International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

# Classification of Android Applications and Performance Evaluation of Machine Learning Model with HFST

**Umesh V. Nikam[#1], Vaishali M. Deshmukh[*2]**

**Submitted**: 25/11/2023 **Revised**: 05/01/2024 **Accepted**: 15/01/2024

**Abstract:** In the dynamic landscape of mobile applications, Android devices seamlessly integrate into our daily lives, offering diverse functionalities through various applications. However, the surge in malware and malicious software poses a significant challenge for security professionals and users alike. To address this issue, researchers and cyber security experts actively explore innovative methods. This research paper delves into the crucial domain of Android application categorization, evaluating the performance of a novel machine learning model incorporating a Hybrid Feature Selection Technique (HFST). Initially, the study identifies the top twenty significant features through information gain, feature selection, and chi-square methods. The classifier's performance is then assessed using these features. Subsequently, all 60 features selected through the three techniques are merged and out of the 60 features, 11 are identified as hybrid features that are common in at least two techniques. The study re-evaluates machine learning classifiers' performance using these 11 hybrid features, comparing the results with existing state-of-the-art techniques to illustrate the superiority of the proposed HFST-based technique. Furthermore, the study measures its impact on various performance metrics, including classification accuracy, precision, and f-measure, revealing notable enhancements across these parameters when employing the HFST. The application of this hybrid feature selection technique significantly improves the classification process, ultimately achieving an impressive classification accuracy of 98.11%, with precision at 97.56 and an f-measure of 97.99 for distinguishing between malicious and benign apps.

***Keywords****: Machine learning algorithms, Hybrid features, Classification and detection, Performance evaluation.*

## 1. Introduction

Targeted cyber attacks pose a significant and alarming threat on the Internet. The use of conventional executable files as a standard payload remains a prevalent choice among attackers. A report has indicated that executable files rank second among the most common types of malicious email attachments. To bypass antivirus programs, malicious actors often employ obfuscated malware. Traditional antivirus programs, relying on pattern matching-based detection, struggle to identify novel forms of malware. Although dynamic analysis is a potent technique, it consumes substantial time when scrutinizing suspicious files from the web. Moreover, it necessitates high-performance servers and licenses, which encompass commercial operating systems and applications.

To address this challenge, static detection methods, in conjunction with machine learning, offer a viable solution. These methods involve the extraction of features from an app's manifest file. The AndroidManifest.xml file within the APK contains crucial information about the application, encompassing permission, activities, services, and receivers. The extraction & analysis of this information facilitate a deeper understanding of app's capabilities and prerequisite.

The recent advancements in machine learning have amplified the effectiveness of app permissions in the realm of malware detection. Consequently, a combination of app permissions and machine learning techniques proves to be a valuable asset in the fight against malware.

In this study, we leverage machine learning techniques for the purpose of malware detection. Our research underscores the effectiveness of app permissions in conjunction with machine learning in a practical setting. Our dataset, sourced from various outlets, comprises 4890 samples spanning 13 distinct malware families. The experimental results strongly support the efficiency of our approach in detecting malware, even against packed malware and anti-debugging mechanisms. This paper contributes following key findings:

- App permissions, when combined with machine learning techniques, exhibit effectiveness in practical malware detection.
- Our method extends its effectiveness not only to known malware types but also to emerging ones.
- Our approach is robust against the packed malware & countermeasures against debugging.

[1].*Department of Computer Science & Engineering,*
*Prof. Ram Meghe Institute of Technology &research, Badnera, India*
*umeshnikam3@gmail.com*
*ORCID ID : 0009-0005-8257-6656*
[2] *Department of Computer Science & Engineering,*
*Prof. Ram Meghe Institute of Technology &research, Badnera, India*
*vmdeshmukh@mitra.ac.in*
**Corresponding Author Email: umeshnikam3@gmail.com*

The paper is structured as follows: Section 2 delves into related studies, while Section 3 explores the machine learning techniques pertinent to this research. Section 4 provides a detailed account of our experiment, and Section 5 evaluates the model using some parameters and comparative analysis with latest techniques. Finally, Section 6 concludes this study.

## 2. Related Work

The paper introduces a new machine learning model for the detection of Android malware, emphasizing the abnormal co-existence patterns of permissions and APIs in malware compared to benign apps. They create a novel dataset with varying co-existed permissions and API call levels, employing the FP-growth algorithm to extract relevant features. The model is evaluated against conventional machine learning methods and achieves a high accuracy of 98% in classifying Android malware, surpassing the state-of-the-art model (87% accuracy). Notably, frequent API co-existence proves more effective than using API features alone. This approach holds promise for future use with dynamic features in malware detection. [1]

The paper introduces a two-stage detection framework, FE-CaDF, for Android malware detection during the spread or downloads stage. It employs CNN for classifying binary malicious apps in the first stage and Principal Component Analysis (PCA) for multi-classification of malware types in the second stage. Features extracted are combined with traffic payload for classification. The cascade deep forest method adapts to various sample scales and proves effective in detecting encrypted Android malware transmission, even for unknown attacks. The framework has potential for extending to iOS and Windows application detection in future research. [2]

This paper offers a comprehensive review and taxonomy of machine learning methods for malware detection, addressing the challenges posed by evolving cyber threats. It analyzes 77 research works, focusing on accuracy, analysis type, and detection approaches. The research classifies machine learning algorithms into categories, evaluates recent methods, discusses detection challenges, and proposes solutions. An empirical study assesses multiple machine learning algorithms, with the aim of advancing malware detection techniques and inspiring future research in cybersecurity. [3]

Recent research has underlined the significant threat of malware in the digital world, leading to the adoption of new security measures. Traditional methods have struggled against modern, obfuscated malware. Deep Learning (DL) has gained prominence for its ability to detect novel malware and provide quick analysis. This paper investigates DL-based malware detection systems, focusing on various malware types like mobile, Windows, IoT, APTs, and Ransomware. It highlights the importance of proactive security and addresses the limitations of traditional methods. The research offers insights and a taxonomy for developing more effective mitigation approaches against both common and complex malware. [4]

The paper introduces NT-GNN, a novel graph neural network model for detecting Android malware based on network traffic graphs. Unlike other systems focusing on pairwise traffic, NT-GNN considers complex structural relationships, achieving a 97% accuracy on Android malware datasets. It outperforms deep learning methods with high precision, recall, and F1 scores. Future work involves classifying malware into families and comparing NT-GNN with other graph representation models, as well as enhancing performance by extracting static and dynamic features from the dataset. [5]

The increasing popularity of mobile devices has led to a surge in malicious Android apps, which are becoming harder to detect due to advanced obfuscation. Manual and static methods are insufficient, and dynamic analysis is time-consuming. To tackle this, a hybrid approach combining static and dynamic analysis features is proposed. Two datasets for malware detection and family classification were created for research, and machine learning algorithms were used. This hybrid approach outperformed using static or dynamic features alone, offering benchmark datasets for testing new techniques. Future work will address data imbalance in malware classification using deep learning and big data tools. [6]

Traditional malware detection methods are ineffective against new and generic malware. Researchers created a dataset with 16,300 records and 215 features from various malware sources. They propose a supervised machine learning approach using feature reduction and ensembling techniques, with CatBoost showing the highest performance (93.15% accuracy, ROC 0.91, Kappa Score 81.56%). This study underscore using machine learning for detecting malware and highlights the need for comprehensive datasets. The proposed method, particularly with the CatBoost classifier, yields promising results in accurate malware prediction and classification. [7]

This paper assesses the effectiveness of multiple machine learning techniques for detection and classification of malwares. Techniques like SMOTE, feature normalization, and PCA are applied to enhance accuracy. The paper introduces a Light Gradient Boosting Model to classify Android malware into five categories. The research uses a substantial dataset of 11,598 APKs from diverse sources provided by the Canadian Institute of Cybersecurity, addressing the challenges of malware classification. [8]

Executable files, often obfuscated, remain a popular threat to endpoint computers. Dynamic analysis of such files is

time-consuming. This paper suggests using natural language processing (NLP) techniques on printable strings for efficient malware detection. It's applied to a dataset of over 500,000 samples, proving effective against various types of malware, including new and packed ones. Although the study has limitations, it lays the foundation for practical performance evaluation. Future work will concentrate on analyzing sample specifics & enhance packer detection. [9]

This paper introduces a hybrid analysis approach, merging static and dynamic malware analysis for better Android malware detection and classification. The framework consists of three phases: pre-processing, feature selection, and a detection model using an enhanced neural network and improved HHO optimization. This hybrid approach demonstrates enhanced accuracy in Android malware detection and classification compared to analyzing static and dynamic aspects separately. [10] Summary of related work is shown in Table1 below.

**Table 1.** Related Work Summary

| References | Dataset Used | Algorithms Used | Advantages |
|---|---|---|---|
| **[1]** | New dataset with different levels of co-existed permissions and API calls | Frequent pattern growth (FP-growth) algorithm | • High accuracy in classifying Android malware. Outperforms state-of-the-art model with 98% accuracy using Malgenome dataset compared to 87% with state-of-the-art.<br>• Frequent API co-existence more effective than using API features alone. |
| **[2]** | Various datasets including encrypted transmission of Android malware | CNN, PCA<br>cascade deep forest method | • Effective in detecting encrypted transmission of Android malware, including unknown attacks.<br>• Potential for detecting iOS and Windows applications. |
| **[3]** | 77 selected research works | Various machine learning algorithms based on performance accuracy, analysis type, and detection approaches | • Provides taxonomy for classifying ML algorithms in malware detection.<br>• Aims to foster improvements in malware detection techniques. |
| **[4]** | Android malware datasets CICAndMal2017 and AAGM | NT-GNN (Graph Neural Network model) | • Outperforms other DL approaches with high precision, recall, and F1 scores. |
| **[5]** | Two datasets for detection and family classification of Android malware | Hybrid approach with machine learning algorithms | • Improved accuracy in detecting and classifying Android malware compared to static or dynamic features alone. |
| **[6]** | Custom dataset with various malware files | Machine learning classifiers. | • Accuracy of machine learning algorithms is improved with hybrid features. |
| **[7]** | Comprehensive dataset of 16,300 records and 215 features | Supervised machine learning classifiers, feature reduction techniques, ensembling techniques | • CatBoost classifier shows highest performance with 93.15% accuracy, ROC value of 91%, 81.56% Cohen Kappa Score. |
| **[8]** | Dataset consisting of 11,598 APKs | Synthetic Minority Oversampling Technique (SMOTE), feature normalization, Principal Component Analysis (PCA) | • Light Gradient Boosting Model classifies Android malware into five classes with high accuracy. |
| **[9]** | Dataset with over | NLP techniques on printable | • Effective against existing, packed and |

| | | | |
|---|---|---|---|
| | 500,000 samples | strings. | new malware and anti-debugging techniques. |
| **[10]** | Custom datasets for detection and family classification of Android malware | Hybrid approach with neural network optimized by advanced HHO version. | • Improved accuracy in detecting and classifying Android malware compared to static or dynamic features alone. |

## 3. Proposed Work

Below Figure 1, shows a proposed methodology for hybrid feature selection technique. All the phases are explained below in three phases:

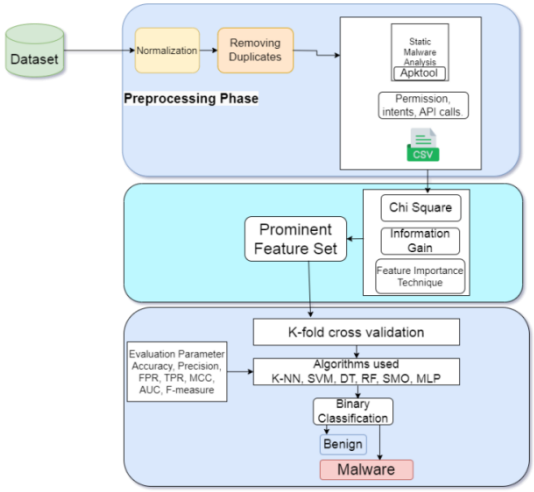


**Fig. 1** Proposed Model

**a. Pre-processing Phase:** For optimal results in machine learning, thorough pre-processing is vital. This includes tasks like removing duplicates, handling missing values (NaN), and normalizing or scaling data. MinMax scaling is a method used for feature normalization, particularly suitable for datasets with low variance. [11] Normalization involves rescaling values to a specified interval, often between 0 and 1, ensuring proper scaling for models dependent on input feature values. The formula for MinMax scaling (Equation 1) is employed for data normalization.

$$Y_{\text{norm}} = \frac{Y_i - Y_{\min}}{Y_{\max} - Y_{\min}} \qquad \text{Eq.(1)}$$

Here, Yi represents the initial value of the feature, and the denominator reflects the range between a new normalized max and minimum values for that feature. Duplicate Android apps can be removed with drop_duplicates() function. [12] Malware features are extracted through static analysis, which involves collecting API calls, intents, permissions and command strings using a custom Python script with Apk tool. A prominent feature set with the highest importance is then created using hybrid feature selection techniques. [13]

**b. Prominent Feature Selection Phase**: Selecting the right features plays a pivotal role in virus detection. Inaccurate feature choices can lead to diminished model accuracy, while judicious selections can yield a high level of precision. [14] To address this concern, we utilize three distinct methods of feature selection namely: information gain, a feature importance and a chi square technique. In an initial step, we pick the top 20 attributes from each of these techniques. Subsequently, we aggregate all 60 features, and the ones that overlap across at least two of these methods are designated as the ultimate hybrid features. [15] The subsequent section outlines these techniques of feature selection in detail.

**Algorithm1: Hybrid Feature Selection Algorithm**

**Input:** Training a dataset( D) using static feature set (S)

**Output:** The subset hybrid features, denoted as Fnew

1. Define a function feature select (D, S) for feature selection as: def feature_select(D,S).
2. Find the importance of features with the help of chi square, Information Gain, & Feature Importance Technique, based on the features in S.
3. Create S1 of top 20 features by removing features with scores of 0 and NaN using the chi-square method. S1 = remove_features(with 0 score & NaN values)
4. Create S2 of top 20 features by removing features with scores of 0 and NaN using I. G method. S2 = remove_features (with 0 score & NaN values)
5. Create S3 of top 20 features by removing features having a 0 score & NaN valuses using Feature Importance method. S3 = remove_features(with 0 score & NaN values)
6. Compute an average performance model (A) by aggregating features selected in S1, S2, and S3. A = average_model(S1,S2,S3)
7. Identify the best subset of features (M) from S1, S2, and S3. M = best_feature_subset(S1,S2,S3)
8. Update S with the best feature subset obtained in the previous step. S = best-subset(S1,S2,S3)
9. Return the prominent subset of features as Fnew. return Fnew

Information gain technique furnishes us with the gain associated with each feature in a given dataset. The utmost significance lies in the feature with the highest gain value. Figure 2 serves as an illustrative catalogue of top 20 features that have been singled out using the IG methodology.

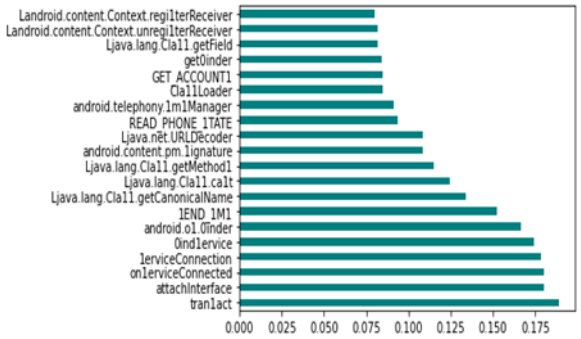


**Fig. 2** Top (20) Features with Information Gain.

Chi-Square technique, aims to identify robust associations between dependent and independent features. [16] By considering the highest 20 association values, this approach selects 20 features from a four used datasets as shown in Figure 3.

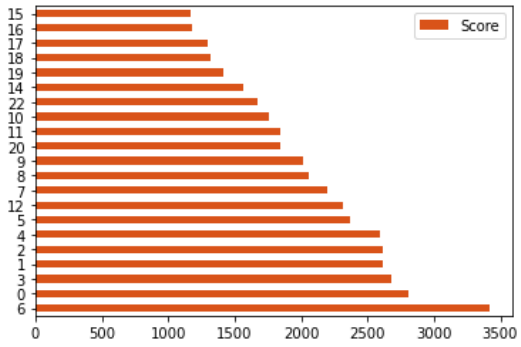


**Fig. 3 Top (20) Features with Chi-square.**

Feature Importance Approach, allows for the assessment of the importance of each feature within a dataset, with greater importance associated with higher feature weight. [17] List of these top 20 features in a dataset based on the weight attributed to each feature is shown in Figure 4 below.

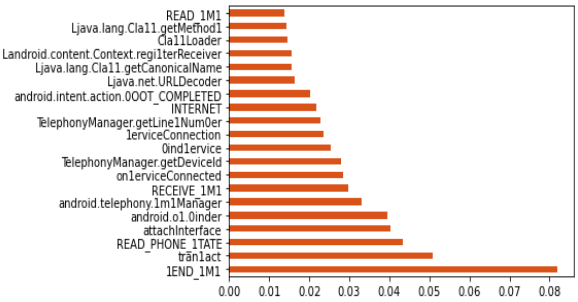


**Fig. 4** Top (20) Features with Feature Importance.

From a close examination of Figure 2, 3, and 4, it becomes evident that each of these techniques has unearthed a distinct set of the top 20 features. To harness the collective strength of these methodologies for effective malware identification, we combine all the 60 features derived from three techniques. Subsequently, we pinpoint & designate as "hybrid features" those are common to at least two (2) of these methods. This results in the identification of 11 hybrid features out of the total 60. Below is a list for hybrid features shown in Table 2.

**Table 2.** Hybrid Features Common in At least Two Techniques.

| S. N | Common Hybrid Features. |
|---|---|
| **a.** | RECEIVE_1M1 |
| **b.** | LANDROID.CONTENT.CONTEXT.REGI1TERRECEIVER |
| **c.** | ANDROID.TELEPHONY.1M1MANAGER |
| **d.** | ANDROID.CONTENT.PM.1IGNATURE |
| **e.** | LJAVA.NET.URLDECODER |
| **f.** | 1END_1M1 |
| **g.** | ANDROID.O1.0INDER |
| **h.** | 1ERVICECONNECTION |
| **i.** | 0INDLERVICE |
| **j.** | ONLERVICECONNECTED |
| **k.** | TRANLACT |

Below algorithm explain steps of finding hybrid features.

**a. Evaluation Parameters & Experimental Setup**: Various metrics employed to assess the performance of classifiers are explained below:

**True Positive Rate-Recall:** TPR is also called Recall. It can be calculated with the division of count of correctly identified positive samples by the total count of positive samples. [18] As depicted in the equation (2). It is computed using the below formula:

$$TPR = TP/(TP+FN) \quad \text{Eq. (2)}$$

**False Positive Rate:** FPR indicates a fraction of false positive instances concerning the overall count of of true negative instances. [19] Below equation (3) outline computation of this. $FPR = FP/(TN+FP)$ Eq. (3)

**Precision:** Precision is calculated with the division of number of correct instances by the total count of accurate instances. [20] It is computed with equation (4).

$$Precision = TP/(TP+FP) \quad \text{Eq. (4)}$$

**F-Measure:** It represents the harmonic average of recall & precision. [21] It is determined with equation (5). $F\text{-measure}=(2*Precision*Recall)/(Precision+Recall)$ Eq. (5)

**Accuracy:** It can be computed by a division of count of cases with a sum of true negatives and true positive instances. [22] The calculation for accuracy is as follows in equation (6).

$$Accuracy=(TP+TN)/(TP+FP+FN+TN) \quad \text{Eq. (6)}$$

**MCC (Matthews correlation coefficient):** It serves as a benchmark for assessing the performance of binary classifiers. [23] It falls within the numeric range of +1 to -1. In this context, a value of +1 means accurate prediction, & a -1 meaning opposite prediction. The calculation for MCC, as represented by equation (7), is as follows:

$$MCC=(TP*TN-FP*FN)/\sqrt{[(TP+FN)(TP+FP)(TN+FP)(TN+FN)]} \quad \text{Eq. (7)}$$

**AUC-ROC Curve:** This curve is used for the assessment of a classification model. [24] Its function is to quantitatively measure model's ability to distinguish between different classes.

In this context, True Positives (TP) represents instances correctly classified as "Yes," whereas False Positive (FP) is when an instance is mistakenly classified as "Yes." [25] True Negative denotes cases that were correctly excluded from the "Yes" category when expected. Conversely, False Negatives (FN) refers to cases that were predicted to not belong to the "Yes" category but actually did. [26]

## 4. Experiment

An experiment was conducted to assess the effectiveness of two sets of features in enhancing machine learning classifier performance: one set of hybrid features presented in Table 2 and another set of top 20 features displayed in Figure 2, 3 & 4. As depicted in Figure 1 under the Methodology section, a ten-fold cross-validation technique was employed, which is statistically reliable for evaluating classifier performance. 70:30 atios are used for training & testing of a dataset. Accuracy, FPR, TPR, Precision, F-measure, MCC & AUC were utilized to gauge classifier performance. Table 4, 5 & 6 showcases the comparative performance of classifiers with the use of chi-square, IG and feature importance methods. Table 4 reveals that chi-square technique led to improved performance across all classifiers as compared to IG, and feature selection techniques. With chi-square technique as shown in Table 4, the Random Forest algorithm has shown the highest precision, accuracy and F Measure while Table 7 illustrates performance of classifiers when employing HFST technique with chi-square features.

Following a performance evaluation of classifier with 20 top features, the impact of hybrid features was also examined. A subsequent experiment was carried out, and the same classifiers were assessed using the hybrid features detailed in Table 2. The results are presented in Table 8. A comparison between the outcomes in Table 7 and Table 8 clearly demonstrates a significant enhancement in classifier performance with the inclusion of hybrid features.

**a. Dataset Used:** In this study, four distinct datasets were employed, sourced from Kaggle [27]. These datasets comprise a collection of 1910 instances of malware samples and 2980 instances of benign samples. Together, they encompass 215 attributes, with the distribution as follows: manifest permissions account for 53%, API call signatures for 33%, and the remaining 14% encompass other attributes. Each dataset entry pertains to the attributes associated with various applications, with values denoted as 0 or 1. A value of 0 signifies that a specific attribute does not necessitate permission, whereas a value of 1 indicates the requirement for permission. [28] Additionally, each dataset incorporates a column indicating whether an application is categorized as malicious or benign. For a detailed breakdown of the collected samples used to construct our dataset, please refer to Table 3.

**Table 3.** Distribution of Used Dataset.

| Used Dataset | Total Sample | Malwares Count | Benign Count |
|---|---|---|---|
| DREBIN | 1400 | 450 | 950 |
| CICANDMAL 2017 | 1240 | 450 | 790 |
| APK MIRROR | 1200 | 410 | 790 |
| VIRUS SHARE | 1050 | 600 | 450 |
| TOTAL | 4890 | 1910 | 2980 |

**b. Detecting Malwares with Static Features:** As previously discussed, the chi square method has proven to be the preferred choice for selecting static features. Table 4 to 6 present findings related to classification precision, accuracy, and F-measure scores obtained by applying various classification algorithms using chi square, IG and a Feature Importance Techniques respectively. The results in these tables clearly indicate that, on average, chi-square outperforms the other metrics in terms of selecting static features. Table 7 outlines the outcomes of binary detection with static features. Remarkably, with HFST, we attain a peak accuracy of 96.91%, showcasing its remarkable performance. As for other methods such as SMO, KNN, SVM, DT, RF, NB, and MLP, their accuracy rates range from 92.31% to 95.52%. It's worth noting that Naïve Bayes accuracy relies on probability

distribution, and it could potentially benefit from additional data examples to enhance its performance. In summary, HFST technique demonstrates commendable performance in binary classification with static features. HFST, in particular, achieves an impressive MCC (Matthews correlation coefficient) of 93.8%, signifying a substantial performance improvement compared to other standard models. During testing, HFST consistently reaches a peak accuracy value of 96.91% on a 7th epoch, while the accuracy of training varies between 81.11% to 98.71%, indicating a stable convergence of training accuracy. Test accuracy falls within the range of 79.55% to 95.11%.

**Table 4.** Evaluation of Classifiers with Chi-square method.

| ALGORITHMS | KNN | SMO | SVM | Random Forest | Decision Tree | Naïve Bayes | MLP | Average |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.9231 | 0.9350 | 0.9231 | 0.9582 | 0.9504 | 0.9421 | 0.9351 | **0.9382** |
| Precision | 0.9172 | 0.9291 | 0.9081 | 0.9423 | 0.9336 | 0.9171 | 0.9332 | **0.9258** |
| F Measure | 0.9172 | 0.9291 | 0.9162 | 0.9543 | 0.9451 | 0.9365 | 0.9294 | **0.9325** |

**Table 5.** Evaluation of Classifiers with Information Gain Method.

| ALGORITHMS | KNN | SMO | SVM | Random Forest | Decision Tree | Naïve Bayes | MLP | Average |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.9041 | 0.9152 | 0.9023 | 0.9123 | 0.9411 | 0.9223 | 0.9421 | 0.9199 |
| Precision | 0.9252 | 0.9254 | 0.9012 | 0.9092 | 0.9334 | 0.9103 | 0.9305 | 0.9193 |
| F Measure | 0.9041 | 0.9123 | 0.9012 | 0.9044 | 0.9402 | 0.9164 | 0.9335 | 0.9160 |

**Table 6.** Evaluation of Classifiers with Feature Selection Method.

| ALGORITHMS | KNN | SMO | SVM | Random Forest | Decision Tree | Naïve Bayes | MLP | Average |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.9322 | 0.9332 | 0.9162 | 0.9342 | 0.9483 | 0.9444 | 0.9311 | 0.9342 |
| Precision | 0.9221 | 0.9273 | 0.9102 | 0.9312 | 0.9163 | 0.9154 | 0.9271 | 0.9213 |
| F Measure | 0.9254 | 0.9215 | 0.9112 | 0.9281 | 0.9444 | 0.9404 | 0.9282 | 0.9284 |

**Table 7.** Performance Evaluation of various classifiers & HFST technique with Chi-square features.

| ALGORITHMS | KNN | SMO | SVM | Random Forest | Decision Tree | Naïve Bayes | MLP | HFST Technique |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.9231 | 0.9352 | 0.9234 | 0.9552 | 0.9504 | 0.9421 | 0.9352 | 0.9691 |
| False Positive Rate | 0.0711 | 0.0642 | 0.0776 | 0.0498 | 0.0561 | 0.0691 | 0.0582 | 0.0292 |
| True Positive Rate | 0.9172 | 0.9333 | 0.9242 | 0.9662 | 0.9573 | 0.9572 | 0.9263 | 0.9672 |
| Precision | 0.9172 | 0.9253 | 0.9082 | 0.9422 | 0.9334 | 0.9172 | 0.9334 | 0.9672 |
| F-Measure | 0.9172 | 0.9291 | 0.9162 | 0.9542 | 0.9452 | 0.9362 | 0.9293 | 0.9672 |
| MCC | 0.8452 | 0.8686 | 0.8453 | 0.9156 | 0.8997 | 0.8842 | 0.8695 | 0.9381 |
| AUC | 0.9232 | 0.9312 | 0.9152 | 0.9465 | 0.9396 | 0.9243 | 0.9388 | 0.9692 |

**c. Malware Classification with Hybrid Features:** The challenge of accurately classifying malware arises from its characteristics of execution

stalling and obfuscation, making it difficult to achieve with a single static or dynamic technique. [29] In order to address this issue, we have adopted a hybrid features approach. In an initial step, we pick the top 20 attributes from each of these techniques. Subsequently, we aggregate all 60 features, and the ones that overlap across at least two of these methods are designated as the ultimate hybrid features. In our analysis, we applied seven different machine learning classifiers for detecting and classifying binary malwares. The outcomes of binary classification evaluations, utilizing machine learning techniques on hybrid features, are presented in Table 7. Notably, a HFST model outperforms the previously mentioned classifiers in terms of accuracy. Specifically, HFST achieves an accuracy rate of 98.11%, surpassing Random Forest and Decision Tree, which attain 96.99% & 96.22% result, respectively. Performance of all the classifiers is shown in figure 5.

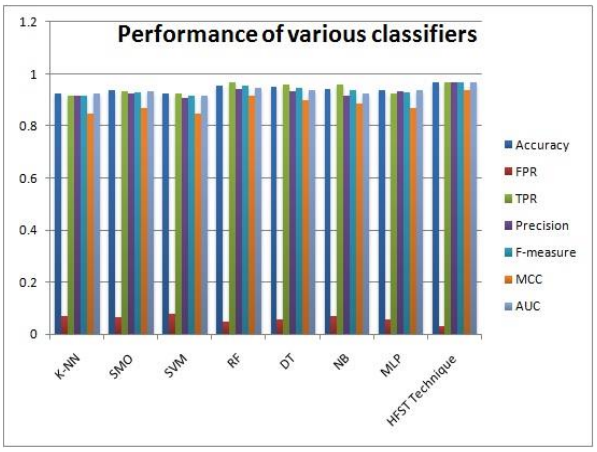


**Fig. 5** Performance of Various Classifiers.

**d. Accuracy Comparison:** As shown in Table 9, HFST technique has shown highest accuracy using both for static & hybrid features. However it has shown accuracy of 98.11% notably with hybrid features and 96.11 with static features. The findings in the Table 9 graphs in figure 6 & 7 reveal that use of hybrid features has enhanced accuracy of all classifiers by nearly 2%.

**Table 8.** Performance of various classifiers & HFST technique with hybrid features.

| ALGORITHMS | KNN | SMO | SVM | Random Forest | Decision Tree | Naïve Bayes | MLP | HFST Technique |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.9466 | 0.9581 | 0.9423 | 0.9699 | 0.9622 | 0.9465 | 0.9466 | **0.9811** |
| False Positive Rate | 0.0681 | 0.0490 | 0.0632 | 0.0355 | 0.0420 | 0.0564 | 0.0566 | **0.0210** |
| True Positive Rate | 0.9655 | 0.9666 | 0.9492 | 0.9755 | 0.9669 | 0.9497 | 0.9499 | **0.9832** |
| Precision | 0.9177 | 0.9423 | 0.9211 | 0.9588 | 0.9509 | 0.9333 | 0.9333 | **0.9756** |
| F-Measure | 0.9404 | 0.9542 | 0.9377 | 0.9666 | 0.9588 | 0.9411 | 0.9411 | **0.9799** |
| MCC | 0.8923 | 0.9152 | 0.8843 | 0.9388 | 0.9233 | 0.8923 | 0.8922 | **0.9612** |
| AUC | 0.9244 | 0.946 | 0.9311 | 0.9623 | 0.9543 | 0.9387 | 0.9388 | **0.9777** |

**Table 9.** Accuracy comparison of various classifiers with HFST technique.

| **ALGORITHMS** | **KNN** | **SMO** | **SVM** | **RF** | **DT** | **NB** | **MLP** | **HFST Technique** | **Findings** |
|---|---|---|---|---|---|---|---|---|---|
| ACCURACY (Hybrid Features) | 0.9466 | 0.9581 | 0.9423 | 0.9699 | 0.9622 | 0.9465 | 0.9466 | 0.9811 | **The use of hybrid features has enhanced the accuracy of all classifiers by nearly 2%.** |
| ACCURACY (Static Features) | 0.9231 | 0.9352 | 0.9234 | 0.9552 | 0.9504 | 0.9421 | 0.9352 | 0.9691 | |

## 5. Comparative Analysis

Precision and recall metrics were assessed, as detailed in Tables X. The research conducted by Laya Taheri et al. [32] involved the utilization of the random forest algorithm to compute precision and recall for the dataset. In contrast, our approach, employing the HFST algorithm, yielded the most impressive outcomes. Our

study represents advancement over previous research in the realms of static feature analysis. As depicted in Table 10, our methodology attains a highest recall rate of 97.99% while classifying malware binaries. In comparison with state-of-the-art methods, our approach significantly enhances the performance of binary malware classification in static categorization. In Table 10, the result also shows that HFST achieves the highest precision level of 97.56%. In contrast, other investigations report their top precision rates at 93.8%, 89%, and 85.9%.

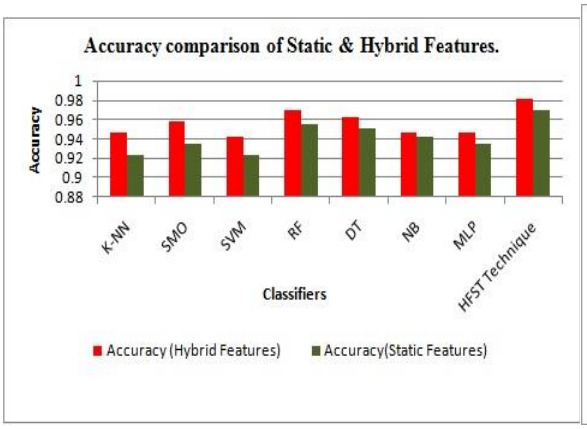


**Fig. 6** Accuracy comparison of static & hybrid features

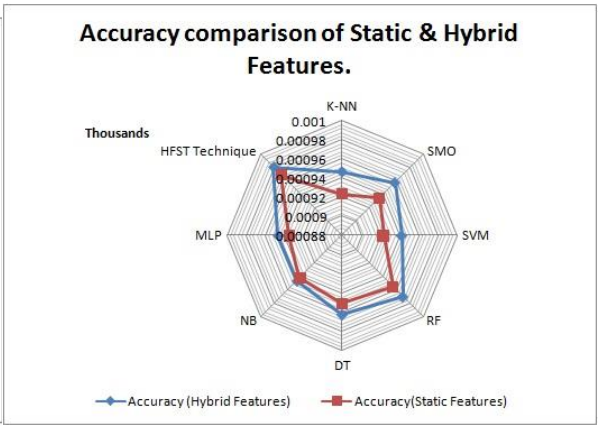


**Fig. 7** Accuracy comparison of static & hybrid features

**Table 10.** Comparative analysis of HFST with latest techniques.

| Various Techniques Results | Precision | Recall |
|---|---|---|
| Mohammad Kamel A. Abuthawabeh et. al. [31] | 89%(Random Forest) | 83.22%(Random Forest) |
| Laya Taheri et. al.[32] | 85.9%(Random Forest) | 88.3%(Random Forest) |
| Mohammad Kamel A. Abuthawabeh et. al. [31] | 85.8%(Decision Tree) | 86.1%(Decision Tree) |
| Arash Habibi Lashkari et. al.[33] | 85.4%(KNN) | 88.2%(KNN) |
| Ibrahim Aljarah et. al. [34] | 93.8%(Decision Tree) | 94.36%(Decision Tree) |
| **HFST Technique** | **97.56% (Hybrid Features)** | **97.99% (Hybrid Features)** |

**a. Feature Selection Effect:** The suggested method for selecting hybrid features has a notable influence on the quantity of features. A reduction in the number of features has a positive effect on the performance of evaluation metrics. [30] In the case of static features, filter methods are employed to identify the most prominent feature subset to engage in the malware detection process. By implementing the suggested model for malware detection in feature selection, this approach enhances detection accuracy while simultaneously diminishing the occurrence of false negatives and false positives in identifying malware applications.

## 6. Conclusion & Future Work

Constant vigilance is crucial for Android users due to the persistent threat of mobile viruses. The paper explores the effectiveness of hybrid features derived from three selection strategies in malware detection, demonstrating that the hybrid approach is the most effective, resulting in 98.11% classification accuracy for distinguishing malicious and benign apps. Future research directions includes adaptive algorithms, deep learning and multi-modal analysis for an improved detection, as well as for the investigation of federated learning, enhancing explain ability, & addressing challenges for large-scale deployment in order to enhance use privacy & effectiveness. This work can be expanded for zero day malware detection, real time monitoring, & in-depth analysis of the Android app market to proactively address evolving threats.


### Acknowledgment

The author wish to convey his appreciation to Dr. Vaishali M. Deshmukh for her assistance and encouragement during the course of the research.


### Author contributions

Umesh V. Nikam is a main author for implementing a concept, calculating results and a documentation of paper. Vaishali M. Deshmukh has reviewed the work

and guided for implementation and writing of this research paper.

**Conflicts of interest**

The authors declare no conflicts of interest.

## References


[1] E. Odat and Q. M. Yaseen, "A Novel Machine Learning Approach for Android Malware Detection Based on the Co-Existence of Features," in IEEE Access, vol. 11, pp. 15471-15484, 2023, doi: 10.1109/ACCESS.2023.3244656.

[2] X. Zhang, J. Wang, J. Xu and C. Gu, "Detection of Android Malware Based on Deep Forest and Feature Enhancement," in IEEE Access, vol. 11, pp. 29344-29359, 2023, doi: 10.1109/ACCESS.2023.3260977.

[3] N. Z. Gorment, A. Selamat, L. K. Cheng and O. Krejcar, "Machine Learning Algorithm for Malware Detection: Taxonomy, Current Challenges and Future Directions," in IEEE Access, doi: 10.1109/ACCESS.2023.3256979.

[4] Gopinath M., Sibi Chakkaravarthy Sethuraman, A comprehensive survey on deep learning based malware detection techniques, Computer Science Review, Volume 47, 2023, 100529, ISSN 1574-0137,https://doi.org/10.1016/j.cosrev.2022.100529.

[5] Liu, Tianyue & Li, Zhenwan & Long, Haixia & Bilal, Anas. (2023). NT-GNN: Network Traffic Graph for 5G Mobile IoT Android Malware Detection. Electronics.12.10.3390/electronics12040789.

[6] Dhalaria, Meghna & Gandotra, Ekta. (2020). A Hybrid Approach for Android Malware Detection and Family Classification. International Journal of Interactive Multimedia and Artificial Intelligence. In Press. 1. 10.9781/ijimai.2020.09.001.

[7] P. Agrawal and B. Trivedi, "Evaluating Machine Learning Classifiers to detect Android Malware," 2020 IEEE International Conference for Innovation in Technology (INOCON), Bangluru, India, 2020, pp. 1-6, doi: 10.1109/INOCON50539.2020.9298290.

[8] AlOmari, Hani & Yaseen, Qussai & Al-Betar, Mohammed. (2023). A Comparative Analysis of Machine Learning Algorithms for Android Malware Detection. Procedia Computer Science. 220. 763-768. 10.1016/j.procs.2023.03.101.

[9] Mimura, M., Ito, R. Applying NLP techniques to malware detection in a practical environment. Int. J. Inf. Secur. 21, 279–291 (2022).

[10] Taher, Dr. Fatma & AlFandi, Omar & Kfairy, Mousa & Al Hamadi, Hussam & Alrabaee, Saed. (2023). DroidDetectMW: A Hybrid Intelligent Model for Android Malware Detection. 10.20944/preprints202305.0333.v1.

[11] Bherde, Gajanan & Pund, Mahendra. (2020). Strategy and Knowledge-Based XML Attack Detection Systems using Ontology. International Journal of Recent Technology and Engineering (IJRTE). 8. 798-801. 10.35940/ijrte.E5786.018520.

[12] A. A. Darem, F. A. Ghaleb, A. A. Al-Hashmi, J. H. Abawajy, S. M. Alanazi and A. Y. Al-Rezami, "An Adaptive Behavioral-Based Incremental Batch Learning Malware Variants Detection Model Using Concept Drift Detection and Sequential Deep Learning," in IEEE Access, vol. 9, pp. 97180-97196, 2021, doi: 10.1109/ACCESS.2021.3093366.

[13] Agrawal, Prerna & Trivedi, Bhushan. (2021). AndroHealthCheck: A Malware Detection System for Android Using Machine Learning. 10.1007/978-981-16-0965-7_4.

[14] Shhadat, Ihab & Al-bataineh, Bara & Hayajneh, Amena & Al-Sharif, Ziad. (2020). The Use of Machine Learning Techniques to Advance the Detection and Classification of Unknown Malware. Procedia Computer Science. 170. 917-922. 10.1016/j.procs.2020.03.110.

[15] Li, Shanxi & Zhou, Qingguo & Zhou, Rui & Lv, Qingquan. (2022). Intelligent malware detection based on graph convolutional network. The Journal of Supercomputing. 78. 10.1007/s11227-021-04020-y.

[16] Zhang, Xiao-Lei & Xu, Menglong. (2022). AUC optimization for deep learning-based voice activity detection. EURASIP Journal on Audio, Speech, and Music Processing. 2022. 10.1186/s13636-022-00260-9.

[17] Akhtar, Muhammad Shoaib & Feng, Tao. (2022). Malware Analysis and Detection Using Machine Learning Algorithms. Symmetry. 14. 2304. 10.3390/sym14112304.

[18] Kim, Jinsung & Ban, Younghoon & Ko, Eunbyeol & Cho, Haehyun & Yi, Jeong. (2022). MAPAS: a practical deep learning-based android malware detection system. International Journal of Information Security. 21. 1-14. 10.1007/s10207-022-00579-6.

[19] Peng, T., Hu, B., Liu, J., Huang, J., Zhang, Z., He, R., & Hu, X. (2022). A Lightweight Multi-Source Fast Android Malware Detection Model. Applied Sciences, 12(11), 5394. MDPI AG. Retrieved from http://dx.doi.org/10.3390/app12115394

[20] Taha, Altyeb. (2016). Classification of Android Malware Applications using Feature Selection and Classification Algorithms. VAWKUM Transactions on Computer Sciences. 10. 1. 10.21015/vtcs.v10i1.412.

[21] Aljanabi, Maryam & Altamimi, Ahmad. (2022). A Comparative Analysis of Machine Learning Techniques for Classification and Detection of Malware.

[22] Diptiben Ghelani. A perspective study on Malware detection and protection, A review. Authorea. September 13, 2022. DOI: 10.22541/au.166308976.63086986/v1

[23] Sunil Gupta, Kamal Saluja, Ankur Goyal, Amit Vajpayee, Vipin Tiwari,Comparing the performance of machine learning algorithms using estimated accuracy, Measurement: Sensors,Volume24,2022,100432,ISSN 26659174,https://doi.org/10.1016/j.measen.2022.100432.

[24] Asam, Dr & Khan, Saddam & Akbar, Altaf & Bibi, Sameena & Jamal, Tauseef & Khan, Asifullah & Ghafoor, Usman & Bhutta, Raheel. (2022). IoT malware detection architecture using a novel channel boosted and squeezed CNN. Scientific Reports. 12. 10.1038/s41598-022-18936-9.

[25] P. R. K. Varma, K. P. Raj and K. V. S. Raju, "Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2017, pp. 294-299, doi: 10.1109/I-SMAC.2017.8058358.

[26] V. J. Raymond and R. J. R. Raj, "Investigation of android malware with machine learning classifiers using enhanced pca algorithm," Computer Systems Science and Engineering, vol. 44, no.3, pp. 2147–2163, 2023.

[27] Albakri, Ashwag & Alhayan, Fatimah & Alturki, Nazik & Ahamed, Saahirabanu & Shamsudheen, Shermin. (2023). Metaheuristics with Deep Learning Model for Cybersecurity and Android Malware Detection and Classification. Applied Sciences. 13. 2172. 10.3390/app13042172.

[28] U. V. Nikam and V. M. Deshmuh, "Performance Evaluation of Machine Learning Classifiers in Malware Detection," 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), Ballari, India, 2022, pp. 1-5, doi: 10.1109/ICDCECE53908.2022.9793102.

[29] U. Nikam and V. M. Deshmukh, “Hybrid Feature Selection Technique to classify Malicious Applications using Machine Learning approach”, J Integr Sci Technol, vol. 12, no. 1, p. 702, Aug. 2023.

[30] Nikam, U.V., Deshmukh, V.M. (2023). Efficient Approach for Malware Detection Using Machine Learning Classifier. In: Tiwari, R., Pavone, M.F., Saraswat, M. (eds) Proceedings of International Conference on Computational Intelligence. ICCI 2022. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-99-2854-5_14

[31] M. K. A. Abuthawabeh and K. W. Mahmoud, "Android Malware Detection and Categorization Based on Conversation-level Network Traffic Features," 2019 International Arab Conference on Information Technology (ACIT), Al Ain, United Arab Emirates, 2019, pp. 42-47, doi: 10.1109/ACIT47987.2019.8991114.

[32] L. Taheri, A. F. A. Kadir and A. H. Lashkari, "Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls," 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 2019, pp. 1-8, doi: 10.1109/CCST.2019.8888430.

[33] A. H. Lashkari, A. F. A. Kadir, L. Taheri and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification," 2018 International Carnahan Conference on Security Technology (ICCST), Montreal, QC, Canada, 2018, pp. 1-7, doi: 10.1109/CCST.2018.8585560.

[34] I. Aljarah et al., "A Robust Multi-Objective Feature Selection Model Based on Local Neighborhood Multi-Verse Optimization," in IEEE Access, vol. 9, pp. 100009-100028, 2021, doi: 10.1109/ACCESS.2021.3097206.