updates, which allow malicious software to gain unauthorized access to devices. Such malware can modify or delete data and may result in serious consequences, including identity theft and the exposure of sensitive personal information [1].

Android malware detection has traditionally relied on two main approaches: static analysis and dynamic analysis. Static analysis examines application code without execution and extracts features, including permissions and command lines, using tools such as Apktool and dex2jar [2]. This approach is efficient but may fail to detect malware that hides its behavior. In contrast, dynamic analysis monitors application behavior during execution in a controlled environment, enabling the detection of malicious runtime activities [3-5]. Hybrid analysis is an effective approach that combines static and dynamic features to provide a better understanding of application behavior and improve detection accuracy.

Motivated by these findings, the current study proposes a hybrid malware detection framework that integrates static and dynamic features and applies multiple feature selection techniques, including L1, L2, and ElasticNet regularization. The main objective of this research is to enhance Android malware detection accuracy and contribute to the development of more reliable and efficient mobile security solutions. The proposed framework achieves a balance between detection accuracy and prediction reliability in Android malware classification. This makes the approach more practical for real-world cybersecurity use, especially in continuously evolving Android environments.

Despite the advancements in Android malware detection, many existing studies focus on either static or dynamic features separately or apply hybrid approaches without using a unified and consistent evaluation setup. In addition, the impact of different regularization-based feature selection methods is often not examined under similar experimental setups. This makes it

difficult to understand how these methods behave when applied to hybrid datasets. Malware detection based on static features has been widely explored [6]. A multi-tiered feature selection model combined with machine learning classifiers achieved high detection performance, reporting an accuracy of 96.28% using Random Forest (RF) [7]. In [8], genetic algorithm-based feature selection outperformed information gain when applied to static features; however, the need for validation on more recent datasets was highlighted.

Dynamic feature-based approaches have also shown promising results. Research has focused on extracting behavioral features such as API calls and system-level activities to improve malware detection performance [9]. In addition, deep learning-based methods have been introduced to enhance classification performance, reporting accuracy above 98% on multiple Android malware datasets [10]. Hybrid feature selection approaches further confirm the benefits of combining static and dynamic features. A system based on statistical feature engineering and metaheuristic optimization achieved high accuracy for both binary and multi-class classification tasks [11]. Deep learning-based hybrid approaches using models such as CNN-LSTM have also demonstrated improved detection accuracy for both static and dynamic analysis compared to traditional methods [12].

The present study addresses the knowledge gap by evaluating L1, L2, and ElasticNet feature selection methods within a single unified framework that combines static and dynamic features. All experiments are conducted under consistent preprocessing, and a cross-validation strategy is deployed to ensure fair comparison. Furthermore, an ensemble model combining Support Vector Machine (SVM) and RF is introduced to improve detection performance. The main contribution of this work lies in providing a practical evaluation framework of a feature selection model for hybrid Android malware detection. Moreover, the study offers a stable and effective classification approach.

## II. METHODOLOGY

### A. *Proposed Approach*

As illustrated in Figure 1, the proposed research framework detects Android malware using machine learning techniques. The framework is organized into four main phases. The first phase involves dataset preparation, in which Android applications are categorized as benign or malicious. The second phase applies feature selection techniques, including L1 (Lasso), L2 (Ridge), ElasticNet, and a baseline scenario without feature selection. In the third phase, machine learning models are trained using static, dynamic, and hybrid feature sets. Finally, the evaluation phase compares the classification performance across different feature selection strategies.

The proposed approach does not rely on raw data directly; instead, the datasets are first reviewed and prepared. At this stage, duplicate entries are removed, labels are converted to numerical values, and the data are checked for consistency. To avoid inconsistencies between static and dynamic sources, only the samples shared by both datasets are retained through an alignment step.

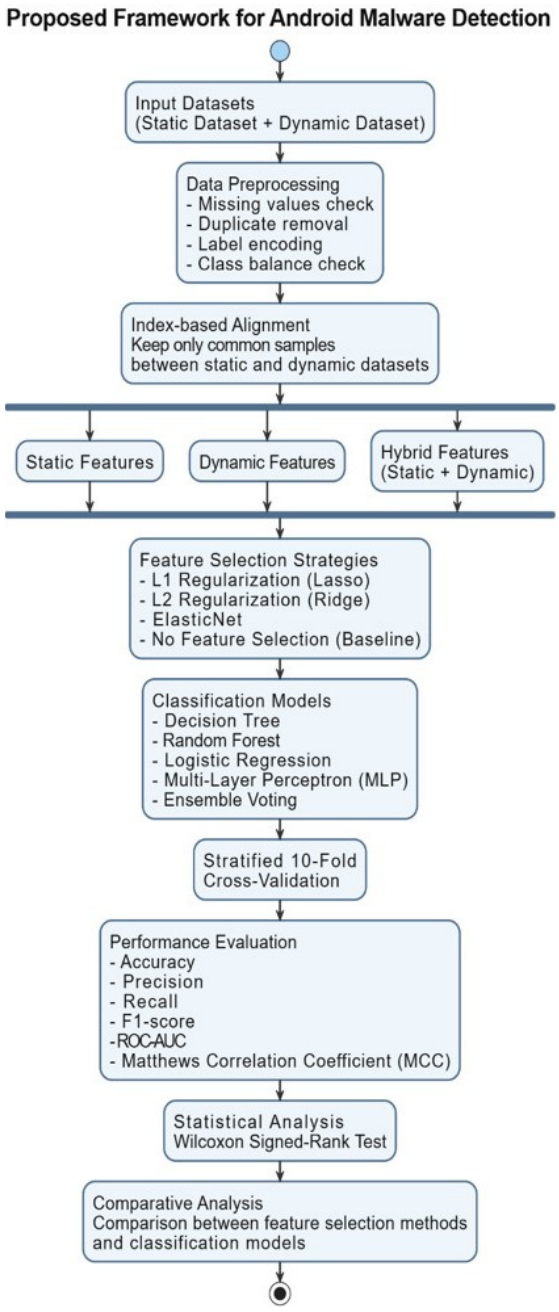


Fig. 1. Block diagram of the proposed framework.

Once the data are prepared, static and dynamic features are combined to form a hybrid representation. This step allows the model to capture different aspects of application behavior rather than relying on a single source of information. Feature selection is then performed using the L1, L2, and ElasticNet methods. Rather than focusing only on reduced features, the models are also tested on the original feature set to provide a clearer comparison. At the modeling stage, an ensemble model combining SVM and RF is used. This approach allows the model to benefit from both linear decision boundaries and non-linear feature interactions, resulting in more stable predictions.

### *B. Dataset Description*

This study uses an Android malware dataset that includes both static and dynamic features. Static features describe the internal structure of applications, while dynamic features capture their runtime behavior. Combining these features creates a hybrid representation that provides a more comprehensive view of application characteristics and supports more accurate malware detection. The present work employs the Android malware detection and classification dataset [13] and the Android malware dataset consisting of static features [14]. The combined dataset contains a total of 3,547 Android applications, including 1,800 benign and 1,747 malicious samples. Static features are extracted without executing the application and include permissions, API calls, command strings, and intents, resulting in 352 features. Dynamic features are collected during execution and include system calls, information leaks, cryptographic operations, and dynamic permissions, with a total of 323 features.

All features are represented in binary form, where a value of 1 indicates the presence of a feature, and 0 indicates its absence. The class label distinguishes between benign (B) and malicious (M) applications. Table I summarizes the feature categories and their distribution across static and dynamic datasets, as well as the total number of features in the hybrid representation.

TABLE I. SUMMARY OF FEATURE CATEGORIES

| Feature type | Category | Number of features |
|---|---|---|
| Static | API calls | 47 |
| Static | Permissions | 277 |
| Static | Command strings | 6 |
| Static | Intents | 22 |
| Dynamic | Cryptographic operations | 79 |
| Dynamic | Dynamic permissions | 71 |
| Dynamic | Information leaks | 123 |
| Dynamic | System calls | 50 |
| Total static features | — | 352 |
| Total dynamic | — | 323 |
| Total hybrid features | — | 675 |

### *C. Data Preprocessing*

The datasets were prepared carefully before starting the training process to ensure data quality and consistency. Both the static and dynamic datasets were first examined for missing values, and none were found, indicating that all features were complete. Duplicate records were identified and removed. This reduced the size of the static dataset from 3,547 to 3,293 samples, whereas the dynamic dataset decreased from 3,547 to 3,469 samples. For the classification task, the labels were converted into a numerical form, where benign applications were assigned the value 0, and malicious applications were assigned 1. The class distribution in both datasets remained relatively balanced. To build the hybrid dataset, an alignment step was performed to retain only the samples that appear in both datasets. This was achieved using index intersection to prevent mismatches between static and dynamic features. After alignment, both datasets were reduced to 3,239 samples, and the labels were verified to ensure consistency. Finally, a hybrid dataset was created by combining the aligned static and dynamic features, resulting in 3,239 samples and 675 features. A final validation confirmed that the dataset contains no missing values and no duplicate records, making it suitable for reliable machine learning experimentation.

### *D. Feature Selection*

This research focuses on improving Android malware detection by reducing feature dimensionality using regularization-based feature selection techniques. The study employs three methods: L1 (Lasso), L2 (Ridge), and ElasticNet. The objective is to identify the most relevant features while minimizing redundancy, thereby enhancing model efficiency and classification performance. The L1 regularization (Lasso) assigns coefficients to features and eliminates less important ones by shrinking their weights to zero, using the absolute value of the coefficients as a penalty term in the loss function [15]. In contrast, L2 regularization (Ridge) applies a squared penalty to the coefficients [16]. ElasticNet combines both L1 and L2 penalties to achieve a balanced feature selection process that benefits from sparsity and stability [17].

### *E. Classification Models*

The classification stage in this study is based on multiple machine learning models, including SVM [18, 19] and RF [20]. In addition, an ensemble model combining both classifiers is utilized to improve prediction performance. The use of multiple models allows the system to capture both linear and non-linear patterns in the data, leading to more stable and reliable results.

### *F. Experimental Setup*

Standard libraries such as Scikit-learn, NumPy, and Pandas were used for model development and evaluation. All experiments were conducted utilizing a fixed random seed (42) to ensure reproducibility. The feature selection methods (L1, L2, and ElasticNet) and classification models were implemented using consistent Scikit-learn settings across all experiments. Instead of relying on a single split, evaluation was performed utilizing stratified 10-fold cross-validation, which provides more stable results. Model performance was assessed employing commonly used metrics such as accuracy, precision, recall, F1-score, ROC-AUC [21], and MCC. The final comparison focused on how different feature selection methods influence performance, with additional statistical testing used to examine the significance of the observed differences.

## III. RESULTS

The experiments were carried out using Python on Google Colab. This platform provides an online environment that makes it easier to run machine learning models without depending on local hardware.

### *A. Cross-Validation Results*

To evaluate the proposed model, a stratified 10-fold cross-validation was used [22]. This method helps keep the class distribution balanced in each fold and provides more reliable results compared to a single train–test split. The results are reported as mean ± standard deviation for each metric. Table II presents the performance of the ensemble model using 10-fold cross-validation.

TABLE II. PERFORMANCE OF THE ENSEMBLE MODEL USING STRATIFIED 10-FOLD CROSS-VALIDATION

| Feature | ROC selection F1-score | AUC | Accuracy MCC | Precision | Recall |
|---|---|---|---|---|---|
| L2 | 0.9880 ± 0.0059 | 0.9859 ± 0.0062 | 0.9891 ± 0.0076 | 0.9875 ± 0.0062 | 0.9987 ± 0.0011 |
| ElasticNet | 0.9861 ± 0.0099 | 0.9834 ± 0.0093 | 0.9878 ± 0.0081 | 0.9856 ± 0.0011 | 0.9988 ± 0.0157 |
| L1 | 0.9855 ± 0.0063 | 0.9828 ± 0.0075 | 0.9872 ± 0.0095 | 0.9849 ± 0.0066 | 0.9990 ± 0.0009 |
| None | 0.9830 ± 0.0078 | 0.9814 ± 0.0087 | 0.9833 ± 0.0100 | 0.9823 ± 0.0082 | 0.9987 ± 0.0011 |

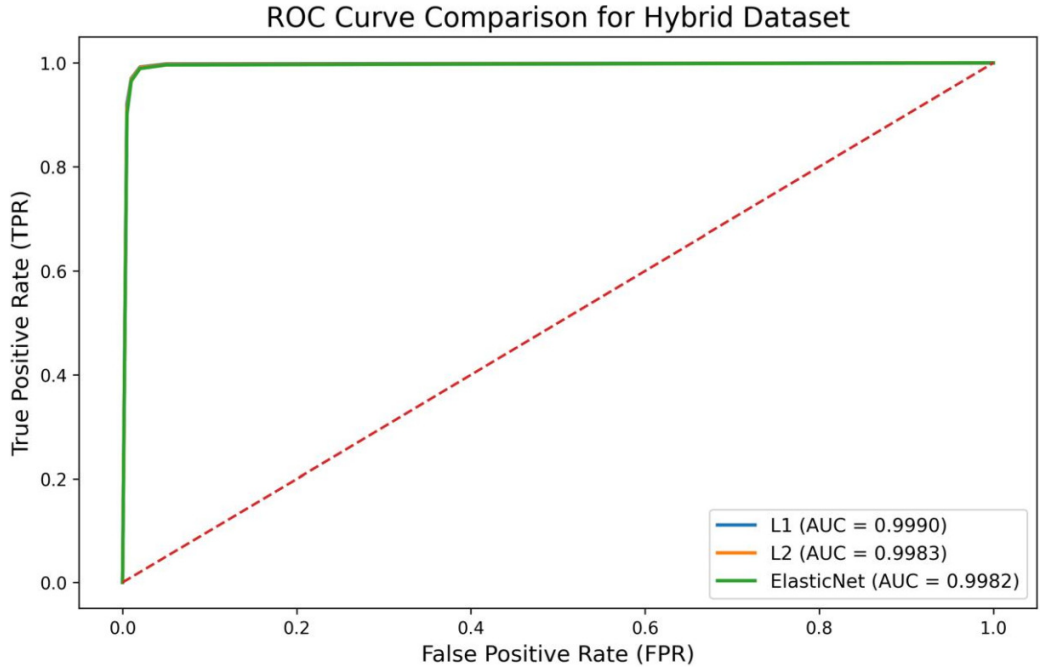


Fig. 2. ROC curve comparison for the hybrid dataset using L1, L2, and ElasticNet.

As presented in Table II, all methods demonstrated high performance, which confirms that the use of hybrid features is effective for Android malware detection. Among the tested methods, L2 achieved slightly more stable results, with an accuracy of 0.988 ± 0.0059 and an F1-score of 0.9875 ± 0.0062. It also achieved the highest MCC, which suggests that the model performed well on both classes. In contrast, ElasticNet and L1 demonstrated similar results; however, these methods underperformed compared to L2. The model without feature selection performed worse in most metrics. Although L1 achieved the highest ROC-AUC value, the difference with the other methods was very small. These results indicate that all methods were able to identify benign and malicious samples effectively.

Figure 2 illustrates the comparison between the ROC curves for the hybrid dataset using L1, L2, and ElasticNet. The ROC curves show that all models achieved very high performance, with AUC values being close to 1. The curves are very similar, which indicates similar performance between L1, L2, and ElasticNet. The strong ROC performance is related to the important features selected by the model, as these features help capture patterns that distinguish between benign and malicious applications.

### *B. Statistical Significance Analysis*

To check whether the differences between the methods are statistically significant, a Wilcoxon signed-rank test was performed. The results outlined in Table III show that all p-values are greater than 0.05. This means that the differences between L1, L2, and ElasticNet are not statistically significant. Although L2 achieved slightly better accuracy and F1-score, this improvement is small and may be due to normal variation in the data rather than a real difference. This confirms that the three methods have similar performance.

TABLE III. STATISTICAL SIGNIFICANCE ANALYSIS USING THE WILCOXON SIGNED-RANK TEST

| Comparison | Statistic | p-value |
|---|---|---|
| ElasticNet vs L1 | 5.0 | 0.375 |
| ElasticNet vs L2 | 7.0 | 0.203 |

## IV. DISCUSSION

The results offer several insights into the behavior of feature selection methods when applied to hybrid Android malware datasets. Although all evaluated configurations achieved high performance, the differences between them are relatively small, indicating that the dataset itself contains strong discriminative patterns.

The L2-based feature selection exhibited a slightly better performance. Unlike L1, which tends to eliminate features aggressively, L2 keeps most of the features while reducing their influence. In the context of Android malware detection, where several features, including permissions, API calls, and

system behaviors, are often correlated, this characteristic appears to be beneficial. Retaining related features, even with reduced weights, may help the model capture more complete patterns of malicious behavior. ElasticNet, which combines both L1 and L2, showed competitive performance but did not outperform L2. A possible explanation is that the dataset does not require strong sparsity, as many features contribute useful information. As a result, removing features too aggressively may lead to a slight loss of information, which can explain the marginal difference compared to L2.

The ensemble model also plays an important role in the model's performance. By combining SVM and RF, the model benefits from different learning mechanisms. SVM focuses on defining optimal decision boundaries, while RF captures non-linear relationships and interactions between features. This combination appears to produce stable and balanced predictions, as reflected in the high MCC values across all configurations. These classifiers enable the framework to leverage complementary learning mechanisms. Furthermore, the relatively low standard deviations show the stability of prediction and the low variability in the evaluation stage, which demonstrates good stability.

Another important finding is the consistently high ROC-AUC values across all methods. This indicates that the models can separate benign and malicious samples effectively over different decision thresholds. However, since these values are very close to each other, they should be interpreted carefully and not used alone to claim superiority of one method over another. The statistical analysis further supports this observation. The Wilcoxon signed-rank test showed that the differences between ElasticNet and both L1 and L2 are not statistically significant. This suggests that the small variations observed in accuracy and F1-score are likely due to a natural variation in the data rather than a meaningful improvement. Therefore, while L2 appears slightly better in practice, it cannot be considered superior from a statistical perspective.

Overall, these findings indicate that the hybrid feature representation is the primary factor driving the strong performance of the model. The role of feature selection, although important, appears to be secondary in this case. This observation aligns with the idea that combining static and dynamic features provides a more complete view of application behavior, which improves detection performance regardless of the specific selection method used.

### *A. Comparative Analysis*

Research shows a strong trend toward using hybrid features (static and dynamic) to improve Android malware detection. For example, DroidMat achieved an accuracy of 97.87% by combining permission analysis with API calls [23], while deep learning models such as TSDNN reported higher accuracy, reaching 99.63% [24]. In addition, it has been found that using data balancing techniques such as SMOTE-Tomek can improve the performance of RF models, reaching an accuracy of 0.993 with high recall [25].

In [23], accuracy exceeding 99% was obtained using feature selection and ensemble models. The present study also achieved high accuracy using hybrid features with L1, L2, and ElasticNet. While there were some small numerical discrepancies between L1, L2, and ElasticNet, the Wilcoxon signed-rank test suggested that these differences were not statistically significant. This indicates that the gains seen could be due to variation in data and not necessarily to a performance benefit. The results demonstrate that the hybrid feature representation is more effective for detecting malware than the regularization method adopted. When comparing the accuracy differences between different approaches for feature selection, the results must be interpreted carefully because they could be influenced by the variability of the dataset.

The experimental results demonstrate that all the regularization-based feature selection methods had relatively similar performance. Somewhat more consistent results were found across several evaluation metrics using L2 regularization. One possible explanation is that the set of features in an Android malware dataset is often highly correlated, and the permissions and API calls that result in malware execution have similar characteristics. While L1 tends to aggressively remove features, L2 preserves correlated information and is less likely to affect the classification behavior (as Feature weights), resulting in more stable behavior.

These trends align with recent experimental results that recorded an accuracy of 0.9880 when applying feature selection (L2) to hybrid features. Furthermore, the Wilcoxon signed-rank test confirms that superior performance is primarily based on the quality and comprehensiveness of the extracted hybrid features rather than the feature selection algorithm itself.

### *B. Feature Interpretation*

Figure 3 shows the top ten features selected by the L2 method on the hybrid dataset. Most of these features are related to system calls, permissions, and API usage, which are often linked to malicious behavior in Android applications. The good performance of the hybrid dataset suggests that the fused dataset can better represent the behavior of Android applications by integrating with both static and dynamic features. Static analysis is able to learn of structural data like permissions or API calls, and dynamic analysis can learn of activities and patterns at runtime. Combining both types of features eliminates the restrictions of using a single type of analysis. This indicates that having diverse features contributes to significant Android malware detection performance improvements.

These features appear across different tests, meaning that they are stable and not selected by chance. This shows that the model is learning real patterns from the data.

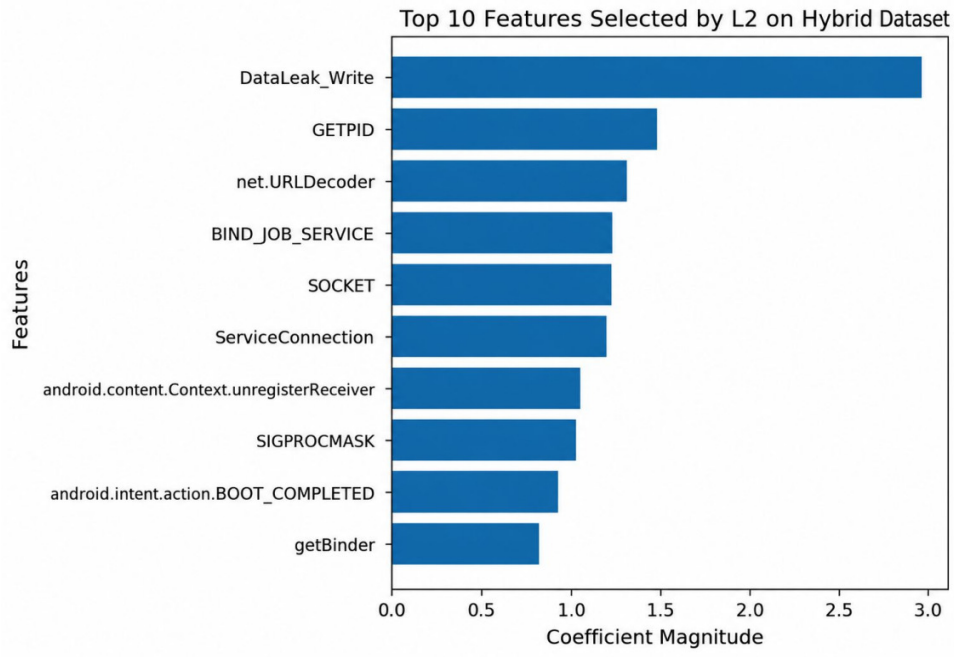


Fig. 3. Top ten features selected by L2 on the hybrid dataset.

Identification of the most important features helps understand how the model classify if an application is malicious or not. Selected features are primarily related to permission granted, API invocations, and system activity that are often linked to malicious activity on Android. These characteristics can be suspicious behavior, such as improper access to sensitive information, background execution of activities, and irregular communication behavior. The presence of these features across different validation folds demonstrates their consistency and informative nature regarding behavioral patterns, while bypassing randomness. The association of well-known malware features further adds to the trustworthiness of the proposed structure.

### *C. Limitations*

The obtained results are specific to the datasets and experimental setup used in this study. Therefore, the conclusions drawn should not be generalized without further validation on different datasets. Future work may explore more diverse datasets and investigate the use of deep learning models to further improve generalization.

There are a number of variables that could impact the validity of the reported results. The dataset used in the study may not contain all label inconsistencies that could be detected in the pre-processing stage, or there could be sampling bias that cannot be eliminated in the pre-processing stage. Also, the results of this publication have not been validated on independent datasets that enable the generalization beyond the assessed experimental configuration. In addition, the current study is based on publicly available data; thus, the complexity of the current Android malware ecosystems might not be captured. Future work should address these limitations with larger and more diverse datasets.

Even though the model achieved strong performance, there are a few dataset-related limitations that need to be acknowledged. The dataset used in this study does not provide detailed insights into possible label noise. In real-world settings, some applications might be mislabeled as benign or malicious, and this can influence how the model learns and slightly affect the reported results. While no clear inconsistencies were found during the preprocessing stage, the presence of labeling errors cannot be fully ruled out. Moreover, the dataset did not include information about malware families or any temporal structure. Therefore, it was not possible to apply family-based or time-based splitting strategies. For this reason, the evaluation was carried out using random cross-validation. This setup does not fully represent real-world conditions, where models are usually evaluated on newer or previously unseen malware samples.

These limitations should be considered when interpreting the findings. Using more recent datasets with richer annotations in future work could provide a better understanding of how well the proposed model generalizes in more realistic scenarios.

## V. CONCLUSION

This study proposed a hybrid model for Android malware detection by combining static and dynamic features with feature selection and ensemble learning. The proposed framework was evaluated using stratified 10-fold cross-validation on a hybrid dataset comprising 3,239 Android applications and 675 combined static and dynamic features. Among the evaluated methods, L2 achieved the highest accuracy (0.9880 ± 0.0059), F1-score (0.9875 ± 0.0062), and MCC (0.9759 ± 0.0119).

The results demonstrated that hybrid features achieved comparatively consistent performance across the evaluated metrics. In addition, the ensemble model Support Vector Machine-Random Forest (SVM-RF) provided stable classification performance across validation folds. It also helped to improve prediction stability and maintain a balance between classes. In real cybersecurity scenarios, minimizing false classifications and ensuring consistent detection behavior are important for the security of mobile app analysis. While statistical testing demonstrated no statistically significant difference between the various methods applied during feature selection, the results indicate that the performance of the proposed framework is influenced by the adopted framework of the hybrid features rather than by a particular regularization approach. Overall, the findings indicate that combining different features is more important than relying on a specific feature selection method. Despite these limitations, the proposed framework may help improve the effectiveness of malware detection.

However, the study has a few limitations due to the limited dataset and evaluation protocols. The lack of temporal and malware family data in the dataset reduces the capacity of the proposed framework, especially in a more realistic deployment environment with new variants of previously unknown malware. For future work, it is proposed to test the model on recent datasets and explore deep learning approaches to further improve performance. AI explainability methods, the usage of more extensive datasets across platforms, and evaluation methods considering time can be explored to enhance generalization and interpretability. Overall, the findings of this study should be taken as proof of performance for the dataset and experimental protocol, and not as proof of overall superiority.

## DECLARATION OF COMPETING INTERESTS

The authors declare no competing interests.

## ACKNOWLEDGMENT

Not applicable to this work.

## DATA AVAILABILITY

The datasets used in this study were collected from [12] and [13].

## AI USE AND DECLARATION OF GENERATIVE AI USE

The authors declare that no generative AI tools were used for the purpose of content generation. However, an AI-assisted language editing tool was used for grammar and readability improvement.

## REFERENCES

[1] A. Rawat, A. Kumar, A. K. Singh, and K. Arora, "Exploring Android Security Landscape: Threats, Vulnerabilities, and Best Practices," *International Research Journal on Advanced Engineering and Management (IRJAEM)*, vol. 2, no. 06, pp. 1831–1839, Jun. 2024, https://doi.org/10.47392/IRJAEM.2024.0271.

[2] C. Tumbleson, "APKtool." Feb. 2023, [Online]. Available: https://github.com/iBotPeaches/Apktool.

[3] E. Amer and S. El-Sappagh, "Robust Deep Learning Early Alarm Prediction Model Based on the Behavioural Smell for Android Malware," *Computers & Security*, vol. 116, May 2022, Art. no. 102670, https://doi.org/10.1016/j.cose.2022.102670.

[4] G. Pitolli, G. Laurenza, L. Aniello, L. Querzoni, and R. Baldoni, "MalFamAware: Automatic Family Identification and Malware Classification Through Online Clustering," *International Journal of Information Security*, vol. 20, no. 3, pp. 371–386, Jun. 2021, https://doi.org/10.1007/s10207-020-00509-4.

[5] A. Maryam, U. Ahmed, M. Aleem, J. C.-W. Lin, M. Arshad Islam, and M. A. Iqbal, "cHybriDroid: A Machine Learning-Based Hybrid Technique for Securing the Edge Computing," *Security and Communication Networks*, vol. 2020, pp. 1–14, Nov. 2020, https://doi.org/10.1155/2020/8861639.

[6] R. A. M. Alsaidi, W. M. S. Yafooz, H. Alolofi, G. A.-M. Taufiq-Hail, A.-H. M. Emara, and A. Abdel-Wahab, "Ransomware Detection Using Machine and Deep Learning Approaches," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 11, 2022, https://doi.org/10.14569/IJACSA.2022.0131112.

[7] P. Bhat and K. Dutta, "A Multi-Tiered Feature Selection Model for Android Malware Detection Based on Feature Discrimination and Information Gain," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, pp. 9464–9477, Nov. 2022, https://doi.org/10.1016/j.jksuci.2021.11.004.

[8] 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, Nov. 2021, Art. no. 2813, https://doi.org/10.3390/math9212813.

[9] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in *Proceedings 2014 Network and Distributed System Security Symposium*, San Diego, CA, USA, 2014, https://doi.org/10.14722/ndss.2014.23247.

[10] N. Xie, Z. Qin, and X. Di, "GA-StackingMD: Android Malware Detection Method Based on Genetic Algorithm Optimized Stacking," *Applied Sciences*, vol. 13, no. 4, Feb. 2023, Art. no. 2629, https://doi.org/10.3390/app13042629.

[11] S. Sharma, P. R. Chhikara, and K. Khanna, "An Efficient Android Malware Detection Method Using Borutashap Algorithm," *International Journal of Experimental Research and Review*, vol. 34, pp. 86–96, Oct. 2023, https://doi.org/10.52756/ijerr.2023.v34spl.009.

[12] A. R. Zaidi, T. Abbas, H. Zahid, and S. A. Ramay, "Effectiveness of Detecting Android Malware Using Deep Learning Techniques," *Journal of Nanoscope (JN)*, vol. 4, no. 2, pp. 1–21, Nov. 2023, https://doi.org/10.52700/jn.v4i2.90.

[13] M. Dhalaria and E. Gandotra, "Android Malware Detection and Classification." Kaggle, 2021, [Online]. Available: https://www.kaggle.com/datasets/meghnadhalaria/android-malware-detection-and-classification.

[14] M. Dhalaria, "Android Malware Dataset Consisting of Static Features." GitHub, 2019, [Online]. Available: https://github.com/Meghna-Dhalaria/Android-malware-dataset.

[15] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, Jan. 1996, https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

[16] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, Feb. 1970, https://doi.org/10.1080/00401706.1970.10488634.

[17] H. Shah, V. Shah, N. Soni, V. Vadhavana, and K. Patel, "A Comparative Analysis for Android Malware Detection Using Machine Learning Models," in *6th International Conference on Mobile Computing and Sustainable Informatics*, Goathgaun, Nepal, Jan. 2025, pp. 1040–1047, https://doi.org/10.1109/ICMCSI64620.2025.10883385.

[18] A. Museeb, Y. Hamed, Y. Baashar, A. M. Jamal Kanaan-Jebna, A. Amazigh Hamza, and R. Sokkalingam, "Android Malware Detection Using API Calls and Permissions With Random Forest Classifier," *IEEE Access*, vol. 14, pp. 6464–6480, 2026, https://doi.org/10.1109/ACCESS.2026.3651861.

[19] W. M. S. Yafooz, "Enhancing Business Intelligence with Hybrid Transformers and Automated Annotation for Arabic Sentiment Analysis," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 8, 2024, https://doi.org/10.14569/IJACSA.2024.0150821.

[20] V. Kouliaridis, K. Barmpatsalou, G. Kambourakis, and S. Chen, "A Survey on Mobile Malware Detection Techniques," *IEICE Transactions on Information and Systems*, vol. E103.D, no. 2, pp. 204–211, Feb. 2020, https://doi.org/10.1587/transinf.2019INI0003.

[21] M. A. Haq and M. Khuthaylah, "Leveraging Machine Learning for Android Malware Analysis: Insights from Static and Dynamic Techniques," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15027–15032, Aug. 2024, https://doi.org/10.48084/etasr.7632.

[22] W. M. S. Yafooz, A. Alsaeedi, R. Alluhaibi, and A.-H. Mohamed Emara, "Enhancing Multi-Class Web Video Categorization Model Using Machine and Deep Learning Approaches," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 3, Jun. 2022, Art. no. 3176, https://doi.org/10.11591/ijece.v12i3.pp3176-3191.

[23] E. J. Alqahtani, R. Zagrouba, and A. Almuhaideb, "A Survey on Android Malware Detection Techniques Using Machine Learning Algorithms," in *Sixth International Conference on Software Defined Systems*, Rome, Italy, Jun. 2019, pp. 110–117, https://doi.org/10.1109/SDS.2019.8768729.

[24] M. S. Masari, M. A. Danladi, I. L. Onyinye, and L. K. Tohomdet, "Android Malware Detection Using Machine Learning with SMOTE-Tomek Data Balancing," *Journal of Computing Theories and Applications*, vol. 3, no. 3, pp. 302–313, Jan. 2026, https://doi.org/10.62411/jcta.15084.

[25] Md. A. Hossain and Md. S. Islam, "Enhanced Detection of Obfuscated Malware in Memory Dumps: A Machine Learning Approach for Advanced Cybersecurity," *Cybersecurity*, vol. 7, no. 1, Jan. 2024, Art. no. 16, https://doi.org/10.1186/s42400-024-00205-z.