Kufa Journal of Engineering
Vol. 16, No. 2, April 2025, P.P. 96 -118
Article history: Received 21 June 2024, last revised 6 August 2024, accepted 5 September 2024

CLASSIFYING ANDROID MALWARE CATEGORIES BASED ON DYNAMIC FEATURES: AN INTEGRATION OF FEATURE REDUCTION AND SELECTION TECHNIQUES

Abdullah Al-Sraratee[1*] and Ahmed Al-Azawei[2]

[1*]College of Information Technology, University of Babylon, Babil, Iraq.
Email:Abdullahallawim.sw@student.uobabylon.edu.iq

[2] College of Information Technology, University of Babylon, Babil, Iraq.
Email:ahmedhabeeb@itnet.uobabylon.edu.iq

https://doi.org/10.30572/2018/KJE/160206

ABSTRACT

Android malware has grown steadily into a major internet threat. Despite efforts to identify and categorize malware in seemingly safe Android apps, addressing this issue is still lacking. Therefore, understanding the unique behaviors of common Android malware categories is essential. This study utilizes machine learning techniques namely, K-Nearest Neighbor, Random Forest and Decision Tree to classify Android malware based on dynamic analysis. As feature selection and reduction techniques, Mutual Information and Principle Component Analysis are used. The research analyzes a large dataset, containing fourteen primary malware categories using the CCCS-CIC-AndMal2020 dataset. Unlike previous research, the proposed method makes a balance between the number of features and classifiers' performance, resulting in an overall detection accuracy of 98% in the fourteen analyzed categories and excluding 78.87% of the original dataset's features. The research, thus, introduces an efficient Android malware detection method that reduces the computational cost and improves the classification accuracy.

KEYWORDS

Android, Malware, Dynamic Analysis, Machine Learning, Malware Category Classification.

1. INTRODUCTION

As Android devices have become practically a part of our lives, they have unfortunately attracted some unwanted guests. This may include, but are not limited to, a growing swarm of malicious software, or malware, targeting these devices. Android malware poses a serious threat to our privacy, communication, finances, and even how well our devices work (Xu et al., 2023). It constantly changes its tactics to sneak past our defenses (Vijay, Portillo-Dominguez and Ayala-Rivera, 2022). This is why it is important to accurately and quickly identify and categorize malwares. However, traditional ways of analyzing malware, which involves looking at its code, often have limitations because malware creators are growing better at hiding their tracks. To meet this challenge, researchers are turning to another approach called dynamic analysis. It is essential to investigate virus activity as a detective when observes a suspect (V et al., 2023). Even if the application attempts to conceal its genuine aims and how it operates, its true objectives can be discovered if the application is run in a secure and controlled environment (Cui et al., 2023).

The fight against malicious software for Android devices is comparable to a never-ending game of cat and mouse. Criminals in the cyber world are coming up with more ways to avoid being caught. Some of these ways may include encrypting their code, using techniques that stop analysis, and even changing the way the malware works (A. Mawgoud, Rady and Tawfik, 2021)(Hussain and Mohideen S, 2023).Criminals are also using machine-learning techniques to trick security systems (G, P and S, 2023). Overcoming the constraints of code analysis can provide a more accurate depiction of the characteristics of malicious software using several different approaches such as static, dynamic, and hybrid analysis. This work focuses on dynamic analysis as it observes the way a virus acts in real time (Le et al., 2020). This analysis is used to highlight and appear malware's masks and its true intentions, regardless of how hard the malware tries to hide. Using this method, a lot about its behavior can be observed such as what kind of calls it makes to the system, what it does on the network, and how it uses resources. Recently, deep learning (Mohammed, Kareem and Mohammed, 2022) and machine learning techniques have been widely adopted to look at the information gathered in order to successfully classify different types of malware (Xu, Zhang and Tang, 2023). Although data mining techniques achieved high accuracy in previous literature, their use is not without limitations.

Earlier research achieves high prediction accuracy with the utilization of a large number of features. For example, Islam et al. (Islam et al., 2023) obtained an overall accuracy of 95% with 45 features out of the initial pool of 141 features. Accordingly, this present study aims to reduce the number of features in predicting Android malware based on integrating feature reduction and selection techniques. To enhance classification accuracy, while minimizing feature dimensionality, an optimal balance between them is achieved. This can lead to ensuring that the chosen techniques not only streamline the dataset but also contribute in improving model performance.

This research adds several contributions in comparison with earlier literature. To introduce a systematic and operational strategy for the detection of Android malware categories, it focuses mainly on dynamic features to overcome the constraints associated with static analysis such as ignoring malware behavior and actual intention. Hence, the proposed approach addresses issues related to malware detection such as the incapacity in identifying Android malware applications that employed concealment techniques. Moreover, this research confirms the performance and efficiency of the proposed model through the utilization of machine-learning classifiers. Third, it compares the effectiveness of various machine learning techniques, including Decision Trees (DT), Random Forest (RF), and K-Nearest Neighbors (KNN). Finally, the research provides a comparative analysis with earlier literature, highlighting the distinctive contributions and advancements achieved in Android malware detection.

The reminder of this study is structured as follows. Section two introduces and reviews some of related studies. Section three covers the dataset analysis, data preprocessing methods, feature selection and reduction process, and the machine learning models that are implemented. Section four provides a comprehensive analysis of the experiment's outcomes and conducts a comparative analysis with relevant results from other studies. Section five shows the practical implications of this research, whereas Section six summarizes and concludes its key concepts.

2. RELATED WORK

Malware analysis techniques are typically categorized as static and dynamic analyses. This is based on the type of features that are used and how malware samples are processed. Here, recent studies that used either static or dynamic analysis are reviewed, focusing on the number of features included and accuracy achieved.

With static analysis, characteristics are extracted without the need to execute or install the sample. The detector's exceptional performance and rapidity are attributed to its unique characteristic of identifying samples belonging to well-known malware families. However, this approach exhibits suboptimal performance when confronted with samples that employ obfuscated technology. Within the domain of Android malware detection, static analysis is frequently employed to derive characteristic features of samples, such as code information, API calls, or permissions. A research study conducted in 2022 (Shatnawi, Yassen and Yateem,

2022) was based on Android permissions and API calls where three machine learning algorithms (Support Vector Machine, K-Nearest Neighbor, Naive Bayes) are used with 50 important features. The average accuracy rate achieved is approximately 94% using permission features and 83% according to API call features. Another study proposed the AdDroid framework that implemented Adaboost with a Decision Tree algorithm (Mehtab et al., 2020). The achieved classification accuracy was 99.11%. The limitation of AdDroid can be enhanced by using a larger dataset of both malicious and benign applications. Furthermore, Zhao et al. (Ma et al., 2019) developed a ML Approach for identifying Android malware by creating a control flow graph and extracting API data by constructing the control flow graphs (CFG) for each application to create three types of datasets: API usage dataset, API frequency dataset, and API sequence. An ensemble model based on application of C4.5, Deep Neural Network, and Long Short-Term Memory methods achieved an overall detection accuracy of 98.98%. A recent study presented a framework called DIDroid (Rahali et al., 2020). It is a deep neural network approach that utilizes images to classify twelve types of android malware and 191 malware families. The system achieved an accuracy rate of 93.36%.

Another category of research is directed into exploring dynamic features. This type of literature also incorporates machine learning methods. The integration of feature selection techniques appears as a strategic approach, offering the potential to yield more refined insights with a reduced set of features. This not only enhances the effectiveness of the model but also contributes to mitigating complexity, thereby optimizing the overall malware detection process Tiwari et al. (Tiwari and Shukla, 2018) proposed a machine learning model that extract permissions and API from classes.dex and generate two types of feature vector: common and combined feature vector. Both types classified with logistic regression which achieved 97.25% accuracy with common features and 95.87% accuracy for 131 feature of permissions. Similarly, Gronat et al. (Gronat, Aldana-Iuit and Balek, 2019) constructed MaxNet model on dataset consisting of 361.265 samples where the API calls and system calls were extracted from Android applications. Combining the LSTM with the recurrent neural network approach was done to enhance the model's performance with an overall accuracy of 96.2%. In 2019, the support vector classifier was utilized by the TFDroid framework (Lou et al., 2019) to identify malware using the source, sink, and application description of Android apps. The model divides applications into functional clusters and detects abnormal behavior in each cluster to identify potential malicious applications. Only benign applications were deployed to train the classification algorithm, which achieved 93.65% accuracy. Finally A machine learning ensemble model integrating Random Forest, Multi-level Perceptron, Decision Tree, K-Nearest

Neighbor, Support Vector Machine, and Logistic Regression algorithms was introduced (Islam et al., 2023) to categorize twelve types of malware using 45 features out of 141 in which the proposed model obtained an accuracy rate of 95%.

Based on the above discussed literature, it can be inferred that dynamic analysis offers key advantages in Android malware detection in comparison with static analysis. It reveals hidden intentions by identifying application behavior, not just its code, discovering obfuscated threats and analyzing real-time behavior such as suspicious network calls. For such reasons, this present research focuses on dynamic features in identifying Android malware. Previous research gained either high accuracy with a high number of features or low accuracy with a low number of features. This research, however, aims at achieving high accuracy with low number of features.

3. THE PROPOSED SYSTEM

Fig. 1 depicts the key phases of the proposed system, including data preprocessing, feature selection and reduction, classification, and results evaluation.

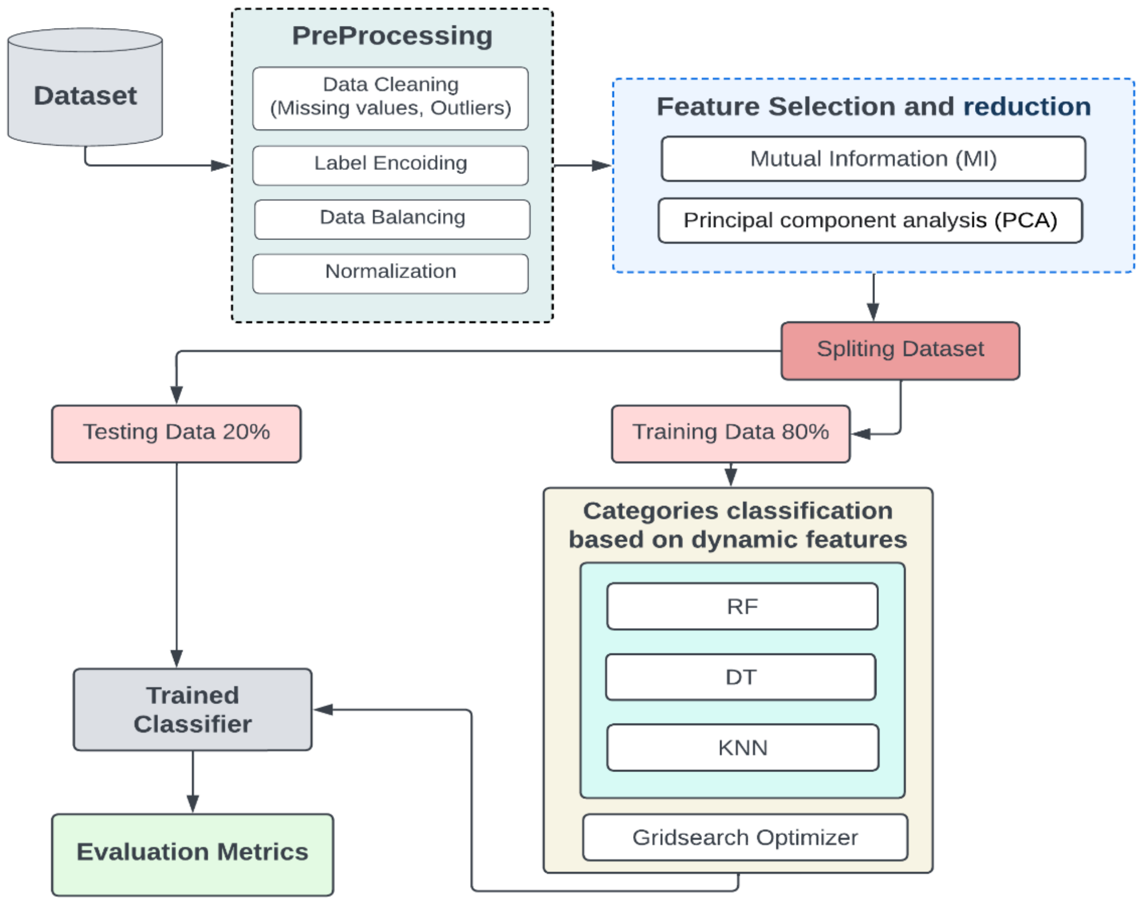


Fig. 1. The proposed system

3.1. The Research Dataset

The CCCS-CIC-AndMal-2020 Android malware dataset, which was jointly published by the University of New Brunswick (UNB) and the Canadian Institute for Cybersecurity (CIC), was used in this study (AndMal 2020 | Datasets | Research | Canadian Institute for Cybersecurity | UNB). It comprises 400,000 Android applications evenly distributed between benign (200,000) and malware (200,000) categories. The examination of applications for Android malware was conducted in a virtual environment. The dataset includes both static and dynamic features. Static elements consist of Android malware families, permissions, and intents. The dynamic features consist of detailed log files that contain process logs, package information, log states, battery states, and other pertinent data. The dataset includes fourteen categories of malware, such as adware, backdoors, file infectors, zero-day, Potentially Unwanted Apps (PUA), ransomware, riskware, scareware, Trojan, trojan-banker, trojan-dropper, trojan-sms, and trojan-spy.

The primary emphasis of this research revolves around the dynamic analysis segment of the dataset, which is partitioned into two distinct components. First, the initial dynamic analysis of Android malware, which encompasses 28,380 Android malware applications before the reboot of an Android emulator, was applied. Second, the dynamic analysis of Android malware consists of 25,059 applications were executed after restarting the Android emulator. This research merges the two parts of dynamic analysis and then splits the resulting dataset into 80% training and 20% testing. Based on different experiments conducted in this research, it was found that the optimum strategy is to allocate 20-30% of the data for testing and the remaining 70-80% for training to achieve the best outcomes (Gholamy, Kreinovich and Kosheleva, 2018).

3.2. Data preprocessing

Data preprocessing plays a crucial role in studies involving machine learning-based classification. Datasets often contain missing values, outliers, or extraneous features. All of which can significantly influence the performance of machine learning algorithms. In this research, firm and precise steps were followed to ensure data cleaning. First, the categories of the dataset were replaced with numeric values from 1 to 14. Furthermore, a check of missing values was conducted, showing that no missing values in the dataset. Additionally, a Boolean check was conducted to discover data duplication where no duplication was identified. Fourth, the random oversampling method was used to balance all the fourteen categories of malware as shown in Fig.2 which is a technique used to address class imbalance in machine learning datasets by increasing the number of instances of the minority class. The reason of choosing oversampling technique is to improve model performance for the minority classes and pay attention to the other thirteen labels, if most of the data belongs to the majority class, the model will prioritize learning to classify those instances accurately and this, in turn, could lead to untrusted and bias classification. The oversampling technique, however, is not without limitations. It raises the possibility of over-fitting by duplicating minority class instances (Chawla et al., 2002) .This drawback was addressed in this research by applying ensemble classifiers such as the Random Forest algorithm. It is strong against overfitting. because Random Forest computes the average of the predictions from many trees. Each tree with some variation due to randomness in training and this, in turn, can lead to reducing the variance and avoiding the memorization of just the training data.

The outliers are also checked, showing that the dataset contains numerous outliers which can be defined as observations that exhibit significant deviations from the rest of the values in a population-based random sample. For this reason, a capping method was used to handle the outliers. This method calculates the Inter Quartile Range (IQR) (Frery, 2023) to determine data variation. Outliers are values outside the (25th percentile –1.5x Inter Quartile Range) to (75th percentile + 1.5x Inter Quartile Range). Outlier values were then replaced with the calculated maximum or minimum capped values. Fig.3 demonstrates an example of some features before and after handling outliers. Finally, a StandardScaler normalization was used which is a method employed in data preprocessing to transform and scale features within a dataset. It follows the standardization approach, which involves rescaling the features to have a total mean of zero and a standard deviation of one. The rationale behind the application of this technique is to prevent bias towards features with larger scales. Equation 1 is used to calculate standard scaling(Gopal, Patro and Kumar Sahu, 2015). This normalization technique is particularly useful when dealing with features that exhibit varying scales, ensuring that each feature contributes equally to the model's learning process.

$$Z = \frac{(x - \mu)}{\sigma} \qquad (1)$$

where:

Z is the value that has been standardized.

X is the initial value of the feature.

μ is the average value of the characteristic.

σ is the measure of variability or dispersion of the property.

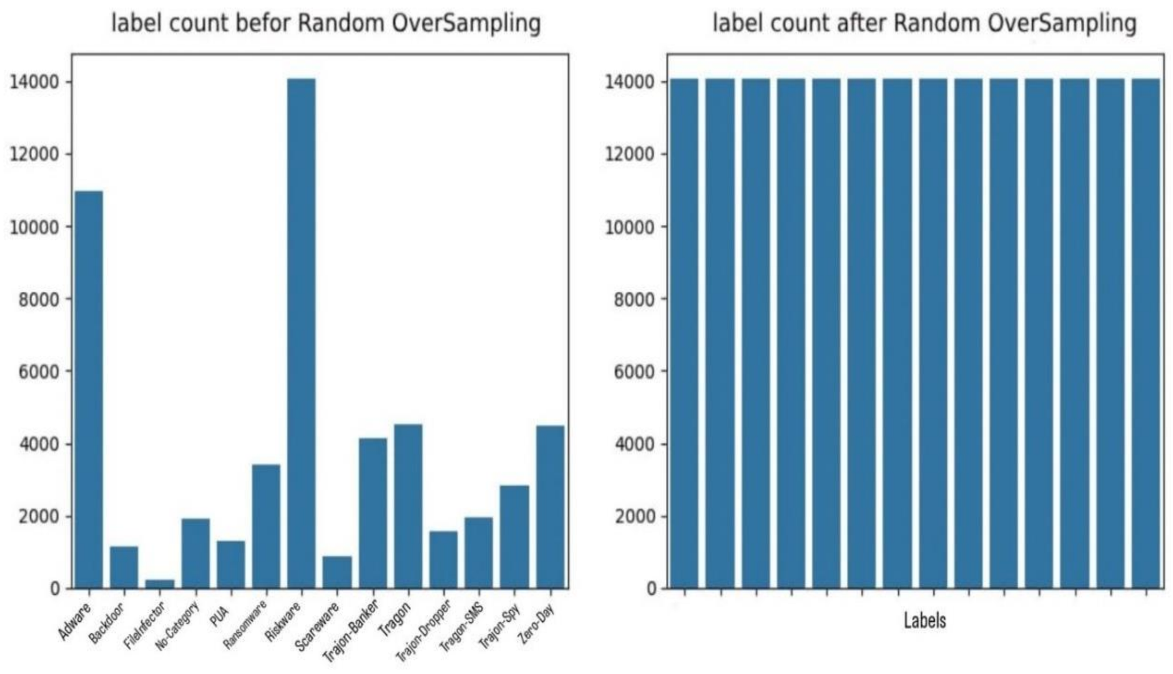


Fig. 2. The application of oversampling method for data balancing

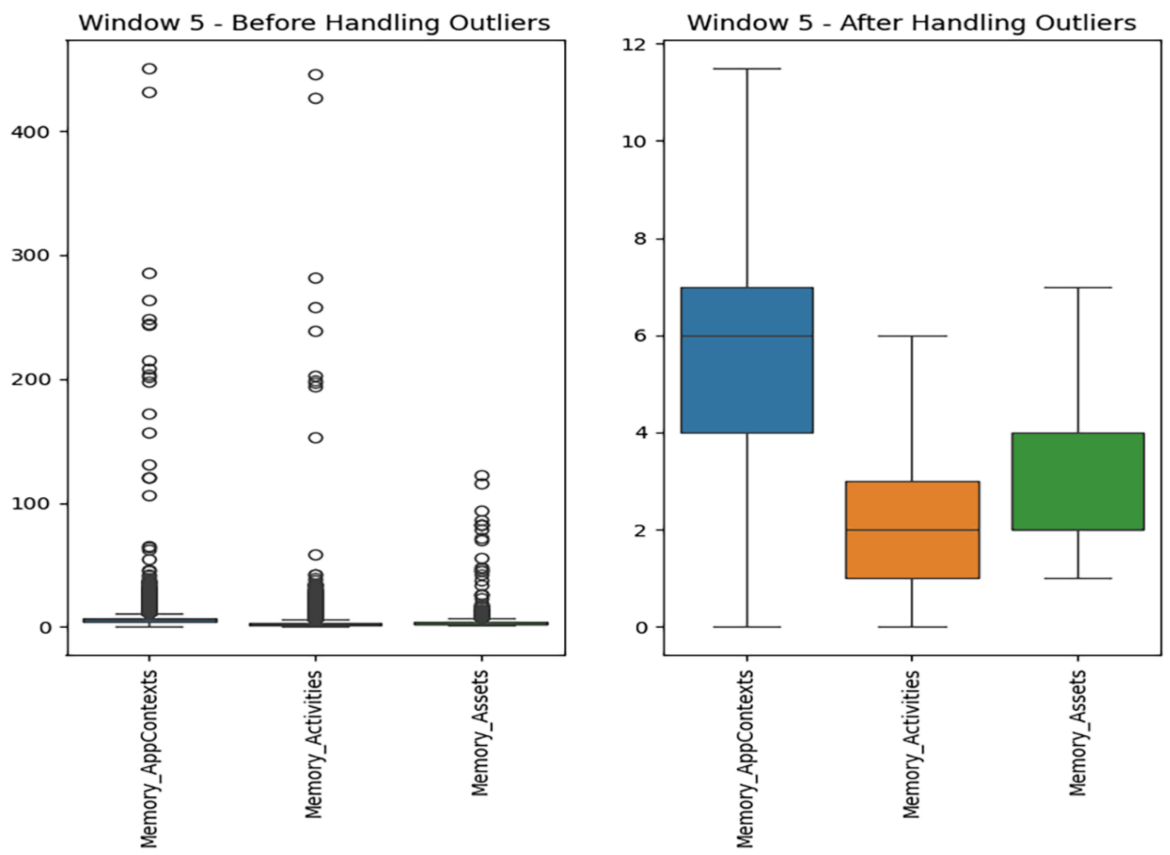


Fig. 3. An example of features before and after handling outliers

3.3. Feature selection

Feature selection includes choosing a portion of the most important features from a dataset with the objectives of reducing dimensionality and computational cost, improving model interpretability, and enhancing predictive performance. In the present study, the Mutual Information method was employed (Zhou, Wang and Zhu, 2022), resulting in the selection of only 50 features out of the original 142 features as shown in Fig.4. The reason behind the selection of this method includes its capacity to capture complex interactions between features and the target variable. Mutual Information outperforms approaches such as correlation that are limited to linear relationships, especially in complicated real-world datasets (Battiti, 1994)The choice of this specific number was determined through a series of experiments, striking a balance between achieving optimal accuracy and minimizing the number of features incorporated in the model. Equation 2 estimates the amount of information shared by two variables, X and Y (Liu et al., 2009).

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} P(X,Y) LOG \frac{P(X,Y)}{P(X)P(Y)} \quad (2)$$

where:

X, Y: random variables.

P(x) is the probability density function of X.

P(Y) is the probability density function of Y.

According to this definition, if X and Y are strongly related to one another, the value of I(X; Y) will be quite high, whereas if I(X; Y) = 0, this indicates that these two variables are completely unrelated.

3.4. Dimension reduction

In this study, the Principal Component Analysis (PCA) was applied to perform dimensionality reduction on the dataset. PCA is a statistical approach used to decrease the number of dimensions in datasets. The process discovers and ranks the primary characteristics, known as main components, by capturing the highest amount of variation in the data. The linear combinations of the original features provide a reduced-dimensional representation of the data while preserving maximum information (Li et al., 2020). The number of principal components that were selected based on this technique was only thirty three features. This helps reduce the number of features from fifty features obtained based on the Mature Information feature selection approach to only thirty three features and this, in turn, leads to decreasing the computational cost.

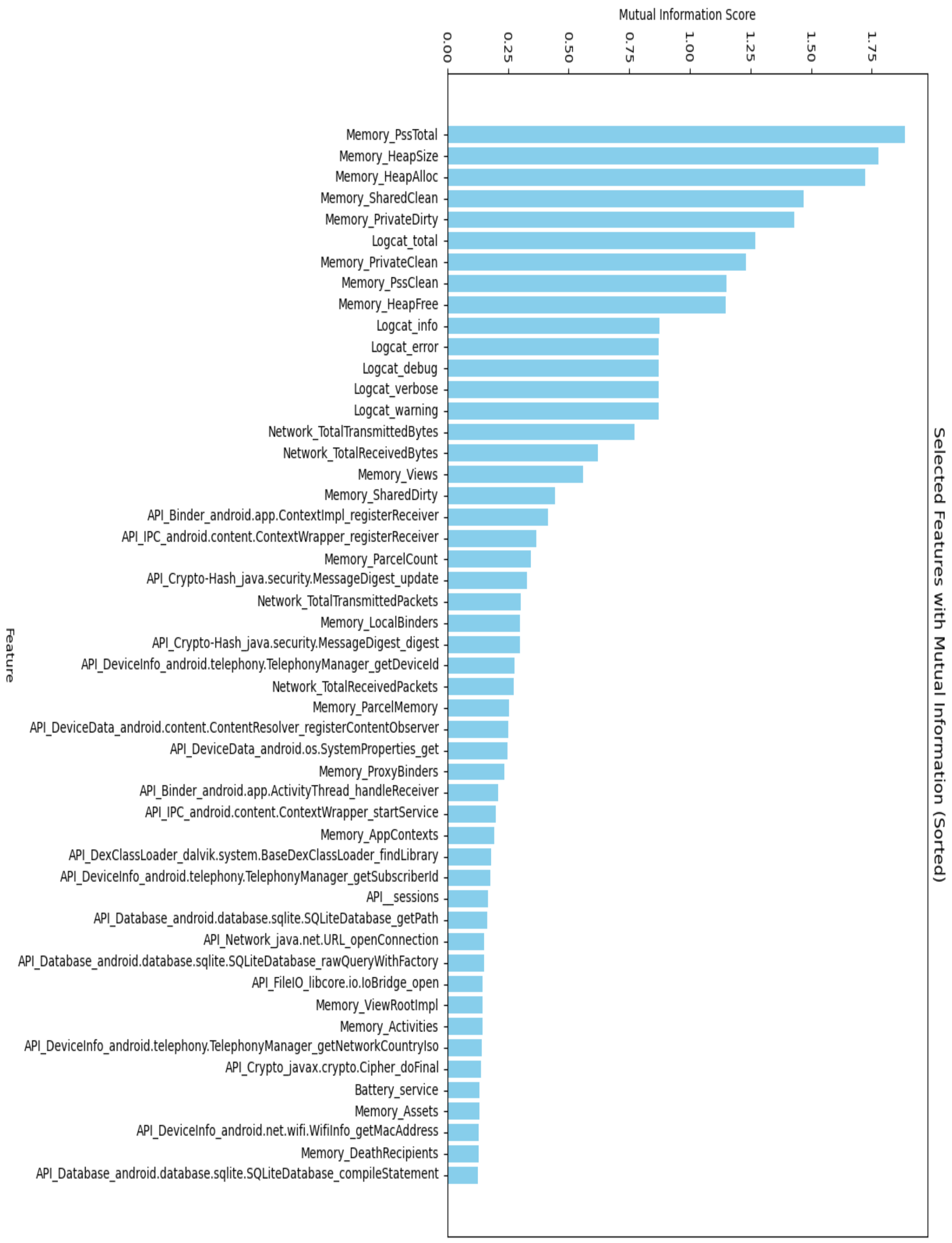


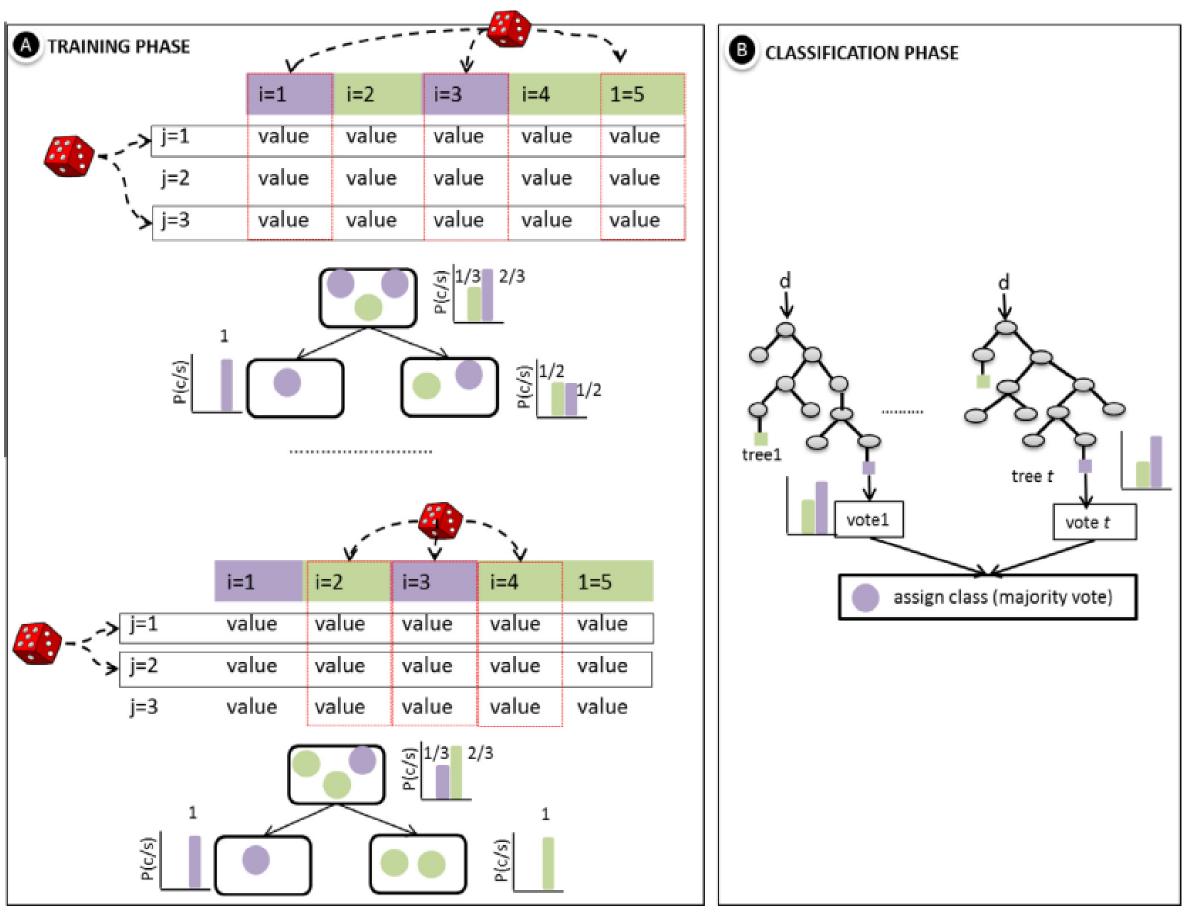
Fig. 4. The top fifty important features selected by the Mutual Information method

3.5. Classification techniques

In this study, three machine learning algorithms are applied to evaluate and compare the effectiveness of the Android malware detection namely, Random forest, Decision tree and K-Nearest Neighbors.

3.5.1. Random Forest (RF)

Random Forest is one of the most popular ensemble machine-learning techniques (Dudek, 2022). It can achieve accurate and reliable predictions by utilizing the collective knowledge. Every tree in Random Forest explores the data from a different perspective which is shaped by bootstrapped training sets and randomized feature selection (Rajendiran and Rethnaraj, 2023). The ensemble's strength is powered by its inherent diversity. Each tree gives a "vote" for the most likely result when produced a prediction. For regression and classification, the average or majority wins out. By reducing the biases of individual trees, this democratic method produces resilient and broadly applicable results. Because of its versatility, simplicity, and ability to solve both regression and classification problems, Random Forest is used in a wide range of fields, including environmental science, healthcare, marketing, finance, and environmental science (Breiman, 2001). Fig.5 illustrates the key principles of the Random mechanism (Belgiu and Drăguț, 2016).



Training and classification phases of Random Forest classifier: *i* = samples, *j* = variables, *p* = probability, c = class, s = data, *t* = number of trees, d = new data to be classified, and value = the different values that the variable *j* can have.

Fig. 5. The Random Forest mechanism

3.5.2. Decision Trees (DT)

Decision Trees are tree-like graphs with core nodes representing attribute tests, branches indicating test outcomes, and leaf nodes indicating class labels. Classification rules are based on the path from the root node to the leaf. The root node is selected as the primary property for dividing input data. Assigning properties and values to each intermediary node of the tree allows for input data analysis (Tan, Steinbach and Kumar, 2014). After forming a tree, it may anticipate new data by traveling from root to leaf nodes, visiting internal nodes based on attribute test requirements.

3.5.3. K-Nearest Neighbors (KNN)

The K-Nearest Neighbors method is a widely used and simple machine learning technique that classifies real-world scenarios by identifying the closest neighbors. The K-Nearest Neighbors algorithm calculates the distance between normal values and attacks, then selects object values that are close to the k-values of the class. The process starts by acquiring network data through the input data (Liu et al., 2022). The Euclidean distance function (Ei) was employed in this study to calculate the distance between the values of objects. The Euclidean distance function can be expressed as in Equation 3.

$$\mathrm{Ei} = \sqrt{(\mathrm{a1} - \mathrm{a2})^2 + \mathrm{b}(\mathrm{b1} - \mathrm{b2})^2} \tag{3}$$

where Ei is the sum of the squared difference between (a1 - a2) and (b1 - b2), where a1, a2, b1, and b2 are variables in the input data.

3.6. The Optimization Techniques

In this research, the Grid Search Optimizer is used to fine-tune the hyperparameters of the data mining techniques. It is a popular technique for figuring out a classification model's proper hyperparameters. If there are enough grid nodes, it might be able to arrive at the optimal solution (Erdogan Erten, Bozkurt Keser and Yavuz, 2021). In this study, the three algorithms used to choose the ideal hyperparameters are integrated with the grid search-based optimization. In the grid search optimizer approach, the dataset is randomly divided into test and training sets using cross-validation. Optimizing hyperparameters can greatly enhance prediction accuracy, but may also increase time complexity. However, since this process is conducted offline, the additional computational cost is generally manageable. Its mechanism can be explained in Algorithm 1.

```
Algorithm 1: Grid Search Optimizer
Input: List of classifier's parameters.
Output: Return best_paramameters.
Step 1: Initializing best score = -inf (negative infinity) and best_paramameter = None.
Step 2: while parameters list  != null
          Begin
                Create a clone of the model with the current hyperparameter combination.
                Train the cloned model on your data.
                Evaluate the trained model using evaluation metric.
                If (current score > best_score):
                   Update best_score with the current score.
                   Update best_params with the current hyperparameter combination.
          End
Step 3: Return best_paramameter.
```

3.7. Performance Evaluation

It is crucial to evaluate the expected accuracy of algorithms utilizing machine learning methods for classification. When constructing a model, the data is separated into a training and evaluation sets. The learning set is utilized to construct the model, while the evaluation set is employed to assess its accuracy. Precision, recall, accuracy, and F1-score are metrics utilized to evaluate the pattern identification abilities of machine learning techniques. These metrics can be defined in four different scenarios (M and M.N, 2015). Accuracy is defined in Equation 4. It speaks of the percentage of correct predictions generated from the test data. It can be easily calculated by dividing the total number of predictions by the number of accurate forecasts. However, the accuracy Paradox causes problems with predictive analysis. Because of this, it may be confirmed using Precision and Recall indicators, which determine whether the "Negative" ratio of the real data provides an accurate classification of situations that are unlikely to occur. Precision is defined by Equation 5. The percentage of accurate positive results to those that the classifier predicted to be positive is how it is computed. Positive Predictive Value is another term for accuracy (PPV). Equation 6 computes the recall by dividing the total number of true positive findings by the total number of applicable samples. The Recall is also known as the True Positive Rate (TPR). Recall and precision serve as indicators of divergent concepts.

Verifying the F1 Score, the Harmonic Mean of Precision, and Recall could improve the correctness of the model. The F1 Score is established by Equation 7. To assess how well the model is performing, harmonic means are utilized to balance both the Precision and Recall indicators when one of them is low and nearly zero. Moreover, the "Confusion Matrix" approach was utilized to assess the efficacy of training-based methods by comparing what was expected to the real ones (Breiman, 2001). Assessment metrics evaluate the relationship between the model's responses and the real replies.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}} \qquad (4)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \qquad (5)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \qquad (6)$$

$$\text{F1} = 2\,\frac{\text{Precision+Recall}}{\text{Precision}*\text{Recall}} \qquad (7)$$

4. RESULTS AND DISCUSSION

This research aims to identify Android malware based on dynamic features. However, several different improvements are achieved to obtain the best classification accuracy. This includes dimensionality reduction, feature selection, and optimization. First, the accuracy is obtained based on the original 142 features of the dataset as a baseline. The only preprocessing step that was implemented with the baseline is handling the missing values. The findings are summarized in Table 1. It can be noticed that the accuracy for the Random Forest classifier is the highest in comparison with the other two classifiers namely, K-Nearest Neighbor and Decision Tree.

Table 1: Accuracy of the baseline implementation with the 142 features

Classifier	Accuracy	Precision	Recall	F1-score	Handling outliers	Normalization
RF	0.83	0.83	0.83	0.83	X	X
DT	0.74	0.74	0.74	0.74	X	X
KNN	0.60	0.59	0.60	0.59	X	X

Table 2 reports the accuracy after applying preprocessing techniques include such as data balancing, handling outliers and performing features' normalization. Furthermore, the Mutual Information selection algorithm was applied with PCA. Only the top fifty features were selected by Mutual Information based on their weight which became the input for the PCA algorithm that reduced the number of features to only thirty three features which are illustrated in Fig.6. The PCA results suggest that the Memory, API and Logcat features are the most effective categories. This is because they have the highest representation among the selected components. Memory features in particular with 14 out of 33 features being from this category appear to be critical, potentially due to their ability to capture various aspects of system performance and behavior.

The next most significant category of features is API features. It is shown by the results that nine out of 33 features were chosen, with specific API calls and cryptographic operations. Logcat features also play a vital role which were represented with only six features. This can capture different types of system messages. Moreover, only four network features were chosen, reflecting the role of data transmission and reception in classifying the data. The absence of Battery and Process features in the selected components suggests that these categories might not provide substantial discriminatory power for the classification tasks in this dataset.

Table 2: Accuracy with the 33 features after applying preprocessing and feature selection techniques

Classifier	Accuracy	Precision	Recall	F1-score	Handling outliers	Normalization
RF	0.97	0.97	0.97	0.97	✓	✓
DT	0.95	0.95	0.95	0.95	✓	✓
KNN	0.90	0.90	0.90	0.90	✓	✓

It is obvious that the accuracy of all classifiers was improved after applying the preprocessing stage. Although preprocessing might increase the model complexity, it can help reduce the number of features fed into the classification algorithms and this, in turn, can lead to reducing complexity and computational cost. Since the model was operated offline, the accuracy-complexity tradeoff prioritizes the accuracy over the complexity.

Table 3 includes the findings of the classifiers of thirty three features after using the Mutual Information feature selection with PCA and grid search optimizer which was also visualized in Fig. 7. For the Random Forest model, the best accuracy was found at 100 n_estimators. For the Decision Tree min_samples_split was set to 2 and max_depth: None. For k-nearest neighbor, n_neighbors were set to 3 and 'weights' to ‘distance’. Furthermore, by comparing expected and actual values, the "Confusion Matrix" was created for each classifier to assess the efficacy of training-based prediction for the fourteen malware categories.

Table 3: A comparison of accuracy after using the Grid Search Optimizer

Classifier	Accuracy	Precision	Recall	F1-score
RF	0.98	0.98	0.98	0.97
DT	0.95	0.95	0.95	0.95
KNN	0.96	0.96	0.96	0.96

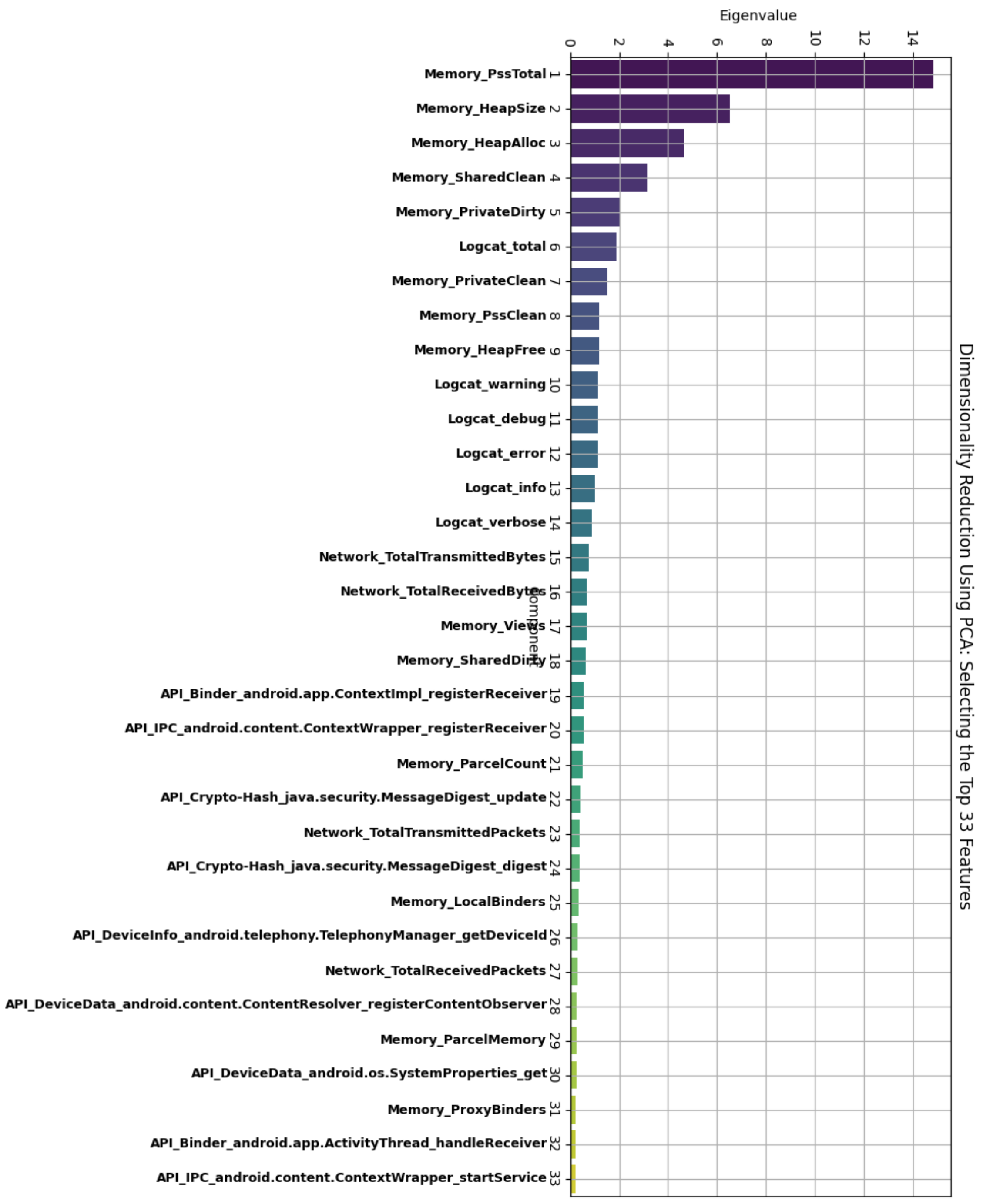
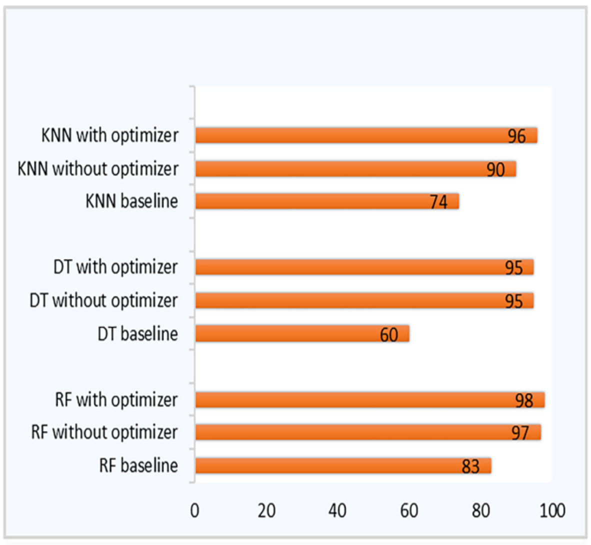


Fig. 6. The selected features using the PCA method

Pertaining to the use of Random Forest as it achieves the highest accuracy, the total of the diagonal elements in the confusion matrix shows that 38364 (97.50%) malware samples are correctly categorized into their appropriate categories. while only 982 (2.50%) samples are incorrectly identified. By considering the categories backdoor, file infector, PUA, scareware and Trojan-Dropper which are labelled 2, 3, 5, 8 and 11 respectively as shown in Fig.8, it can be seen that all samples are 100% accurately categorized. On the other hand, the investigation of the Riskware category (label 7) shows that 546 samples out of 2854 were incorrectly identified, which is the highest false identification that reduced the overall classification accuracy of the proposed model. The potential cause of this false classification is a significant variation in sample sizes between categories of the dataset, which requires using data balancing techniques. The random oversampling method is applied to avoid bias towards the largest label, despite raising the overfitting level to an acceptable level. Regardless of the model's remarkable accuracy, the prediction still contains a number of false positive and false negative samples. This is demonstrated in Fig. 8.



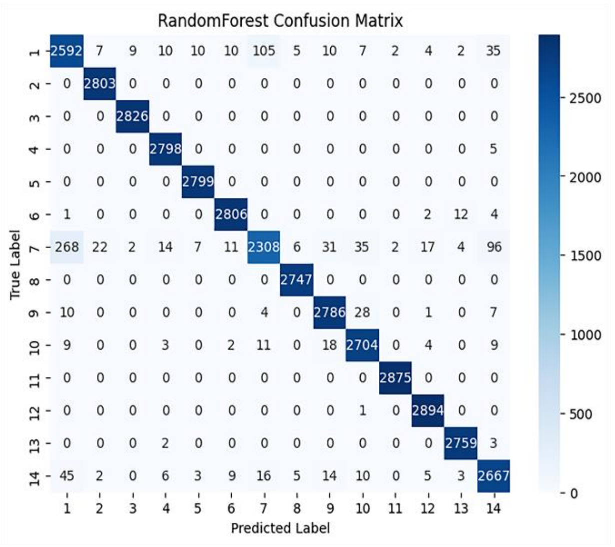


Fig. 7. Accuracy of classifiers **Fig. 8. The RF Confusion Matrix**

Table 4 shows a comparison between the output findings of this study and other previous works. Only literature that used dynamic approach are included in this comparison. The majority of the studies employed either a machine learning or deep learning techniques. The selected studies in this comparison are similar to this present research in terms of dataset and number of predicted categories. Although some of the earlier literature predicted twelve categories only based on the notion that not all categories information are available in the dataset (Islam et al., 2023),(Rahali et al., 2020), this study focused on identifying all malware categories in the CCCS-CIC-AndMal-2020 dataset because categories that have not all information are for static features only as mentioned in the dataset original website (AndMal 2020 | Datasets | Research | Canadian Institute for Cybersecurity | UNB). As shown in Table 4, the highest accuracy achieved was 95% in (Islam et al., 2023), while our suggested model outperforms their findings by achieving 97.52% of accuracy for the fourteen categories. The reasons behind such achieved accuracy can be attributed to 1) the combination of both before and after reboot files in one data frame which makes the classifiers learn more about malware behavior in training phase, resulting in improving classification accuracy, 2) an efficient preprocessing steps were followed to prepare the dataset for classification phase which helped improve the model performance, and 3) the use of two stages of feature reduction led to shrinking the feature vector from 142 features to only thirty three important features.

Table 4: A comparison between the findings of this research and previous literature

Reference	Analysis	Dataset	No. of Classes	Year	Algorithm	Accuracy
(Musikawan et al., 2023)	Dynamic	CCCS-CIC-AndMal-2020	14	2022	DNN	78.82
	Static	CCCS-CIC-AndMal-2020	2		DNN	97.72
(Rahali et al., 2020)	Static	CCCS-CIC-AndMal-2020	12	2020	SSD	93.36
(Islam et al., 2023)	Dynamic	CCCS-CIC-AndMal-2020	12	2023	Ensemble ML	95
This study	Without a Grid search optimizer	Dynamic CCCS-CIC-AndMal-2020	14	2024	RF	97.48
					DT	95.40
					KNN	90.27
	With Grid search optimizer	Dynamic CCCS-CIC-AndMal-2020	14		RF	97.50
					DT	95.48
					KNN	96.25

5. PRACTICAL IMPLICATIONS

This study improved prediction accuracy and reduced implementation complexity. This was done by applying grid search optimization, feature selection, feature reduction, feature normalization, and missing and outlier handling. It supports multi-class characterization on huge dataset and exceeded prior studies with fewer characteristics, even though it was grounded on past research. The study has many practical implications for people, organizations, and academics. Better business security protects important data and mobile workers. Security applications and antivirus software should provide better detection algorithms that can protect users from risky apps and activities.

6. CONCLUSIONS

This research introduced a machine learning model leveraging Random Forest, Decision Tree, and K-Nearest Neighbors algorithms for the purpose of classifying fourteen prominent malware categories within a recently published dataset titled CCCS-CIC-AndMal-2020. It encompasses a comprehensive variety of safe and harmful applications. The study followed a data balancing mechanism and a clearly established data preparation method which is appropriate for various dynamic analysis features. A robust outlier handling process, coupled with mutual information feature selection and PCA, resulted in the elimination of 76.8% of features. Consequently, only thirty three features out of the initial 142 were incorporated into the model. This reduction in feature set not only addresses issues of complexity but also contributes to improved accuracy. The final accuracy achieved by the model is 97.50%.

Although this study adds significant contributions in comparison with earlier literature, it is not without limitations that may invite further research. The dataset used in the study, CCCS-CIC-AndMal-2020, had some limitations and class imbalance. Future work can involve the use of more diverse and balanced datasets to evaluate the performance of the ensemble model. This can extend the scope of evaluation beyond the current dataset. Furthermore, this study focused on dynamic feature analysis for multi-classification. However, exploring the effectiveness of both static and dynamic feature analysis may help improve the model's performance. Finally, the use of another efficient data balancing method may lead to avoiding the overfitting limitation.

7. REFERENCES

A. Mawgoud, A., Rady, H.M. and Tawfik, B.S. (2021) ‘A Malware Obfuscation AI Technique to Evade Antivirus Detection in Counter Forensic Domain’, in A.-E. Hassanien, M.H.N. Taha, and N.E.M. Khalifa (eds) Enabling AI Applications in Data Science. Cham: Springer International Publishing, pp. 597–615. Available at: https://doi.org/10.1007/978-3-030-52067-0_27.

AndMal 2020 | Datasets | Research | Canadian Institute for Cybersecurity | UNB (2020). Available at: https://www.unb.ca/cic/datasets/andmal2020.html (Accessed: 7 February 2024).

Battiti, R. (1994) ‘Using mutual information for selecting features in supervised neural net learning’, IEEE Transactions on Neural Networks, 5(4), pp. 537–550. Available at: https://doi.org/10.1109/72.298224.

Belgiu, M. and Drăguţ, L. (2016) ‘Random forest in remote sensing: A review of applications and future directions’, ISPRS Journal of Photogrammetry and Remote Sensing, 114, pp. 24–31. Available at: https://doi.org/10.1016/j.isprsjprs.2016.01.011.

Breiman, L. (2001) ‘Random Forests’, Machine Learning, 45(1), pp. 5–32. Available at: https://doi.org/10.1023/A:1010933404324.

Chawla, N. V. et al. (2002) ‘SMOTE: Synthetic Minority Over-sampling Technique’, Journal of Artificial Intelligence Research, 16, pp. 321–357. Available at: https://doi.org/10.1613/jair.953.

Cui, J. et al. (2023) ‘Malware behavior detection method based on reinforcement learning’, in V. Varadarajan, J.C.-W. Lin, and P. Lorenz (eds) International Conference on Computer Application and Information Security (ICCAIS 2022). SPIE, p. 36. Available at: https://doi.org/10.1117/12.2671736.

Dudek, G. (2022) ‘A Comprehensive Study of Random Forest for Short-Term Load Forecasting’, Energies, 15(20), p. 7547. Available at: https://doi.org/10.3390/en15207547.

Erdogan Erten, G., Bozkurt Keser, S. and Yavuz, M. (2021) ‘Grid Search Optimised Artificial Neural Network for Open Stope Stability Prediction’, International Journal of Mining, Reclamation and Environment, 35(8), pp. 600–617. Available at: https://doi.org/10.1080/17480930.2021.1899404.

Frery, A.C. (2023) ‘Interquartile Range’, in B.S. Daya Sagar et al. (eds) Encyclopedia of Mathematical Geosciences. Cham: Springer International Publishing, pp. 664–666. Available at: https://doi.org/10.1007/978-3-030-85040-1_165.

G, R., P, V. and S, A. (2023) ‘Evading Machine-Learning-Based Android Malware Detector for IoT Devices’, IEEE Systems Journal, 17(2), pp. 2745–2755. Available at: https://doi.org/10.1109/JSYST.2022.3215014.

Gholamy, A., Kreinovich, V. and Kosheleva, O. (2018) ‘Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation’, Departmental Technical Reports (CS) [Preprint]. Available at: https://scholarworks.utep.edu/cs_techrep/1209 (Accessed: 17 March 2024).

Gopal, S., Patro, K. and Kumar Sahu, K. (2015) ‘Normalization: A Preprocessing Stage’, IARJSET, pp. 20–22. Available at: https://doi.org/10.17148/iarjset.2015.2305.

Gronat, P., Aldana-Iuit, J.A. and Balek, M. (2019) ‘MaxNet: Neural Network Architecture for Continuous Detection of Malicious Activity’, in 2019 IEEE Security and Privacy Workshops (SPW). IEEE, pp. 28–35. Available at: https://doi.org/10.1109/SPW.2019.00018.

Hussain, S. and Mohideen S, P. (2023) ‘Advanced Machine Learning Approach for Suspicious Coded Message Detection using Enigma Cipher’, in 2023 Second International Conference on Electronics and Renewable Systems (ICEARS). IEEE, pp. 800–803. Available at: https://doi.org/10.1109/ICEARS56392.2023.10085339.

Islam, R. et al. (2023) 'Android malware classification using optimum feature selection and ensemble machine learning', Internet of Things and Cyber-Physical Systems, 3, pp. 100–111. Available at: https://doi.org/10.1016/j.iotcps.2023.03.001.

Le, N.C. et al. (2020) 'A Machine Learning Approach for Real Time Android Malware Detection', in 2020 RIVF International Conference on Computing and Communication Technologies (RIVF). IEEE, pp. 1–6. Available at: https://doi.org/10.1109/RIVF48685.2020.9140771.

Li, L. et al. (2020) 'Comprehensive evaluation of robotic global performance based on modified principal component analysis', International Journal of Advanced Robotic Systems, 17(4), p. 172988141989688. Available at: https://doi.org/10.1177/1729881419896881.

Liu, Gaoyuan et al. (2022) 'An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs', Sensors, 22(4), p. 1407. Available at: https://doi.org/10.3390/s22041407.

Liu, H. et al. (2009) 'Feature selection with dynamic mutual information', Pattern Recognition, 42(7), pp. 1330–1339. Available at: https://doi.org/10.1016/j.patcog.2008.10.028.

Lou, S. et al. (2019) 'TFDroid: Android Malware Detection by Topics and Sensitive Data Flows Using Machine Learning Techniques', in 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT). IEEE, pp. 30–36. Available at: https://doi.org/10.1109/INFOCT.2019.8711179.

M, H. and M.N, S. (2015) 'A Review on Evaluation Metrics for Data Classification Evaluations', International Journal of Data Mining & Knowledge Management Process, 5(2), pp. 01–11. Available at: https://doi.org/10.5121/ijdkp.2015.5201.

Ma, Z. et al. (2019) 'A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms', IEEE Access, 7, pp. 21235–21245. Available at: https://doi.org/10.1109/ACCESS.2019.2896003.

Mehtab, A. et al. (2020) 'AdDroid: Rule-Based Machine Learning Framework for Android Malware Analysis', Mobile Networks and Applications, 25(1), pp. 180–192. Available at: https://doi.org/10.1007/s11036-019-01248-0.

Mohammed, H.A., Kareem, S.W. and Mohammed, A.S. (2022) 'A COMPARATIVE EVALUATION OF DEEP LEARNING METHODS IN DIGITAL IMAGE CLASSIFICATION', Kufa Journal of Engineering, 13(4), pp. 53–69. Available at: https://doi.org/10.30572/2018/KJE/130405.

Musikawan, P. et al. (2023) ‘An Enhanced Deep Learning Neural Network for the Detection and Identification of Android Malware’, IEEE Internet of Things Journal, 10(10), pp. 8560–8577. Available at: https://doi.org/10.1109/JIOT.2022.3194881.

Rahali, A. et al. (2020) ‘DIDroid: Android malware classification and characterization using deep image learning’, ACM International Conference Proceeding Series, pp. 70–82. Available at: https://doi.org/10.1145/3442520.3442522.

Rajendiran, G. and Rethnaraj, J. (2023) ‘Lettuce Crop Yield Prediction Analysis using Random Forest Regression Machine Learning Model in Aeroponics System’, in 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS). IEEE, pp. 565–572. Available at: https://doi.org/10.1109/ICAISS58487.2023.10250535.

Shatnawi, A.S., Yassen, Q. and Yateem, A. (2022) ‘An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms’, in Procedia Computer Science. Elsevier B.V., pp. 653–658. Available at: https://doi.org/10.1016/j.procs.2022.03.086.

Tan, P.-N., Steinbach, M. and Kumar, V. (2014) Introduction to data mining. First Edition. Pearson (Intl).

Tiwari, S.R. and Shukla, R.U. (2018) ‘An Android Malware Detection Technique Based on Optimized Permissions and API’, in 2018 International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, pp. 258–263. Available at: https://doi.org/10.1109/ICIRCA.2018.8597225.

V, A. et al. (2023) ‘Malware Detection using Dynamic Analysis’, in 2023 International Conference on Advances in Intelligent Computing and Applications (AICAPS). IEEE, pp. 1–6. Available at: https://doi.org/10.1109/AICAPS57044.2023.10074588.

Vijay, A., Portillo-Dominguez, A.O. and Ayala-Rivera, V. (2022) ‘Android-based Smartphone Malware Exploit Prevention Using a Machine Learning-based Runtime Detection System’, in 2022 10th International Conference in Software Engineering Research and Innovation (CONISOFT). IEEE, pp. 131–139. Available at: https://doi.org/10.1109/CONISOFT55708.2022.00026.

Xu, L., Zhang, C. and Tang, K. (2023) ‘A malware analysis method based on behavioral knowledge graph’, in Y. Yue (ed.) International Conference on Electronic Information Engineering and Computer Science (EIECS 2022). SPIE, p. 71. Available at: https://doi.org/10.1117/12.2668119.

Xu, Q. et al. (2023) ‘Android Malware Detection Based on Behavioral-Level Features with Graph Convolutional Networks’, Electronics, 12(23), p. 4817. Available at: https://doi.org/10.3390/electronics12234817.

Zhou, H., Wang, X. and Zhu, R. (2022) ‘Feature selection based on mutual information with correlation coefficient’, Applied Intelligence, 52(5), pp. 5457–5474. Available at: https://doi.org/10.1007/s10489-021-02524-x.