World Wide Web (2024) 27:50
https://doi.org/10.1007/s11280-024-01287-y

FSSDroid: Feature subset selection for Android malware detection

Nikolaos Polatidis[1] · Stelios Kapetanakis[2,4] · Marcello Trovati[3] · Ioannis Korkontzelos[3] · Yannis Manolopoulos[5]

Received: 6 April 2024 / Revised: 27 June 2024 / Accepted: 2 July 2024

Abstract
Android malware has become an increasingly important threat to individuals, organizations, and society, posing significant risks to data security, privacy, and infrastructure. As malware evolves in sophistication and complexity, the detection and mitigation of these malicious software instances have become more challenging and time consuming since the required number of features to identify potential malware can be very high. To address this issue, we have developed an effective feature selection methodology for malware detection in Android. The critical concern in the field of malware detection is the complexity of algorithms and the use of features that are used to detect malware. The present paper delivers a methodology for pre-processing datasets to select the most optimal features that will allow detecting malware, while maintaining very high accuracy. The proposed methodology has been tested on two real world datasets and the results indicate that the number of features is significantly reduced from 489 to between 19 and 28 for the first dataset and from 9503 to between 9 and 27 for the second dataset, whilst the accuracy is maintained as if all features were used.

Keywords Android · Malware detection · Feature selection · Machine learning · Binarization · Pre-processing

✉ Yannis Manolopoulos
Yannis.Manolopoulos@ouc.ac.cy

Nikolaos Polatidis
N.Polatidis@brighton.ac.uk

Stelios Kapetanakis
Stelios@dstr.co.uk

Marcello Trovati
Marcello.Trovati@edgehill.ac.uk

Ioannis Korkontzelos
Yannis.Korkontzelos@edgehill.ac.uk

1 University of Brighton, Brighton BN2 4GJ, UK

2 Distributed Analytics Solutions, London E14 6FD, UK

3 Edge Hill University, Ormskirk L39 4QP, UK

4 Middlesex University, London NW4 4BT, UK

5 Open University of Cyprus, Nicosia 2220, Cyprus

Published online: 16 July 2024

1 Introduction

The pervasive nature of malware poses a substantial threat to digital ecosystems worldwide. Malicious software, commonly known as malware, encompasses a broad spectrum of threats, including viruses, worms, trojans, ransomware, spyware, and others. The spread of malware attacks has escalated exponentially in recent years, resulting in severe financial losses, compromised privacy, and disrupted operations for both individuals and organizations. Consequently, detecting, and mitigating malware has become a critical priority in the cybersecurity landscape. Traditional signature-based detection systems, which rely on predefined patterns to identify known malware, are increasingly ineffective against sophisticated and polymorphic malware variants. In response, researchers have turned to machine learning (ML) algorithms and artificial intelligence (AI) techniques to develop more robust and adaptive malware detection systems. ML-based approaches leverage the power of data driven models to detect and classify malware by learning from vast amounts of labelled samples, capturing intricate patterns and behaviours that distinguish malicious software from benign programs [1–4].

Whilst ML algorithms have demonstrated remarkable success in malware detection, their effectiveness comes at the cost of using too many features, which then require more processing. Many state-of-the-art ML models, such as Deep Neural Networks (DNNs) have been used along with app features making it challenging to process and also understand how they arrive at their conclusions. Datasets used in malware detection have become increasingly complex containing a substantial number of features that can go up to thousands and contain non-binary numerical values. Moreover, when the data are complex, there is a necessity to develop more complex algorithms such as neural networks and deep learning-based algorithms, which are time and energy consuming since when new data arrive, usually they will need to be retrained.

Thus, it is of high interest to the community to identify a way to use the minimum possible number of optimally selected features, which will allow using less processing intense algorithms, such as Decision Trees and Random Forest algorithms that can provide very high accuracy in smaller, balanced, binary datasets.

The primary objective of this research work is to deliver a robust methodology that converts all non-binary values of an Android malware dataset into binary values, and further identify a small number of optimal features that can be used to detect malware, while maintaining high accuracy. The contributions of this article can be summarized as follows:

1. A novel methodology that provides an innovative feature selection methodology specifically tailored for Android malware detection is delivered.
2. The methodology has been evaluated using two real datasets, and the results indicate that both the number of features is highly reduced, whilst high accuracy is yet maintained.

The paper is structured as follows. Section 2 presents and discusses the related work. Section 3 introduces the proposed methodology. Section 4 reports the experimental evaluation results, whereas Section 5 summarizes the conclusions and gives future work directions.

2 Related work

Feature selection is an important issue for ML [5, 6]. It is noticeable that Android malware detection has been an area of active research in the last decade with several contributions

across the world. In this sequel, we present the main characteristics of these seminal contributions in chronological order.

Drebin is a lightweight hybrid method, which was proposed in 2014. It uses both static and dynamic information to detect malware during runtime in an Android device. *Drebin* performs a broad static analysis of Android applications and automatically identifies typical patterns of malicious activities that can be presented and explained to the user. *Drebin* enables detecting 94% of the malware in a large dataset with few false alarms [3].

Along the same period, a relevant contribution addresses the increasing concern over information security on Android mobile devices, where user control over sensitive data is overshadowed by the proliferation of applications. Focusing on permission-based malware detection, the study analyses feature selection methods and classification algorithms. Findings indicate that Random Forest and J48 decision tree algorithms exhibit higher performance across various feature selection methods, highlighting their effectiveness in detecting malicious software in Android applications [7].

Another relevant paper addresses the prevalence of malicious applications targeting the Android platform by proposing a ML-based approach for Android malware detection. Utilizing evolutionary Genetic algorithm (GA) for feature selection, the study demonstrates the effectiveness of discriminatory feature selection to reduce the feature dimensionality. Experimental results confirm that GA produces an optimized feature subset, reducing the feature dimension to less than half of the original set. Despite the reduced feature space, ML classifiers maintain a classification accuracy of over 94%, thereby positively impacting computational complexity [8].

Yet, another paper introduces a method for detecting Android malware using feature selection with GAs, recognizing the importance of malware detection systems for Android devices due to widespread use and frequent targeting by malware developers. Three classifier methods with different feature subsets selected using GAs were implemented for comparative analysis of Android malware detection. Combining Support Vector Machines (SVMs) with GAs resulted in the highest accuracy of 98.45% with 16 selected permissions, utilizing a dataset of 1740 samples comprising 1119 malware and 621 benign samples respectively [9].

Close to this work the literature present three relevant works contributions which are briefly described in the following section. *DeepDroid* is introduced to address the vulnerability of smartphones to malware attacks due to their open nature and widespread usage. Focusing on the Android permission model, the study proposes a framework leveraging feature selection methods and DNN classification for malware detection. Through empirical evaluation of 120,000 Android apps and application of five feature selection techniques, the study highlights the efficacy of Principal Component Analysis (PCA) in detecting 94% of Android malware from real-world apps [10]. The second paper introduces *FSDroid* to address the prevalent issue of malware-infected Android apps, emphasizing the importance of precise permission specification by developers to mitigate risks. Through experimental work, an effective malware detection system is developed, employing various feature selection methods to identify optimal features. Utilizing a Least Square Support Vector Machine (LSSVM) learning approach with three kernel functions, including linear, radial basis, and polynomial, the model achieves a remarkable malware detection rate of 98.8% across 200,000 Android apps. *FSDroid* outperforms existing anti-virus scanners and frameworks proposed in the literature, demonstrating a 3% higher detection rate [11]. The third paper introduces *SemiDroid* to address the escalating threat of malware targeting Android devices by focusing on the underlying permission model. By considering permission and API calls as key features, the research develops a malware detection model. Through the implementation of ten feature selection approaches, appropriate feature sets are identified from thirty different categories

of Android apps. Utilizing five unsupervised ML algorithms, distinct models are developed and evaluated on a dataset of 500,000 Android apps. Empirical results highlight the effectiveness of utilizing rough set analysis for feature selection and the farthest first algorithm for achieving a high detection rate of 98.8% in identifying malware from real-world apps [12].

Didroid is a pioneering approach employing deep image learning for Android malware classification and characterization. Despite extensive efforts in ML-based detection and classification, few ventures delve into deep learning for this purpose. Addressing the pressing need to combat Android malware, *Didroid* utilizes a DNN to classify and characterize malware samples from a comprehensive dataset. Achieving an impressive accuracy of 93.36% and a low log loss for both training and testing sets, *Didroid* showcases the effectiveness of image-based deep learning in enhancing malware analysis and mitigation efforts [13].

Salah et al. [14] focuses on static analysis for Android malware detection, aiming to identify symmetric features across malware applications to enhance detection efficiency. Unlike existing methods that extract asymmetric patterns like application permissions, the proposed approach selects crucial features representing symmetric patterns for classification. Introducing a novel feature selection method inspired by Term Frequency - Inverse Document Frequency (TF-IDF) and a technique for merging Android application URLs, the study significantly reduces feature space and memory size while achieving the highest reported accuracy for the *Drebin* dataset to date. Evaluation results indicate a remarkable accuracy of 99% using linear SVM classifiers [14].

Another paper that was published around the same period addresses the growing threat of Android malware and the need for effective detection methods to safeguard sensitive information on smartphones. Utilizing the CICInvesAndMal2019 dataset, the study employs Android permissions and intent as feature sets for malware detection. PCA is utilized for feature selection. Through experimentation with various ML models, Random Forest emerges as the best classifier, achieving an impressive accuracy of 96.05%. This highlights the potential of ML techniques in large-scale detection of Android malware [15].

EntropLyzer, a novel entropy-based behavioural analysis technique, is introduced to address the escalating threat of Android malware. Unlike existing methods that focus on static features or API calls, *EntropLyzer* leverages multiple dynamic characteristics to classify the behaviour of prominent Android malware. Through extensive experimentation on 12 major malware categories and 147 malware families from the CCCS-CIC-AndMal2020 dataset, *EntropLyzer* effectively identifies and characterizes Android malware behaviour using six classes of dynamic features. Results demonstrate the efficacy of the entropy-based analysis in accurately classifying malware behaviour, even after emulator rebooting [16].

JOWMDroid is a novel Android malware detection scheme addressing feature importance disparity. It optimizes weight-mapping and classifier parameters jointly to assign weights to features, improving classification accuracy. Initial feature selection is performed using information gain, followed by weight calculation using three ML models. Five weight-mapping functions refine initial weights, and parameters are optimized using the differential evolution algorithm. Experimental results show *JOWMDroid* outperforms four state-of-the-art methods, enhancing competitiveness in Android malware detection [17].

TC-Droid is an automatic framework for Android malware detection, aimed at addressing the inefficiency of manual feature engineering. Inspired by text classification methods, *TC-Droid* utilizes convolutional neural networks (CNNs) to analyse text sequences of app analysis reports generated by AndroPyTool,[1] extracting significant information without the need for manual feature engineering. Evaluation with real-world samples demonstrates *TC-Droid*'s

[1] https://github.com/alexMyG/AndroPyTool

superior performance over state-of-the-art models like *Drebin*, as well as classic models such as Naïve Bayes, Logistic Regression, K Nearest Neighbour and Random Forest. Through multiple experimental settings and comparisons, *TC-Droid* proves to be effective and flexible in Android malware detection tasks [18].

The effectiveness of GA-based feature selection in Android malware detection is explored and compared with other methods in a recent study. Utilizing nine ML algorithms with GA-based feature selection on a dataset comprising 5000 benign applications and 2500 malware samples, the study evaluates its performance. Results indicate that GA-based feature selection outperforms information gain-based methods and significantly reduces processing time. This suggests that integrating GAs into Android malware detection offers valuable enhancements, particularly in improving detection performance and efficiency [19].

Abawajy et al. [20] discusses the challenge of Android malware detection and the role of feature selection in enhancing ML performance. It formulates feature selection as a quadratic programming problem and evaluates several filter-based methods in the context of Android malware detection. Empirical findings confirm the necessity of feature selection for improving accuracy and reducing runtime. Additionally, the study highlights variations in performance across different learning algorithms, emphasizing the importance of careful selection based on specific requirements [20].

A novel framework for Android malware detection, leveraging wrapping feature selection (WFS) with a hybrid of random forest and greedy stepwise (RF-GreedySW) techniques to optimize feature selection, has appeared recently. This approach aims to address challenges posed by evolving malware variants. Evaluation on the CIC-InvesAndMal2019 dataset shows significant accuracy improvements. Overall, the proposed framework enhances classification accuracy and robustness in detecting Android malware features [21].

PAIRED system is a lightweight Android malware detection system based on explainable ML. In this paper, a feature importance algorithm is introduced that reduces the number of features from 215 to 35 in a particular dataset, while the accuracy remains high, exceeding 98%, whereas at the same time a small footprint on the device is maintained. Then a typical ML algorithm such as a decision tree or random forest is applied. The classifier model is explained using Shapley Additive Explanation (SHAP) values [2].

In another study, a framework is developed to enhance the detection of obfuscated and hidden malware, addressing crucial challenges in cybersecurity. By employing advanced memory feature extraction techniques and a stacked ensemble ML model, the framework efficiently identifies obfuscated malware. The effectiveness of the framework is validated using a specialized malware memory dataset (MalMemAnalysis-2022), achieving impressive accuracy and F-score rates of 99.00% and 99.02%, respectively, in rapidly identifying concealed malware instances [22].

DroidRL is a framework addressing Android malware detection by employing a DDQN algorithm for efficient feature selection. By utilizing a recurrent neural network (RNN) and word embedding for feature representation, *DroidRL* autonomously selects relevant features, achieving high accuracy of 95.6% with only 24 selected features when paired with Random Forest as a classifier [23].

A related work proposes a ML-based Android malware detection system, aiming to distinguish malicious apps from benign ones. Employing a linear regression-based feature selection method, unnecessary features are eliminated during the feature selection stage, reducing the dimensionality of the feature vector and training time. Results indicate a significant performance with an F-score of 96.1% achieved by using a minimum of 27 features, highlighting the effectiveness of the proposed approach in real-time malware detection systems [24].

Another recent paper introduces a high-performance malware detection system utilizing deep learning and feature selection techniques to address the challenges posed by large and high-dimensional data. Two malware datasets undergo pre-processing, followed by correlation-based feature selection to generate different feature-selected datasets. Dense and Long Short-Term Memory (LSTM) based deep learning models are trained using these datasets and evaluated based on various performance metrics. Results indicate that certain feature-selected scenarios maintain performance comparable to the original dataset, with varying degrees of reduction in feature space and performance degradation depending on the dataset characteristics. The reduction rates range from 18.18% to 42.42% for the first dataset and 81.77% to 93.5% for the second dataset, with performance degradation ranging from 0.07% to 5.84% and 3.79% to 9.44%, respectively [25].

Finally, Keyvanpour et al. [26] addresses the growing challenge of malware intrusion on the popular Android operating system by proposing a novel method based on the random forest algorithm. Three different feature selection techniques, namely effective, high weight, and effective group feature selection, are applied and evaluated. Experiments conducted on the *Drebin* dataset demonstrate that applying feature selection methods improves accuracy metrics and reduces processing time. Comparison with baseline algorithms confirms the superiority of the proposed feature selection model with random forest, showcasing its effectiveness in malware detection on Android [26].

While related problems are addressed by several methods that differ in their underlying assumptions, techniques, and goals, our proposed method differs significantly as its foundational idea is to identify the fewest relevant features utilized by each malware family by creating subsets for each one using binarization.

3 Methodology

This work proposes an alternative approach to the literature to select an optimum number of features while maintaining high accuracy in its ML approach. The proposed method is based on pre-processing large datasets that contain features from various malware families and benign applications. Table 1 provides the notation that is being used throughout the methodology section.

The first step of the proposed method is to load a large dataset that contains entries from different malware families and benign applications. We load the dataset, C, and check how many numerical and how many categorical columns exist. If the majority of the features are categorical, then we delete the numerical features if $|N| \leq |C|/10$, where $|C|$ represents the total number of columns of C, and $|N|$ represents the number of numerical columns of C.

On the other hand, if $|N| > |C|/10$, then we use binary encoding to convert these numerical columns to categorical ones. The binarization is applied as follows: For each numerical column $x \in C$, for each value $v \in x$, we set v to 0 if $v < \mu(x)$; otherwise, we set v to 1. Thus, the dataset is converted to contain only binary categorical values, with 0 indicating 'not used' and 1 indicating 'used'.

At the next step, we develop balanced datasets for each malware family, such as Trojan, Spyware, Ransomware and others, by employing the One Versus One (OVO) approach. This method breaks down a multi-class problem into several binary problems. To address class imbalance, we selected the maximum available number from each malware family and an equal number of benign entries randomly [27]. After the balanced subsets are created for

Table 1 Notations

Notation	Description
C	Set of columns that holds the initial large classification dataset.
N	Subset of C with numerical columns only.
	If all columns of C are categorical (binary), then N will be empty.
x	Column of a set.
$\mu(x)$	Mean value of column x.
n	Fixed number ranging from 1 to the total number of malware families in C.
	We set n=3 to choose three random malware families from the 1st dataset,
	and n=10 to choose ten random malware families from the 2nd dataset.
D	Set of n balanced sets created by transforming C.
	For each malware family, a specific member of D will be created.
A	Percentage of 0s in each column, set to 60%.
B	Percentage of 1s in each column, set to 70%.
D'	Set derived from D, where at least 60% of the values in each column are 0s,
	and at least 70% of the values in each column are 1s.

each malware family, a series of steps is followed for each one, where the overused and underused columns/features are dropped; then, new updated balanced datasets are created for each malware family, where ML algorithms can be applied on them. Moreover, Figure 1 gives an overview of the proposed method.

For each malware family, we create a set of subsets $D1$, $D2$, ..., Dn, where n is the total number of malware families in C. Then, on each member of set D, we delete the columns with at least 60% 0s and 70% 1s since this combination of values provided the most accurate results. The output is a new set of balanced datasets $D1'$, $D2'$, ..., Dn', where each one holds data for one specific malware family. Subsequently, a classification algorithm - such a decision tree or a random forest - can be applied to each member of this latter set. Algorithm 1 below depicts the particular steps.

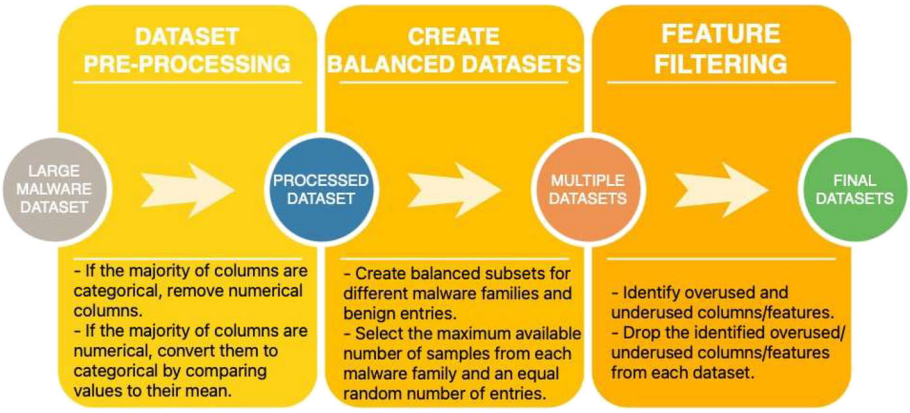

Figure 1 High level architecture

Algorithm 1 Proposed binarization method.

Input: Dataset C

We determine the number of numerical and categorical columns in the C
if the number of numerical columns $N \leq |C|/10$ **then**
  Delete N, all numerical columns, from C
else ▷ apply binarization
  for each numerical column in set C **do**
    for each value v in the column x **do**
      if value $v < \mu(x)$ **then** ▷ $\mu(x)$ is the mean value of the column
        Set value v equal to 0
      else
        Set value v equal to 1
      end if
    end for
  end for
end if
We have transformed set C to a new set C'. From C', for each malware family, we create a set of balanced subsets $D1$, $D2$, ..., Dn
for i = 1 to n **do** ▷ n=3 for the 1st, and n=10 for the 2nd dataset
  Calculate the percentage of zeros in each column x
  if the percentage of zeros in a column $x \geq A$ **then** ▷ threshold A=60%
    Delete column x
  end if
  Calculate the percentage of ones in each column
  if the percentage of ones in a column $x \geq B$ **then** ▷ threshold B=70%
    Delete column x
  end if
end for
We have transformed the set $D1$, $D2$, ..., Dn to the set $D1'$, $D2'$, ..., Dn'

Output: Set $D1'$, $D2'$, ..., Dn' consists of balanced sets, each one generated for each malware family. Then, a classification algorithm can be applied to each set.

4 Experimental evaluation

The experimental evaluation was performed on an intel i7 machine running Linux, with Python programming language and the scikit ML library. For our experimentation we used two public datasets: KronoDroid[2] and CCCS-CIC-AndMal 2020.[3] The settings used in all experiments for the decision tree and random forest algorithms are the default ones from the scikit learn library. Finally, it is mentioned that both threshold values A and B for the KronoDroid dataset have been set to 60%, whereas for the CCCS-CIC-AndMal 2020 dataset they have been both set to 70%.

In the following subsections the settings, the data, the evaluation metrics, and the results are presented. In particular, Sections 4.1 and 4.2 carry the title and the information of each original non processed large dataset and the tables in each subsection display the statistical information for each family dataset that has been created from the large datasets.

[2] https://github.com/aleguma/kronodroid

[3] https://www.unb.ca/cic/datasets/andmal2020.html

Table 2 KronoDroid dataset

Malware family	Malware	Benign	Total size	Total features	Remaining features
Airpush	7775	7775	15550	489	28
BankBot	1297	1297	2594	489	19
SMS	5019	5019	10038	489	27

4.1 KronoDroid

This dataset contains 78,137 entries, 41,382 of which are malware and 36,755 of which are benign with a total of 489 static and dynamic features [28]. There are multiple categories of malware in this dataset but for the evaluation purposes we selected the following three: Airpush, BankBot, and SMS and we developed three balanced subsets, the statistical details of which are shown in Table 2.

Remaining features This dataset has 489 features. After the application of the proposed method the Airpush category has 28, the BankBot 19 and the SMS 27 features. Table 3 shows which malware family utilizes which features.

Table 3 KronoDroid remaining features

Item	Air Push	Bankbot	SMS
setrlimit	✓		
prctl	✓		
munmap	✓		
mprotect	✓		
madvise	✓		
getdents64	✓		
fstatfs64	✓		
socketpair	✓		
setsockopt	✓		
sysinfo	✓		✓
epoll_ctl	✓	✓	
clone	✓		
getrlimit	✓		
sys_306	✓		
sys_315	✓		
sys_317	✓		
sys_339	✓		✓
access_coarse_location	✓		✓
access_fine_location	✓		✓
access_wifi_state	✓		✓
read_phone_state	✓	✓	✓
receive_boot_completed	✓	✓	✓
write_external_storage	✓	✓	✓
nr_permissions	✓	✓	✓

Table 3 continued

Item	Air Push	Bankbot	SMS
normal	✓		✓
dangerous	✓	✓	✓
total_perm	✓	✓	✓
detection_ratio	✓	✓	✓
lseek		✓	
bind_device_admin		✓	
call_phone		✓	
get_tasks		✓	✓
read_contacts		✓	
read_sms		✓	✓
send_sms		✓	✓
wake_lock		✓	✓
signature		✓	✓
custom_yes		✓	✓
scanners		✓	✓
sys_340			✓
change_configuration			✓
change_network_state			✓
change_wifi_state			✓
mount_unmount_filesystems			✓
vibrate			✓
write_settings			✓

4.2 CCCS-CIC-AndMal 2020

This dataset consists of 400,000 entries, 200,000 of which are malware and 200,000 of which are benign. The dataset contains several different well-known malware families with a total number of 9,503 unlabelled features [13, 16]. From this dataset, we developed 10 balanced subsets, the statistical details of which are shown in Table 4.

Remaining features This dataset does not name the features and we have used the column number instead. The collected data comprises a comprehensive set of features related to various malware categories, each with a unique combination of feature numbers. The Backdoor category has 21 features, the Banker category consists of 14 features, the Dropper category contains 16 features, the File Infector category has 20 features, the PUA category consists of 20 features, the Ransomware category has 17 features, the Scareware category contains 9 features, the SMS category has 15 features, the Spyware category consists of 23 features and finally the Trojan category has 27 features. Table 5 shows which malware family utilizes which features.

4.3 Evaluation metrics

We used the accuracy, precision, and recall evaluation metrics, which have been extensively used in the literature for similar research [2, 22]. These are defined in (1), (2), and (3)

Table 4 CCCS-CIC-AndMal 2020 dataset

Malware family	Malware	Benign	Total size	Total features	Remaining features
Backdoor	1538	1538	3076	9503	21
Banker	887	887	1774	9503	14
Dropper	2302	2302	4604	9503	16
File Infector	669	699	1338	9503	20
PUA	2051	2051	4102	9503	20
Ransomware	6202	6202	12404	9503	17
Scareware	1556	1556	3112	9503	9
SMS	3125	3125	6250	9503	15
Spyware	3540	3540	7080	9503	23
Trojan	13559	13559	27118	9503	27

Table 5 Features and their corresponding malware categories

Feature	Backdoor	Banker	Dropper	File Infector	PUA	Ransomware	Scareware	SMS	Spyware	Trojan
13	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
14	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
24	✓		✓						✓	✓
26	✓		✓		✓	✓		✓	✓	✓
28	✓				✓				✓	✓
32				✓				✓		
35	✓	✓	✓	✓	✓	✓	✓		✓	✓
36	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
37			✓	✓	✓					✓
38	✓	✓	✓		✓	✓				✓
39		✓		✓						✓
48										✓
50	✓	✓	✓	✓	✓	✓		✓	✓	✓
51						✓		✓	✓	
52	✓				✓				✓	✓
55								✓		
57				✓						✓
58				✓						✓
63										✓
66						✓		✓	✓	
1537	✓				✓				✓	✓
1538	✓				✓				✓	✓

Table 5 continued

Feature	Backdoor	Banker	Dropper	File Infector	PUA	Ransomware	Scareware	SMS	Spyware	Trojan
2057	✓		✓						✓	
3052										✓
3053	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3281	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3282	✓		✓		✓	✓		✓	✓	✓
3283				✓					✓	✓
3293				✓						✓
3335				✓					✓	✓
3538						✓				
9134	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
9136	✓	✓		✓	✓	✓			✓	✓
9143				✓						
9332	✓	✓		✓	✓	✓			✓	✓

respectively. TP stands for True Positive, TN for True Negative, FP for False Positive, and FN for False Negative.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

4.4 Results

Tables 6 and 7 present the evaluation results for the KronoDroid and CCCS-CIC-andMal 2020 datasets respectively. The first column in each table presents the malware family initially using the total number of features, followed by the reduced number of features. Then, the remaining three columns present the results for the Accuracy, Precision and Recall metrics for the Decision Tree (DT) and Random Forest (RF) algorithms. Our proposed method represents a novel approach to addressing the problem of Android malware detection utilizing binarization methods for feature engineering. While there are existing methods that tackle related issues, they differ significantly from our approach in terms of their underlying assumptions, techniques, and goals. For example, other methods assume that one dataset is used or do not explicitly pre-process the data, which our core approach. As such, a direct comparison with existing methods will not provide a meaningful or fair evaluation of our approach. Instead, we have focused on demonstrating the effectiveness and validity of our method through rigorous experimentation and analysis, and have highlighted its unique features and contributions throughout the paper.

The proposed method significantly reduces the features which allows for simple algorithms to run and identify malware with very high accuracy, while it is easy to understand which are the core characteristics of each malware family since there are only very few features

Table 6 KronoDroid results

Malware family	Accuracy	Precision	Recall
Airpush (489 Features)	DT: 99% RF: 99%	DT: 99% RF: 99%	DT: 99% RF: 99%
Airpush (28 Binary features)	DT: 99% RF: 99%	DT: 99% RF: 99%	DT: 99% RF: 99%
BankBot (489 Features)	DT: 100% RF: 100%	DT: 100% RF: 100%	DT: 100% RF: 100%
BankBot (19 Binary features)	DT: 99% RF: 100%	DT: 99% RF: 100%	DT: 100% RF: 100%
SMS (489 Features)	DT: 99% RF: 99%	DT: 99% RF: 99%	DT: 99% RF: 99%
SMS (27 Binary features)	DT: 99% RF: 99%	DT: 99% RF: 99%	DT: 99% RF: 99%

Table 7 CCCS-CIC-AndMal 2020 results

Malware family	Accuracy	Precision	Recall
Backdoor (9503 Features)	DT: 95.30% RF: 97.60%	DT: 95.80% RF: 97.50%	DT: 95.20% RF: 97.40%
Backdoor (21 Binary features)	DT: 95.70% RF: 97.50%	DT: 95.60% RF: 97%	DT: 95.90% RF: 97.30%
Banker (9503 Features)	DT: 97.90% RF: 98.90%	DT: 97.90% RF: 98.70%	DT: 97.90% RF: 98.90%
Banker (14 Binary features)	DT: 97.80% RF: 99%	DT: 97.90% RF: 98.80%	DT: 97.50% RF: 99%
Dropper (9503 Features)	DT: 97.10% RF: 98.30%	DT: 97.30% RF: 98.50%	DT: 96.70% RF: 98.50%
Dropper (16 Binary features)	DT: 97.20% RF: 98.30%	DT: 97.10% RF: 98.30%	DT: 97.10% RF: 98.50%
File Infector (9503 Features)	DT: 98.20% RF: 98.80%	DT: 97.70% RF: 99.10%	DT: 97.80% RF: 99.10%
File Infector (20 Binary features)	DT: 98% RF: 98.50%	DT: 98.40% RF: 98.80%	DT: 98.20% RF: 98.30%
PUA (9503 Features)	DT: 91.70% RF: 95.10%	DT: 91.70% RF: 96.10%	DT: 92.20% RF: 95%
PUA (20 Binary features)	DT: 91.90% RF: 95.20%	DT: 92.10% RF: 95.30%	DT: 92.20% RF: 94.80%

Table 7 continued

Malware family	Accuracy	Precision	Recall
Ransomware (9503 Features)	DT: 98.70% RF: 99.30%	DT: 98.70% RF: 99.30%	DT: 98.70% RF: 99.20%
Ransomware (17 Binary features)	DT: 98.70% RF: 99.50%	DT: 98.60% RF: 99.40%	DT: 98.80% RF: 99.20%
Scareware (9503 Features)	DT: 95.60% RF: 98.20%	DT: 95.50% RF: 98.10%	DT: 96% RF: 98%
Scareware (9 Binary features)	DT: 95.70% RF: 97.90%	DT: 95.20% RF: 97.80%	DT: 95.50% RF: 97.90%
SMS (9503 Features)	DT: 99.80% RF: 99.70%	DT: 99.90% RF: 99.70%	DT: 99.80% RF: 99.80%
SMS (15 Binary features)	DT: 99.80% RF: 99.70%	DT: 99.90% RF: 99.70%	DT: 99.90% RF: 99.80%
Spyware (9503 Features)	DT: 99.90% RF: 99.80%	DT: 99.90% RF: 99.80%	DT: 99.80% RF: 99%
Spyware (23 Binary features)	DT: 99.80% RF: 99.80%	DT: 99.80% RF: 99.80%	DT: 99.80% RF: 99.80%
Trojan (9503 Features)	DT: 97.40% RF: 98.50%	DT: 97.30% RF: 98.50%	DT: 97.30% RF: 98.40%
Trojan (27 Binary features)	DT: 97.20% RF: 98.50%	DT: 97.30% RF: 98.50%	DT: 97.30% RF: 98.60%

used. For the first dataset there is considerable reduction of features from 55 to around half of those, while on the second dataset considering that there are a total of 9503 features there is a significant reduction of features. Moreover, as shown by the results the accuracy remains similar as using all features while in some cases the accuracy is improved which shows that when the most significant features are identified malware can be detected with higher accuracy. Finally, our research employs binarization in the detection of Android malware, utilizing smaller datasets, each representing a distinct malware family. In contrast, other studies do not employ similar methods, rendering direct comparisons inequitable.

5 Conclusions and future work

Finding a way to select a number of features from Android datasets that is close to optimal and that can be used with simple ML algorithms is of tremendous importance because of their simplicity to be used and understood by less experienced people. Most advanced algorithms such as neural networks are complex and difficult to interpret, thus understanding exactly how an algorithm decides what is a malware or benign application plus it is not efficient to use too many features. In the literature there are many algorithms that are very accurate but consequently they are very advanced algorithms which are difficult to implement and time consuming to develop. However, simpler ML algorithms such as Decision Trees are not very

accurate when the datasets are complex. Thus, in this work we show how we can process datasets to simplify them to be able to detect malware with comparably high accuracy using simpler ML algorithms such as Decision trees and Random Forests. Then, we used large Android malware datasets, which contain both malware from different families and benign apps. In the first step we pre-process these data by converting all numerical values that are not binary into binary (0 and 1), which simplifies the processing time and simplicity significantly. Still, large datasets are based on many features (columns) since many malware families are represented. Feature engineering can be used to remove columns, however too many features will be available still, which once again will make processing difficult. To bypass this issue, we convert large datasets into smaller and balanced datasets, one for each malware family. All these steps allow simpler ML algorithms to be very accurate as well. The simplicity of the smaller datasets allows interpreting features in a yes/no basis where we can see that if a particular feature exist then a particular entry is malware.

The experimental results indicate that when the datasets are pre-processed into binary and into individual families the accuracy remains very high and in certain cases higher than when using all features. This highlights the fact that with the correct use of data we can achieve very accurate results. In conclusion, feature selection is of paramount importance in the field of Android malware detection. The proposed methodology that is based on data pre-processing improves the selection of features since it can provide very high accuracy using simple ML algorithms, such as Decision Trees and Random Forests.

In the future we plan to further work in this area by developing a reinforcement learning algorithm to automatically identify the optimal settings and number of features in a dataset and apply our methodology in other areas.

Author Contributions All authors contributed to developing this manuscript.

Funding Open access funding provided by the Cyprus Libraries Consortium (CLC). No funding received to assist with the preparation of this manuscript.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Ethical approval Not applicable.

Competing interests The authors declare no competing interests.

References

1. Ozogur, G., Erturk, M.A., Gurkas Aydin, Z., Aydin, M.A.: Android malware detection in bytecode level using TF-IDF and XGBoost. Comput. J. **66**(9), 2317–2328 (2023). https://doi.org/10.1093/comjnl/bxac198

2. Alani, M.M., Awad, A.I.: PAIRED: An explainable lightweight Android malware detection system. IEEE Access. **10**, 73214–73228 (2022). https://doi.org/10.1109/access.2022.3189645
3. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K.: Drebin: Effective and explainable detection of Android malware in your pocket. In: Proceedings of the 21st Annual Network & Distributed System Security Symposium (NDSS), pp. 23–26 (2014). https://www.ndss-symposium.org/ndss2014/drebin-effective-and-explainable-detection-android-malware-your-pocket
4. Li, Y., Xiong, Z., Zhang, T., Zhang, Q., Fan, M., Xue, L.: Ensemble framework combining family information for Android malware detection. Comput. J. **66**(11), 2721–2740 (2023). https://doi.org/10.1093/comjnl/bxac114
5. Moorthy, R.S., Pabitha, P.: Accelerating analytics using improved binary particle swarm optimization for discrete feature selection. Comput. J. **65**(10), 2547–2569 (2022). https://doi.org/10.1093/comjnl/bxab089
6. Zheng, W., Jin, M.: Improving the performance of feature selection methods with low-sample-size data. Comput. J. **66**(7), 1664–1686 (2023). https://doi.org/10.1093/comjnl/bxac033
7. Pehlivan, U., Baltaci, N., Acartürk, C., Baykal, N.: The analysis of feature selection methods and classification algorithms in permission based Android malware detection. In: Proceedings of the IEEE Symposium on Computational Intelligence in Cyber Security (CICS), pp. 1–8 (2014). https://doi.org/10.1109/cicybs.2014.7013371
8. Fatima, A., Maurya, R., Dutta, M.K., Burget, R., Masek, J.: Android malware detection using genetic algorithm based optimized feature selection and machine learning. In: Proceedings of the 42nd International Conference on Telecommunications & Signal Processing (TSP), pp. 220–223 (2019). https://doi.org/10.1109/tsp.2019.8769039
9. Yildiz, O., Doğru, I.A.: Permission-based Android malware detection system using feature selection with genetic algorithm. International Journal of Software Engineering & Knowledge Engineering. **29**(2), 245–262 (2019). https://doi.org/10.1142/s0218194019500116
10. Mahindru, A., Sangal, A.L.: DeepDroid: Feature selection approach to detect Android malware using deep learning. In: Proceedings of the 10th IEEE International Conference on Software Engineering & Service Science (ICSESS), pp. 16–19 (2019). https://doi.org/10.1109/icsess47205.2019.9040821
11. Mahindru, A., Sangal, A.L.: FSDroid: A feature selection technique to detect malware from Android using machine learning techniques. Multimedia Tools & Applications. **80**(8), 13271–13323 (2021). https://doi.org/10.1007/s11042-020-10367-w
12. Mahindru, A., Sangal, A.L.: SemiDroid: A behavioral malware detector based on unsupervised machine learning techniques using feature selection approaches. International Journal of Machine Learning & Cybernetics. **12**(5), 1369–1411 (2021). https://doi.org/10.1007/s13042-020-01238-9
13. Rahali, A., Lashkari, A.H., Kaur, G., Taheri, L., Gagnon, F., Massicotte, F.: Didroid: Android malware classification and characterization using deep image learning. In: Proceedings of the 10th International Conference on Communication & Network Security (ICCNS), pp. 70–82 (2020). https://doi.org/10.1145/3442520.3442522
14. Salah, A., Shalabi, E., Khedr, W.: A lightweight Android malware classifier using novel feature selection methods. Symmetry. **12**(5), 858 (2020). https://doi.org/10.3390/sym12050858
15. Sangal, A., Verma, H.K.: A static feature selection-based Android malware detection using machine learning techniques. In: Proceedings of the International Conference on Smart Electronics & Communication (ICOSEC), pp. 48–51 (2020). https://doi.org/10.1109/icosec49089.2020.9215355
16. Keyes, D.S., Li, B., Kaur, G., Lashkari, A.H., Gagnon, F., Massicotte, F.: EntropLyzer: Android malware classification and characterization using entropy analysis of dynamic characteristics. In: Proceedings of the IEEE Reconciling Data Analytics, Automation, Privacy, & Security: A Big Data Challenge (RDAAPS), pp. 1–12 (2021). https://doi.org/10.1109/rdaaps48126.2021.9452002
17. Cai, L., Li, Y., Xiong, Z.: JOWMDroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. Computers & Security. **100**, 102086 (2021). https://doi.org/10.1016/j.cose.2020.102086
18. Zhang, N., Tan, Y.A., Yang, C., Li, Y.: Deep learning feature exploration for Android malware detection. Appl. Soft Comput. **102**, 107069 (2021). https://doi.org/10.1016/j.assoc.2020.107069
19. Lee, J., Jang, H., Ha, S., Yoon, Y.: Android malware detection using machine learning with feature selection based on the genetic algorithm. Mathematics. **9**(21), 2813 (2021). https://doi.org/10.3390/math9212813
20. Abawajy, J., Darem, A., Alhashmi, A.A.: Feature subset selection for malware detection in smart IoT platforms. Sensors. **21**(4), 1374 (2021). https://doi.org/10.3390/s21041374
21. Smmarwar, S.K., Gupta, G.P., Kumar, S.: A hybrid feature selection approach-based Android malware detection framework using machine learning techniques. In: Proceedings of the International Conference on Cyber Security, Privacy & Networking (ICSPN), pp. 347–356 (2022). https://doi.org/10.1007/978-981-16-8664-1_30

22. Carrier, T., Victor, P., Tekeoglu, A., Lashkari, A.H.: Detecting obfuscated malware using memory feature engineering. In: Proceedings of the 8th International Conference on Information Systems Security & Privacy (ICISSP), pp. 177–188 (2022).https://doi.org/10.5220/0010908200003120
23. Wu, Y., Li, M., Zeng, Q., Yang, T., Wang, J., Fang, Z., Cheng, L.: DroidRL: Feature selection for Android malware detection with reinforcement learning. Computers & Security. **128**, 103126 (2023). https://doi.org/10.1016/j.cose.2023.103126
24. Sahin, D.Ö., Kural, O.E., Akleylek, S., Kiliç, E.: A novel permission-based Android malware detection system using feature selection based on linear regression. Neural Computing & Applications, 1–16 (2023) https://doi.org/10.1007/S00521-021-05875-1
25. Alomari, E.S., Nuiaa, R.R., Alyasseri, Z.A.A., Mohammed, H.J., Sani, N.S., Esa, M.I., Musawi, B.A.: Malware detection using deep learning and correlation-based feature selection. Symmetry. **15**(1), 123 (2023). https://doi.org/10.3390/sym15010123
26. Keyvanpour, M.R., Barani Shirzad, M., Heydarian, F.: Android malware detection applying feature selection techniques and machine learning. Multimedia Tools & Applications. **82**(6), 9517–9531 (2023). https://doi.org/10.1007/s11042-022-13767-2
27. Sen, A., Islam, M.M., Murase, K., Yao, X.: Binarization with boosting and oversampling for multiclass classification. IEEE Transactions on Cybernetics. **46**(5), 1078–1091 (2015). https://doi.org/10.1109/tcyb.2015.2423295
28. Guerra-Manzanares, A., Bahsi, H., Nõmm, S.: Kronodroid: Time-based hybrid-featured dataset for effective android malware detection and characterization. Computers & Security. **110**, 102399 (2021). https://doi.org/10.1016/j.cose.2021.102399