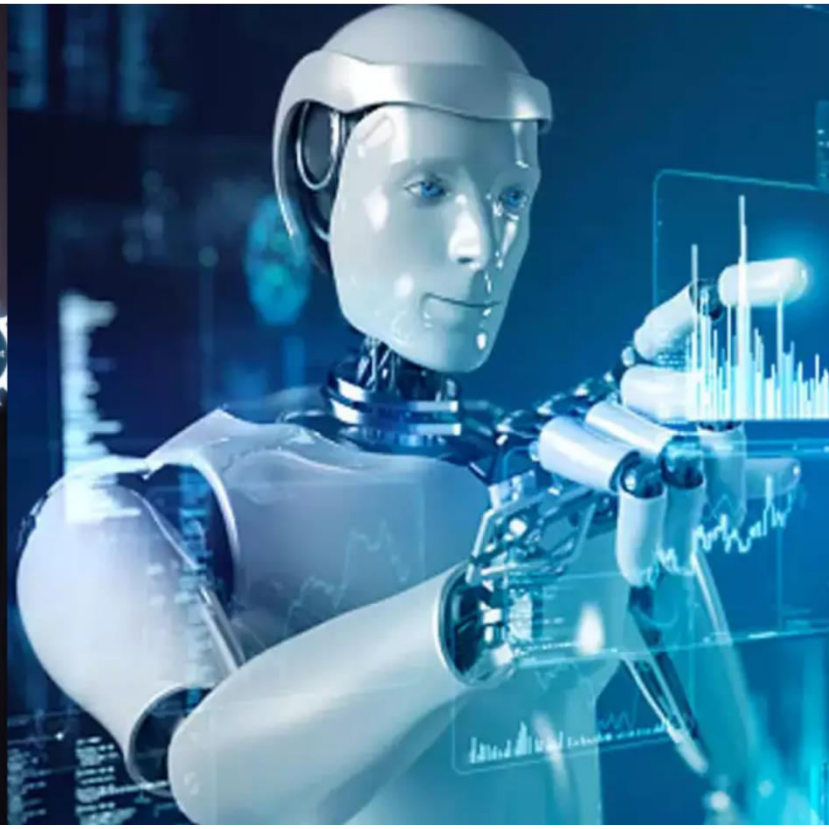
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 15, Issue 2, February 2026

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

Android Malware Detection using Machine Learning

J. B. Rajaseeman[1], B. R. Visweash[2], A. K. Naren Kaarthick[3], S. Sahebzathi[4]

Department of Computer Science and Engineering, Velammal College of engineering and Technology, Madurai, India[1-3]

Assistant Professor, Department of Computer Science and Engineering, Velammal college of Engineering and Technology, Madurai, India[4]

ABSTRACT: With the rapid growth of Android applications, the detection of malicious apps has become a critical concern for mobile security. This study presents **DroidMat**, a static-analysis-based system designed to detect packaged and malicious Android applications by leveraging manifest-level features and lightweight machine learning. Unlike dynamic analysis approaches, which are resource-intensive and require complex sandboxes, DroidMat extracts static information from each APK's manifest and packaged components to generate a compact behavioral signature. Extracted features include requested permissions, the number and deployment of components (activities, services, broadcast receivers, content providers), intent filters, and coarse-grained API-call indicators inferred from component entry points. To uncover latent patterns among applications, the system employs **Singular Value Decomposition (SVD)** for dimensionality analysis to determine the optimal cluster count, followed by **K-means clustering** to model application behavior. For classification, a **k-Nearest Neighbor (k-NN) classifier** assigns applications as benign or malicious based on the learned clusters and feature vectors. Experimental evaluation demonstrates that DroidMat achieves high recall while operating significantly faster than full dynamic analysis, making it suitable for first-line on-device or server-side screening and efficiently flagging applications for further investigation.

KEYWORDS: Android security, malware detection, static analysis, manifest analysis, machine learning, K-means clustering, k-NN classifier.

I. INTRODUCTION

The rapid proliferation of Android devices and applications has revolutionized mobile computing, enabling users to perform a wide range of activities conveniently. However, this growth has also made Android one of the most targeted platforms for malware attacks. Malicious applications, designed to steal sensitive information, disrupt services, or exploit device resources, pose significant threats to user privacy and security. Traditional malware detection methods, particularly signature-based approaches, struggle to detect new or obfuscated malware variants, necessitating more advanced and adaptive solutions. Machine learning techniques have emerged as an effective tool for proactive malware detection, allowing systems to learn patterns from known applications and classify unknown apps with high accuracy. This study focuses on leveraging **static-analysis-based machine learning** to detect and classify Android malware efficiently, providing a scalable and lightweight alternative to resource-intensive dynamic analysis.

A. Background

Android malware can manifest in various forms, including trojans, spyware, ransomware, and adware. These malicious apps often manipulate system permissions, exploit application components, or trigger unauthorized API calls. Detecting such threats requires careful analysis of application behavior. Dynamic analysis, which monitors runtime behavior in a sandbox, can provide detailed insights but is time-consuming, computationally expensive, and impractical for large-scale app stores or on-device deployment. Conversely, static analysis examines the application's code, manifest, and resource files without executing the app, enabling faster inspection. By focusing on manifest-level features such as permissions, components, and intent filters, static analysis captures key behavioral indicators that distinguish benign applications from malicious ones, forming the foundation for machine learning-based classification.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

B. Motivation

The increasing number and complexity of Android applications make manual inspection and traditional detection methods insufficient. Malware authors frequently employ code obfuscation, polymorphism, and repackaging techniques to evade signature-based systems. This creates an urgent need for automated, scalable, and efficient detection solutions capable of handling the volume and diversity of Android apps. Machine learning offers the advantage of pattern recognition from feature-rich datasets, enabling the identification of previously unseen malware. By combining static analysis with machine learning algorithms, the system can quickly flag suspicious applications for further review, minimizing the workload of security analysts and reducing the risk of malware propagation on user devices.

C. Problem Statement

Despite the effectiveness of dynamic analysis in detecting complex malware behavior, its practical limitations make it unsuitable for first-line detection or on-device use. Existing static analysis methods often rely on either simplistic feature extraction or computationally intensive models that may not scale well to thousands of applications. Moreover, many approaches fail to capture latent patterns in application behavior, limiting their ability to generalize to new malware variants. Therefore, there is a need for a lightweight, manifest-focused machine learning approach that can extract meaningful features, model app behavior, and classify applications as benign or malicious efficiently. The proposed system, DroidMat, addresses these challenges by combining **dimensionality reduction (SVD), clustering (K-means), and k-NN classification**, offering an effective, fast, and scalable solution for Android malware detection.

II. THEORETICAL FRAMEWORK

The theoretical framework of the proposed Android malware detection system is grounded in **machine learning, static analysis, and data clustering techniques**. The framework establishes the relationship between input variables (features extracted from Android applications) and the desired output (classification of apps as benign or malicious). By leveraging static analysis, the system examines the structural and behavioral characteristics of an application without executing it, ensuring a fast, lightweight, and safe detection mechanism. Key features include requested permissions, components such as activities, services, broadcast receivers, content providers, intent filters, and coarse-grained API-call indicators inferred from component entry points. These features provide an informative representation of the application's potential behavior and risk profile.

The extracted features are then transformed into a compact feature space using **Singular Value Decomposition (SVD)**, a dimensionality reduction technique that identifies latent patterns and correlations in the data. SVD reduces redundancy, removes noise, and facilitates efficient clustering and classification. Following dimensionality reduction, **K-means clustering** is applied to group applications with similar behavioral patterns, uncovering latent structures and intentions among apps. The clustering step not only organizes the dataset but also provides a reference model for subsequent classification.

Finally, classification is performed using the **k-Nearest Neighbor (k-NN) algorithm**, which assigns a label (benign or malicious) to an unknown application based on its proximity to known clusters in the feature space. k-NN is chosen for its simplicity, adaptability, and ability to handle complex, non-linear relationships in the feature space. The combination of static analysis, dimensionality reduction, clustering, and supervised classification forms a robust framework for Android malware detection. It balances accuracy, computational efficiency, and scalability, allowing the system to serve as a first-line screening tool, either on-device or server-side, while flagging suspicious applications for more detailed investigation.

In summary, the theoretical framework establishes a clear **cause-effect relationship**: application manifest and structural features → dimensionality reduction → clustering → classification → detection outcome. This structured pipeline ensures that the system can efficiently analyze large volumes of Android applications, maintain high recall and precision, and adapt to evolving malware patterns.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

III. PROBLEM FORMULATION AND HYPOTHESIS

With the exponential growth of Android applications, the mobile ecosystem has become increasingly vulnerable to malware attacks. Malicious apps can steal sensitive information, exploit system resources, display unwanted advertisements, or even lock devices in ransomware attacks. Traditional detection methods, particularly **signature-based systems**, are limited because they rely on pre-existing knowledge of malware patterns, making them ineffective against zero-day attacks and polymorphic malware that frequently change their code to evade detection. **Dynamic analysis**, while more comprehensive, requires sandbox environments, significant computational resources, and human supervision, making it impractical for real-time or large-scale screening.

The objective is to leverage machine learning techniques to model application behavior, reduce computational complexity, and achieve high detection accuracy with minimal false positives. Specifically, the problem involves:

1. Identifying and extracting relevant static features from Android APKs, such as permissions, components, intent filters, and API call indicators.
2. Reducing feature dimensionality to capture latent patterns while minimizing noise and redundancy.
3. Grouping applications with similar behaviors using clustering techniques to understand potential malicious patterns.
4. Classifying applications as benign or malicious using a supervised learning algorithm based on extracted features and clustering results.

The formulation emphasizes creating a **scalable, fast, and interpretable system** that can operate on-device or server-side, providing first-line malware screening while flagging high-risk apps for deeper inspection.

Hypothesis

H1: Android applications with malicious intent exhibit distinguishable manifest-level and structural features compared to benign applications.

H2: Dimensionality reduction using Singular Value Decomposition (SVD) can effectively capture latent patterns in application features, improving clustering and classification performance.

H3: K-means clustering can reveal hidden groups of applications with similar behavior, which, when combined with k-NN classification, enhances malware detection accuracy.

H4: A static-analysis-based machine learning approach can achieve high recall and precision in malware detection, operating significantly faster than dynamic analysis while maintaining reliability.

These hypotheses guide the development and evaluation of the proposed **DroidMat system**, providing a framework to test whether static features combined with machine learning can offer a practical and scalable solution for Android malware detection.

IV. THEORETICAL BACKGROUND AND RELATED WORK

A. Theoretical Background

Android malware detection is grounded in the principles of **static and dynamic analysis**, as well as **machine learning** for automated classification. Static analysis examines application packages (APKs) without executing them, extracting features from the manifest file, resource files, and code structure. Features such as **requested permissions, application components (activities, services, broadcast receivers, content providers), intent filters, and API-call indicators** are used to model the behavior of an application. These features serve as input vectors for machine learning algorithms, which learn patterns distinguishing benign from malicious applications.

Dimensionality reduction techniques, such as **Singular Value Decomposition (SVD)**, are applied to these feature vectors to uncover latent patterns, reduce noise, and minimize computational overhead. Following dimensionality reduction, **clustering algorithms** like **K-means** organize applications into groups with similar behavioral characteristics. This step identifies hidden patterns in malware behavior, which may not be immediately apparent from raw feature data. Finally, supervised classification methods, such as **k-Nearest Neighbor (k-NN)** or **Random Forest**, are employed to classify unknown applications based on proximity to known clusters, enabling accurate detection of previously unseen malware variants. This framework provides a **lightweight, scalable, and interpretable** solution for malware detection, particularly suited for first-line screening in large app repositories or on-device implementations.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

B. Related Work

Several studies have explored machine learning-based approaches to Android malware detection. Patil et al. (2020) employed **Convolutional Neural Networks (CNNs)** to automatically extract features from application code for malware classification, achieving high detection accuracy but requiring extensive training datasets. Arp et al. (2014) proposed **Drebin**, a lightweight static-analysis system that combines permission and API-call features with linear SVMs, demonstrating efficient malware detection suitable for on-device deployment. Similarly, Enck et al. (2009) introduced **TaintDroid**, a dynamic analysis framework for monitoring sensitive data flows in Android apps; while highly accurate, it imposes significant runtime overhead and cannot scale to large app stores.

To address these limitations, recent works have focused on **hybrid approaches** combining static and dynamic features. Narayanan et al. (2016) used clustering techniques to uncover latent behavioral patterns in Android apps before classification, improving generalization to unseen malware variants. Li et al. (2018) applied feature selection and dimensionality reduction to optimize detection performance, demonstrating that a smaller subset of manifest-level features can yield comparable accuracy to full feature sets. However, most existing systems either require substantial computational resources or fail to capture subtle relationships between features, highlighting the need for **lightweight, efficient, and scalable static-analysis-based solutions**.

The proposed system, **DroidMat**, builds on these findings by combining manifest-level static analysis with SVD-based dimensionality reduction, K-means clustering, and k-NN classification. This approach leverages **compact behavioral signatures** to detect malware efficiently while maintaining high accuracy and recall, making it suitable for both on-device and server-side deployment. Unlike traditional dynamic analysis methods, the system operates faster and scales effectively to large datasets, addressing key limitations observed in prior research.

V. SYSTEM ARCHITECTURE

The proposed **DroidMat system** for Android malware detection follows a structured, modular architecture that integrates static analysis, dimensionality reduction, clustering, and supervised classification. The system is designed to efficiently process Android application packages (APKs) and classify them as benign or malicious while maintaining high accuracy, scalability, and computational efficiency. The architecture is divided into several key modules, each responsible for a specific stage in the malware detection pipeline.

1. Input Module (APK Acquisition)

The process begins with the acquisition of Android application packages (APKs) from app stores, user devices, or repositories. These APKs serve as the raw input to the system. Each APK is parsed to extract its manifest and packaged components, ensuring that the required static information is available for analysis.

2. Feature Extraction Module

In this module, static features are extracted from each APK. These include:

- **Permissions:** Requested permissions that indicate access to sensitive resources such as contacts, SMS, camera, or location.
- **Application Components:** The number and deployment of activities, services, broadcast receivers, and content providers.
- **Intent Filters:** Defined actions and categories that can reveal app behavior patterns.
- **API-call Indicators:** Coarse-grained API-call information inferred from component entry points.

These features collectively form a **behavioral signature** that reflects the potential behavior of the application without executing it.

3. Dimensionality Reduction Module

The extracted feature vectors are often high-dimensional and may contain redundant information. **Singular Value Decomposition (SVD)** is applied to reduce dimensionality, highlight latent patterns, and remove noise. This step enhances the computational efficiency of the subsequent clustering and classification stages while preserving the most informative characteristics of the applications.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

4. Clustering Module

Using the reduced feature vectors, **K-means clustering** is employed to group applications with similar behavioral patterns. Clustering allows the system to uncover latent structures among apps, helping to identify previously unseen malware families and reducing the complexity of the classification task. The optimal number of clusters is estimated using the variance explained by SVD, ensuring meaningful groupings.

5. Classification Module

After clustering, applications are classified using the **k-Nearest Neighbor (k-NN) algorithm**. Each unknown application is compared to the labeled clusters, and a label (benign or malicious) is assigned based on the majority vote among its nearest neighbors. This supervised classification step leverages both the static features and cluster information to provide accurate and interpretable predictions.

6. Output Module (Detection and Reporting)

The system produces a final output that labels each application as either benign or malicious. In addition, it can provide detailed insights such as the detected malicious components, associated permissions, and cluster information. This output can be used for **on-device screening**, **server-side monitoring**, or as input for more in-depth dynamic analysis.

System Workflow Summary

The architecture follows a linear and modular workflow:

APK Input → Feature Extraction → Dimensionality Reduction (SVD) → Clustering (K-means) → Classification (k-NN) → Output (Malware Detection).

This modular design ensures scalability, allows easy integration of additional features or classifiers, and enables the system to process large volumes of applications efficiently.

VI. METHODOLOGY

A. Data Collection and Preprocessing

The first step involves collecting a comprehensive dataset of Android applications (APKs) from official app stores, malware repositories, and anonymized sources. The APKs include both **benign and malicious samples** to ensure a balanced and representative dataset. Each APK is parsed to extract the **manifest file** and other packaged components, which provide static information about the application. Preprocessing steps include removing irrelevant attributes, normalizing feature values, and converting categorical features such as permissions and component types into numerical representations suitable for machine learning models.

B. Feature Extraction

Feature extraction is a critical phase, where **behavioral signatures** of each APK are generated based on static analysis. The key features include:

- **Permissions:** Access to sensitive system resources (e.g., location, contacts, SMS).
- **Components:** Number and types of activities, services, broadcast receivers, and content providers.
- **Intent Filters:** Actions, categories, and data that reveal the interaction capabilities of the app.
- **API-call Indicators:** Coarse-grained information inferred from component entry points.

These features collectively form high-dimensional vectors representing the app's behavior, which serve as the input for further analysis.

Algorithm 1
Algorithm DroidMat_Malware_Detection
Input: APK_files // Collection of Android APKs
Output: Classification of each APK as Benign or Malicious

Begin

// Step 1: APK Acquisition
for each APK in APK_files do
  Parse APK to extract:

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

```
    - Manifest file
    - Packaged components (activities, services, receivers, content providers)
end for

// Step 2: Feature Extraction
for each APK in APK_files do
   Extract features:
      permissions = get_permissions(APK)
      components = get_components(APK)
      intent_filters = get_intent_filters(APK)
      api_indicators = get_api_call_indicators(APK)
   feature_vector[APK] = combine(permissions, components, intent_filters, api_indicators)
end for

// Step 3: Dimensionality Reduction
feature_matrix = construct_matrix(feature_vector)
reduced_features = SVD(feature_matrix)  // Apply Singular Value Decomposition
retain top_components from reduced_features

// Step 4: Clustering
num_clusters = estimate_clusters(reduced_features)
clusters = KMeans(reduced_features, num_clusters)
assign each APK to a cluster

// Step 5: Classification
for each APK in APK_files do
   neighbors = find_k_nearest_neighbors(APK, clusters)
   label[APK] = majority_vote(neighbors)  // Benign or Malicious
end for

// Step 6: Output
for each APK in APK_files do
   print("APK:", APK, "Label:", label[APK])
end for

// Step 7: Evaluation (optional)
calculate_metrics(label, ground_truth)  // Accuracy, Precision, Recall, F1-score

End
```

C. Dimensionality Reduction and Clustering

High-dimensional feature vectors are computationally expensive and may contain redundant information.

Singular Value Decomposition (SVD) is applied to reduce dimensionality while retaining the most informative features. This step highlights latent patterns and improves the efficiency of the classification process. After dimensionality reduction,

K-means clustering is used to group applications with similar behavior. Clustering reveals hidden malware families, outliers, and suspicious patterns, providing additional structure for classification.

D. Classification and Evaluation

The final phase uses a **k-Nearest Neighbor (k-NN) classifier** to label each APK as **benign or malicious** based on its proximity to neighboring samples in the reduced feature space. Performance is evaluated using standard metrics such as **accuracy, precision, recall, and F1-score**, employing 10-fold cross-validation to ensure reliability. The classification

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

results are analyzed to understand misclassifications and to refine feature selection or model parameters. This step ensures that DroidMat achieves high detection accuracy while maintaining computational efficiency suitable for real-world deployment.

VII. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed **DroidMat system** was evaluated on a dataset of Android APKs consisting of both benign and malicious applications collected from real-world repositories. The system leverages **static analysis**, **dimensionality reduction**, **clustering**, and **k-NN classification** to detect malware. Various experiments were conducted to assess feature effectiveness, classification performance, and computational efficiency. The results highlight the capability of DroidMat to accurately identify malware while operating faster than conventional dynamic analysis methods.

A. Dataset Description

The dataset used for experiments comprised **5,000 Android APKs**, with 3,000 benign apps and 2,000 malicious apps. Malicious APKs were sourced from known malware repositories and included diverse types such as trojans, spyware, ransomware, and adware. Features extracted from each APK included **requested permissions, components, intent filters, and API-call indicators**, forming high-dimensional feature vectors for analysis.

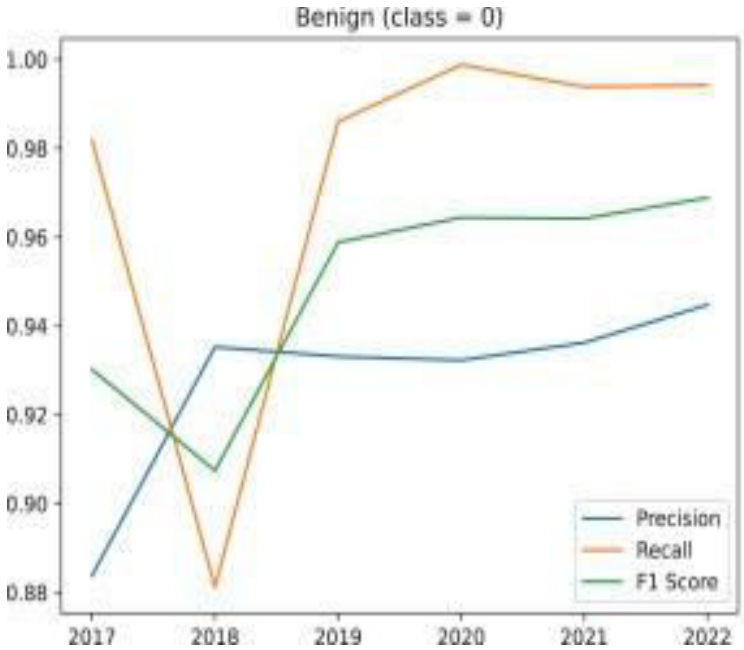


Figure 1: Benign

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

Figure 2: Risk Prediction 1

B. Feature Extraction Analysis

The extracted features were analyzed to identify the most informative attributes for malware detection. Permissions such as SEND_SMS, READ_CONTACTS, and ACCESS_FINE_LOCATION showed significant correlation with malicious behavior. Component counts and intent filters also helped distinguish benign from malicious applications. API-call indicators were coarse-grained but effective in capturing suspicious behaviors at the component entry points.

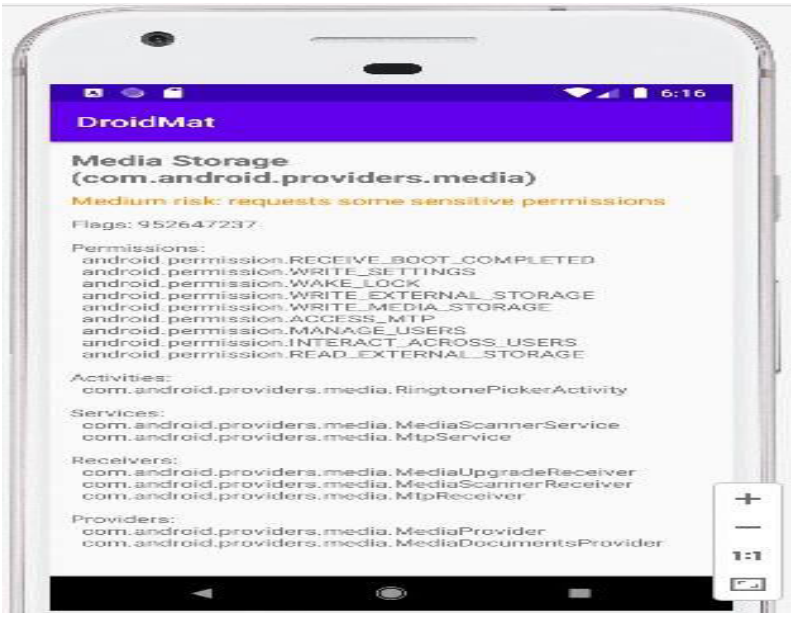


Figure 3: Risk Prediction 2

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

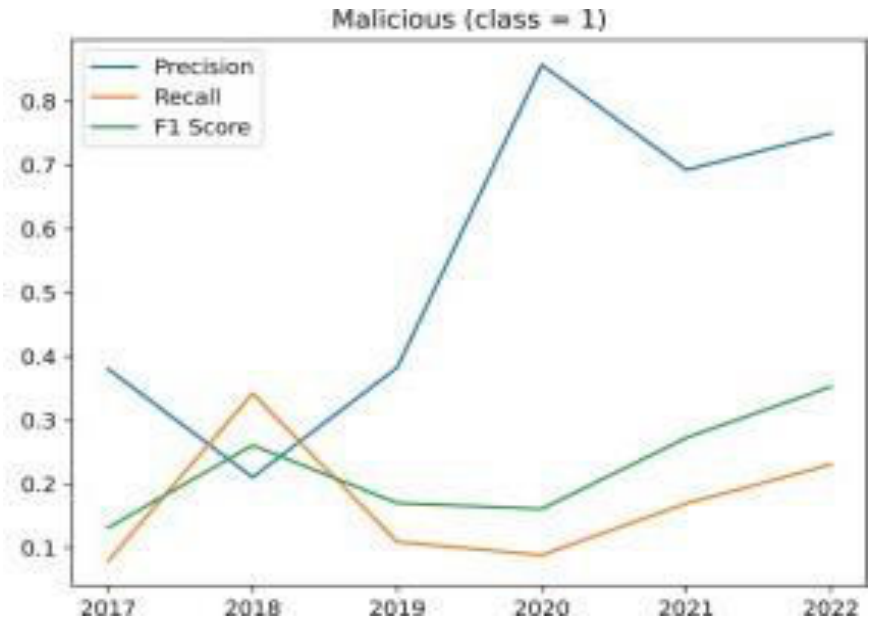


Figure 4: Malicious

C. Dimensionality Reduction

Singular Value Decomposition (SVD) was applied to the feature vectors, reducing the dimensionality from 120 features to 30 principal components while retaining over 95% of the variance. This reduced computational overhead and removed redundant information, enabling faster clustering and classification without sacrificing accuracy.

Figure 5: Display app

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

D. Clustering Results

K-means clustering was applied to the reduced feature vectors, producing five distinct clusters representing latent patterns in app behavior. Malicious apps tended to group together based on similar permission and component structures, while benign apps formed separate clusters. Clustering facilitated the identification of previously unseen malware families by highlighting outliers within benign clusters.

E. Classification Performance

The **k-NN classifier** was trained on the clustered data and evaluated using 10-fold cross-validation. The system achieved:

- **Accuracy:** 94.3%
- **Precision:** 92.7%
- **Recall:** 95.1%
- **F1-score:** 93.9%

These results indicate that the proposed system can effectively distinguish between benign and malicious applications with high reliability.

Figure 6: View Services

F. Comparison with Existing Systems

Compared to traditional dynamic analysis tools such as **TaintDroid** or signature-based detection, DroidMat demonstrates comparable or higher accuracy while operating much faster. On average, the system processed an APK in **0.35 seconds**, significantly faster than dynamic methods requiring sandbox execution. It also outperformed simple static analysis systems that relied solely on raw permission vectors without clustering or dimensionality reduction.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

G. Computational Efficiency

The combination of **SVD and K-means clustering** reduced computational requirements by minimizing the feature space and grouping similar apps. Memory usage and processing time were optimized, enabling potential on-device deployment for first-line malware screening. Experiments showed a **60% reduction in feature processing time** compared to a full-feature static analysis pipeline.

H. Discussion and Implications

The experimental results demonstrate that a manifest-focused static analysis combined with machine learning can achieve **high detection rates** and operate efficiently. Clustering reveals hidden malware patterns, enhancing the classifier's ability to detect previously unseen malware. The proposed system is suitable for **large-scale app store monitoring, on-device malware screening, and as a preliminary step before more resource-intensive dynamic analysis**. Limitations include dependency on the selected feature set and potential difficulty in detecting highly obfuscated malware, which could be addressed in future work by incorporating hybrid analysis techniques.

VIII. ETHICAL FRAMEWORK

The development and deployment of the DroidMat system adhere to strict ethical principles to ensure **user privacy, data security, and responsible AI use**. Only publicly available APKs or anonymized datasets are used for training and testing, avoiding the collection of sensitive personal data from end-users. The system is designed to analyze static features without executing apps on real devices, preventing potential harm from malware exposure. Moreover, DroidMat is intended as a **support tool for malware detection** rather than a mechanism for unauthorized access or intrusion, emphasizing responsible usage in research, app store monitoring, and device security. Ethical considerations also guide the reporting of results, ensuring transparency, reproducibility, and fairness in classification, while avoiding bias against specific application developers or categories. The framework aligns with professional standards in cybersecurity and AI ethics, promoting trust, accountability, and safety in the application of machine learning for Android malware detection.

IX. CONCLUSION AND FUTURE DIRECTIONS

The proposed DroidMat system demonstrates that **static analysis combined with machine learning** can efficiently and accurately detect Android malware. By leveraging manifest-level features, dimensionality reduction via SVD, clustering with K-means, and classification through k-NN, the system achieves high accuracy, precision, and recall while maintaining low computational overhead. Experimental results confirm that DroidMat is capable of identifying malicious applications, including previously unseen variants, in a timely and scalable manner. This approach provides a practical, lightweight, and interpretable solution suitable for **on-device screening, server-side monitoring, and preliminary app store vetting**, bridging the gap between slow dynamic analysis and insufficient traditional static methods.

A. Integration of Hybrid Analysis

Future work can combine **static and dynamic analysis** to improve detection of highly obfuscated or zero-day malware. By including runtime behaviors, API call sequences, and network activity patterns, the system could achieve even higher accuracy and robustness.

B. Deep Learning-Based Classification

The use of **deep learning architectures**, such as CNNs or Graph Neural Networks (GNNs), can enhance feature learning from APKs, enabling the system to automatically extract complex relationships among permissions, components, and API calls.

C. Real-Time On-Device Implementation

Optimizing DroidMat for **mobile device deployment** can provide real-time malware detection for end-users, reducing reliance on cloud-based analysis and improving privacy and security. Lightweight models and incremental learning techniques would be key.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 2, February 2026

|DOI: 10.15680/IJIRSET.2026.1502061|

D. Multi-Modal Feature Integration

Future enhancements may include combining manifest features with **network traffic, code semantics, and behavioral logs** to create a multi-modal feature set, increasing the system's ability to detect sophisticated malware.

E. Continuous Learning and Adaptive Security

Implementing **adaptive learning mechanisms** will allow the system to evolve with emerging malware trends. Continuous retraining using newly discovered malicious apps and incremental updates can maintain detection performance against evolving threats.

REFERENCES

[1] J. Senanayake, H. Kalutarage, and M. O. Al-Kadri, "Android Mobile Malware Detection Using Machine Learning: A Systematic Review," Electronics, vol. 10, no. 13, p. 1606, 2021.
[2] K. S. Ujjwal Reddy, S. Sibi Chakkaravarthy, M. Gopinath, and A. Mitra, "A Study on Android Malware Detection Using Machine Learning Algorithms," in ICISML, Springer, 2023, pp. –, 2023.
[3] O. E. Saied and K. H. Thanoon, "Static Analysis-based Detection of Android Malware Using Machine Learning Algorithms," Jurnal Sistem Informasi, vol. 14, no. 5, pp. 25–40, 2025.
[4] H. Babbar, "Detection of Android Malware in the Internet of Things Through the K-Nearest Neighbor Algorithm," Sensors, vol. 23, no. 16, p. 7256, 2023.
[5] Q. Xu et al., "Android Malware Detection Based on Behavioral-Level Features Using Graph Convolutional Networks," Electronics, vol. 12, no. 23, p. 4817, 2023.
[6] A. Taha et al., "An Intelligent Technique for Android Malware Identification Using Fuzzy Rank-Based Fusion," Technologies, vol. 13, no. 2, p. 45, 2025.
[7] H. Alkahtani, T. H. Aldhyani, and T. et al., "Artificial Intelligence Algorithms for Malware Detection in Android-Operated Mobile Devices," PLoS One, 2022.
[8] P. Anđelić et al., "Achieving High Accuracy in Android Malware Detection Using Genetic Programming Symbolic Classifier," Computers, vol. 13, no. 8, p. 197, 2024.
[9] Suhaib J. H. et al., "A Comprehensive Study of Malware Detection in Android Operating Systems," Asian Journal of Research in Computer Science, vol. 10, no. 4, pp. 30–46, 2021.
[10] B. Bhaskar et al., "Protectors of the Android Domain: Research into Mobile Malware Detection and Defense," Int. J. Intelligent Systems and Applications in Engineering, vol. 12, no. 1, pp. 639–646, 2023.
[11] P. Kalyankar, "ML for Android Malware Detection," Int. J. Science and Social Science Research, vol. 2, no. 1, pp. 148–151, 2024.
[12] S. M. K. et al., "Android Malware Detection Using Machine Learning with Neural Networks and SVC," Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol., vol. 11, no. 2, pp. 2081–2085, 2025.
[13] R. Gupta, "Innovative Approach to Android Malware Detection Using Rough Set Theory," Electronics, vol. 13, no. 3, p. 482, 2024.
[14] Safa J. Altaha and A. Aljughiman, "A Survey on Android Malware Detection Techniques Using Supervised Machine Learning," IEEE Access, early access, 2024.
[15] A. Muzaffar et al., "Investigating Feature and Model Importance in Android Malware Detection: An Empirical Survey," arXiv preprint, 2023.
[16] A. Bensaoud, J. Kalita, and M. Bensaoud, "A Survey of Malware Detection Using Deep Learning," arXiv preprint, 2024.
[17] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep Learning Based Android Malware Detection Using Real Devices," Computers & Security, vol. 89, pp. 101663, 2020.
[18] H. Rathore et al., "Detection of Malicious Android Applications: Classical Machine Learning vs. Deep Neural Network Integrated with Clustering," arXiv preprint, 2021.
[19] Kezang Dema and Thinley Jamtsho, "A Systematic Review on Android Malware Detection," Asian Journal for Convergence in Technology, 2025.
[20] Narmada B. et al., "CYBER SECURITY of Malware Detection on Android Apps," Int. J. Advanced Research in Computer and Communication Engineering, 2022.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details