Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

A multi-tiered feature selection model for android malware detection based on Feature discrimination and Information Gain

Parnika Bhat [a,*], Kamlesh Dutta [a]

[a] Department of Computer Science & Engineering, National Institute of Technology, Hamirpur, HP, India

ARTICLE INFO

Article history:
Received 18 May 2021
Revised 13 October 2021
Accepted 5 November 2021
Available online xxxx

Keywords:
Android
Feature selection
Malware detection
Machine learning
Static analysis

ABSTRACT

With the rise in popularity and its open system architecture, Android has become vulnerable to malicious attacks. There are several malware detection approaches available to fortify the Android operating system from such attacks. These malware detectors classify target applications based on the patterns found in the features present in the Android applications. As the analytics data continues to grow, it negatively impacts the Android defense mechanism. A large number of irrelevant features has become the performance bottleneck of the detection mechanism. This paper presents a multi-tiered feature selection model, which can discover relevant and significant features for improving the accuracy of malware detection approaches. The proposed method applies five machine learning classification techniques to the selected feature set. This work presents the Optimal Static Feature Set (OSFS) and, Most Important Features (MIFs) discovered with each machine learning approach. Rigorous testing and analysis show that Random Forest classification achieves the highest Accuracy rate of 96.28%.

1. Introduction

The omnipresent use of cell phones has made them an indispensable part of daily life activities. Android has undoubtedly gained popularity over all other operating systems available in the market (Statcounter, 2021). The pervasive use of Android has gained increased attention from cyber attackers (Kumar, 2020). The Check Point research, published in September 2019, has revealed a security flaw in modern Android smartphones, which makes them vulnerable to SMS phishing attacks (Artyom Skrobov, 2019). The impact of cyber-crime on the Android platform is deep-rooted, and the open system architecture of Android further exacerbates the problem.

Attackers use.apk files to exploit known vulnerabilities and to penetrate the system (Cebuc, 2020). As.apk files are readily available to Android users, they become an easy source of attacks. Further, there are a few risky features offered to the applications by Google Play Services. Attackers use these Permissions to get into the system. To add to the problem, Google Play Services provides access to some of the features without the user's consent. If one tries to disable them manually, then it affects the functioning of the system, as shown in Fig. 1. The attackers use both inbuilt and user-installed third-party applications to attack the system. Dangerous Permissions requested by these third-party applications at installation time make the system more vulnerable to attacks. Granting such Permissions puts the users' privacy at stake. If the users deny these Permissions, they will not be able to use the application. Once attackers get into the system, they can steal the data or can damage it. As per (Khandelwal, 2019), attackers launch Permission-based attacks in a way where an application with fewer Permissions gain access to the components for which they have not taken Permission to access. After gaining access, attackers use the application to catch sight of user's data. Even the malware developers need Permissions to invoke API calls embedded in the. apk file containing malicious code (Wu et al., 2012). The attackers can get access to the hardware components like camera, microphone, GPS, wifi only if the application that they are using has Permission to use those components (Felt et al., 2011; Mandal et al., 2019). So, Permissions play a vital role among all other static features in malware detection. The privilege-based attacks are the result of the Permission violation by the attackers.

Various detection tools are available to save the system from malicious attacks, which analyze the Android applications and classify them as malicious or benign. This analysis can be Static,

* Corresponding author.
E-mail address: bhatparnika@gmail.com (P. Bhat).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2021.11.004
1319-1578/

Please cite this article as: P. Bhat and K. Dutta, A multi-tiered feature selection model for android malware detection based on Feature discrimination and Information Gain, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2021.11.004

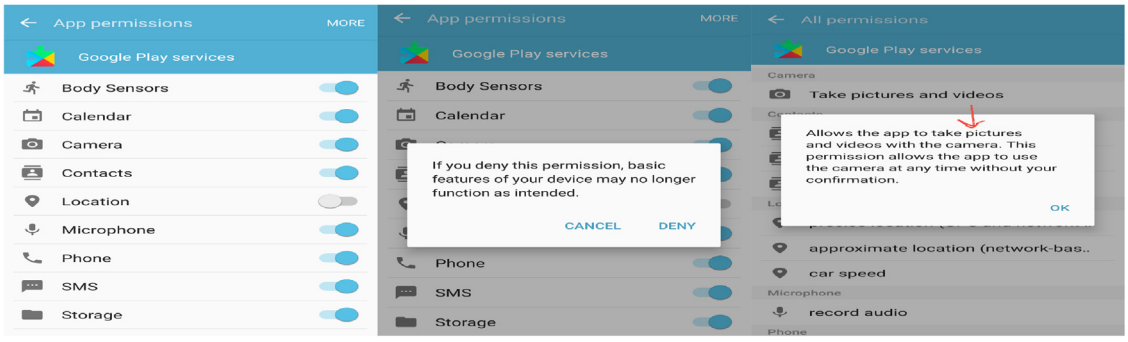

Fig. 1. Default app Permissions provided in Android 6.0.1 (Marshmallow).

Dynamic, or Hybrid. The.apk package contains manifest, resource, and Dalvik executable (dex) files. The static analysis tools extract features like Permissions, Intent, API calls, Hardware components from the manifest and java code file. For dynamic analysis, the detection tools extract features during the execution of the program. Network traffic, system calls, CPU, and memory usage are significant features used in dynamic analysis. Further, the hybrid analysis relies on salient static and dynamic features (Bhat and Dutta, 2019).

Researchers prefer static detection over dynamic as in the static approach, all the static parts of the application are available to be scrutinized. In contrast, dynamic detection is dependent on various runtime factors, such as a specific input as a trigger to demonstrate malicious behavior. In this situation, the static detection approach provides in-depth analysis and better Accuracy than dynamic analysis (Kapratwar et al., 2017; Li et al., 2019). Due to the considerable increase in malicious data, manual detection has become practically impossible. Moreover, many of the malware detection methods that researchers have introduced require a large number of resources. It increases the cost and time complexity to run the detection mechanism on large databases. So, machine learning came into the picture (Yerima et al., 2015). Machine learning algorithms can classify data accurately in less time.

1.1. Motivation

The number of features in the Android file is enormous to the required number of features for optimal classification. Using too many features lays an unnecessary burden on the system. This situation results in high processing time and lowering the overall performance of the system. Researchers have given meticulous attention to devising new methods of malware detection. However, they have done little work in the field of discovering optimal features for malware detection. It is crucial to reduce irrelevant features from the feature set before feeding them to any machine learning algorithm for further processing. The feature selection is an essential part of the malware detection techniques as it can both positively and negatively affect the performance of the detection technique. This paper aims at devising a robust feature set containing relevant features. The filtered feature set can provide in-depth analysis and result in a highly accurate and reliable detection model. The proposed work attempts to discover relevant feature sets using automated feature reduction and a manual testing approach.

1.2. Contribution

The main contribution of this study is to propose a feature selection model which can discover relevant features out of a vast pool. The model is of two parts, where the first part is feature reduction, and the second is testing & analysis on the selected features.

- At first, the approach takes the features as input and feeds them to a multi-tiered feature reduction phase. In the first step, the proposed method removes features based on their frequency, discrimination and, Information Gain scores. Many of the features are not helpful in classification due to various reasons. One crucial reason can be a feature with a low frequency, which cannot improve the result of the classification tool. These features are considered noise, which can also negatively impact the accuracy of the classification tool. Hence, feature selection becomes a pivotal step.
- Further, to analyze the performance of the approach, five machine learning classification approaches were used. Random forest showed the highest Accuracy rate of 96.28%. The method feeds the selected features to multiple machine learning classification techniques using various combinations of the features.
- Finally, the approach ranks feature sets based on the Accuracy, Recall, Precision, F1-Score, and other performance metrics. The features selected can later be used along with malware detection classification approaches to classify and identify whether an Android application is a malicious application or not.

The rest of the paper is as follows: Section 2 discusses the related work. Section 3 illustrates the proposed approach and, Section 4 represents the evaluation metrics, followed by Section 5, presenting experimental results & analysis. Section 6 shows a comparative analysis of the results. Section 7 briefly discusses the current scenario & future research direction and Section 8 concludes the work.

2. Related work

The Android operating system architecture has four layers: (i) hardware layer, (ii) kernel layer, (iii) hardware abstraction layer and, (iv) application layer. Each layer communicates via API calls. Researchers and developers have taken many security measures to ensure the security of these layers and interfaces. They have introduced a sandbox mechanism to secure the components. There are various antivirus and malware detection tools available to protect the system from known and unknown attacks. However, attackers keep on excavating security loopholes present in these layers (Bhat and Dutta, 2019).

Researchers have carried out much work in Android malware detection research using various features. Sahs and Khan (2012) proposed a machine learning approach for malware detection in Android in a static environment. It was a single class SVM model trained on benign samples. The model used built-in Permissions and raw bytecode presented as control flow graph as features for

analysis. The work uses a limited and biased dataset of 2081 benign and 91 malicious files. The nodes in the control flow graphs are labeled based on the last instruction and are small. Such labels fail to capture important information present in the original code.

PUMA, a Permission-based malware detection approach. The dataset comprises 1800 benign applications collected from the unofficial Android Market and 249 malicious samples from the VirusTotal tool. Various machine learning algorithms are applied to the dataset to classify whether the applications are malicious or benign. The approach achieved the highest Accuracy rate of 86.41% using Random Forest classification. Results show a very high false-positive rate with machine learning algorithms. Also, the method relies on only a single feature set for classification (Sanz et al., 2013).

Droid Detective, a rule-based malware detection tool, achieved a 96% Accuracy rate. These rules are a set of Permission combinations extracted using k-map. Any deviation from generated rules indicates malicious behavior. The approach uses a dataset of 741 benign applications from Android Market and 1260 malicious applications from the Malware Genome Project. The work shows how closely Permission combinations and behaviors of malicious applications are related. Results show a high false-positive rate of 50.62% with (k)=1, where k is the frequency of requesting the Permissions (Liang and Du, 2014).

Sharma and Dash (2014) proposed a method that uses Permission and API calls as features. For analysis, the author uses a dataset of 800 malicious files from the Android Malware Genome Project & 800 benign files from official and third-party sources. Correlation-based feature selection is applied to remove the redundant features. For evaluation, the approach uses Naive Bayes and KNN classifiers. KNN achieved the highest Accuracy rate of 95.1%.

Verma and Muttoo (2016) proposed a similar static analysis approach using Permission and Intent as static features. The dataset comprises 850 benign and 620 malicious applications, collected from the Android official application market, Google Play, and contagio mini dump13. The next step applies Information Gain to the dataset for feature selection. For classification proposed approach utilizes K-means clustering and various classification algorithms. The highest Accuracy rate for detecting malicious files is 94%, with a J48 machine learning algorithm for classification.

AndroDialysis uses Intent and Permissions as features for malware detection. The dataset comprises 1846 benign and 5560 malign applications collected from the Google Play store and Drebin. For classification method employs Bayesian Network with four search algorithm-LAGDHillClimber algorithms, Geneticsearch, HillClimber, and K2. The approach achieves the best results with Bayesian Network and LAGDHillClimber combination. Results show that the method achieves an Accuracy rate of 91% with Intent, 83% with Permissions, and 95.5% with both features (Feizollah et al., 2017).

Razak et al., 2018 uses the Particle Swarm Optimization (PSO), Information Gain, and Evolutionary computation method for feature optimization.The approach uses 3500 benign applications from Google Play and 5000 malicious applications from Drebin. Further, the author used various machine learning techniques on the resultant feature set; Random Forest, AdaBoost, KNN, MLP, and J48. Their approach achieved the highest Sensitivity rate with the AdaBoost classifier and Particle Swarm Optimization (PSO), 95.6%, which is less than our approach (97.92% with Random Forest). Also, the AdaBoost classifier shows a high False-Positive Rate 32% and a low F1-Score 87.7%.

Another approach proposed by Shang et al. (2018) uses Permission as features and a Naive Bayes classification algorithm for analysis. For the experiment, the method uses a dataset consisting of 1725 malicious and 945 benign applications. Further assigning weight(w) to the attributes, previously unknown Permissions act as weighting factors for weighting Naive Bayes algorithm. This weight represents the correlation between known and unknown Permission correlation. In the next step, the method calculates the Pearson correlation coefficient for Permissions. The approach removes Permissions with a correlation coefficient value less than a threshold value to improve efficiency. The method achieves an 86.54% detection rate with Naive Bayes and 97.59% with weighted Naive Bayes.

Li et al. (2019) use Permission and API calls as static features. The dataset includes 1,000 malicious and benign applications from VirusShare and Google Store are collected. The next step calculates feature discrimination and selects high discriminating features, and performs frequent pattern mining. Using weighted Naive Bayes and discriminant patterns, the approach detects malicious applications with an Accuracy rate of 88.69%.

Guan et al. (1634 (2020)), uses Permissions and Intent as features to classify the applications. The dataset has 1280 malicious and 1200 benign applications. Datasource for malicious files is AndroMalShare and, MalGenome & non-malicious files are Xiaomi application markets. For feature selection, the method utilizes PSO and Information Gain. The approach applies four machine learning techniques- J48, KNN, SVM, and NB to the dataset to analyze the performance. Without feature selection, the method achieves a 95.2% Accuracy rate with KNN and SVM classifier using both Permissions and Intent as features. Using Permissions as a feature set and applying Information Gain and KNN, the approach achieves an Accuracy rate of 96.1% & with PSO and SVM Accuracy is 96.6%.

Jung et al. (2021), a static analysis approach uses API calls and permissions as features for analyzing the malicious applications. The researchers employ minimal domain knowledge and Gini importance-based methods for feature selection. For classification, the work applies the Random Forest algorithm over 26,276 malicious and 27,041 benign applications. The dataset is collected from AndroZoo and verified using the VirusTotal tool. Random Forest shows an accuracy rate of 96.51/%. Another approach proposed by Sahin et al. (2021) applies linear regression for selecting optimal permission-based features. Various machine learning models are used to classify the dataset comprising of 1000 malicious and benign applications each. MLP(Multilayer Perceptron) model shows the best performance with a 96.1% F-measure score.

Research Gaps

In the proposed work following research gaps are addressed:

- The pre-existing research focuses on improving the accuracy of the malware detection system overlooking how including all the features impacts the computational cost. Machine learning approaches require features to classify a given dataset (Crussell et al., 2012). Using all the features present in the dataset will lead to dimensional disaster. A large number of insignificant and redundant features make the model hard to interpret. The proposed approach applies multiple feature reduction techniques to reduce irrelevant features and improve computational performance.
- It will be easier for the malware developer to evade a detection tool that relies on a single feature set for patterns and insights. A feature set comprising of various features will be more reliable. The proposed work considers the most used feature set; Permissions, API calls, and Intents for analysis.
- It is of utmost importance to figure out the best possible feature sets to improve the overall performance of the detection system. The proposed approach uses the aforementioned static features for the experiment. After removing irrelevant features in the feature selection step, the method runs all possible combinations of the resultant features on five machine learning

approaches. The feature sets are ranked based on performance metrics. The main focus of the work is to achieve a feature set that will provide a high accuracy rate and low false positive alarms.
- The related work does not discuss the impact of using the biased dataset, where the number of malicious files is smaller or more than benign files. The proposed work analysis the performance of the detection system for both biased and unbiased dataset.

The following section discusses the proposed work in detail.

3. Proposed approach

As can be seen from the flowchart in Fig. 2, the proposed work comprises of two phases; feature reduction followed by malware classification using well-defined machine learning algorithms. The feature reduction technique removes the features which are too infrequent to have any impact on the classification. Secondly, the feature discrimination approach reduces the number of feature sets. Further, Information Gain is applied to the reduced features to achieve a set of twenty-one features. Later, this section discusses these processes in detail. Further, as the flowchart depicts, in the second phase, the method applies five machine learning algorithms listed below on all possible combinations of the feature set obtained in the previous step to find the best feature set along with a classification algorithm.

1. Decision Tree(DT)
2. Logistic Regression(LR)
3. Support Vector Machine(SVM)
4. Random Forest(RF)
5. Naive Bayes(NB)

3.1. Dataset Used

The dataset consists of malicious and benign.apk files. The malware is procured from third-party sources; Virustotal (Virustotal, 2019), VirusShare (VirusShare, 2019), and Drebin (Arp et al., 2014) & the benign applications from the play store. Also, the applications were verified using an online tool VirusTotal. The main emphasis during the preparation of the dataset was to ensure the diversity of data.

For analysis, we use dataset (D) comprising 11,449 applications. There are 5279 malicious applications, and 6170 non-malicious apps. For the experiment, we use almost half of the malign (2640) and benign (3085) applications for training (D_t) and the remaining for testing purposes. To analyze the performance of the approach for the biased and unbiased dataset, we divide it into three subsets, as shown in Table 1. The sample size is selected such that there are biased and unbiased subsets of the dataset. D_b is a biased dataset, where the number of malicious files (2640) is more than non-malicious (1543) files. The number of non-malicious files is half the total number of benign files in (D_t). Likewise, D_c consists of a large number of non-malicious applications. The number of malicious files (1320) is half the number of malign files in (D_t). In D_a, the number of malicious (1320) and non-malicious (1543) files is almost equivalent. All the.apk files are selected randomly. The following subsection will discuss the features used in the proposed work.

3.2. Features Used

Android applications comprise many features like Permission, API calls, Intent Filter, Network Address, and Hardware Components. As per the state-of-the-art, Android Permissions are the

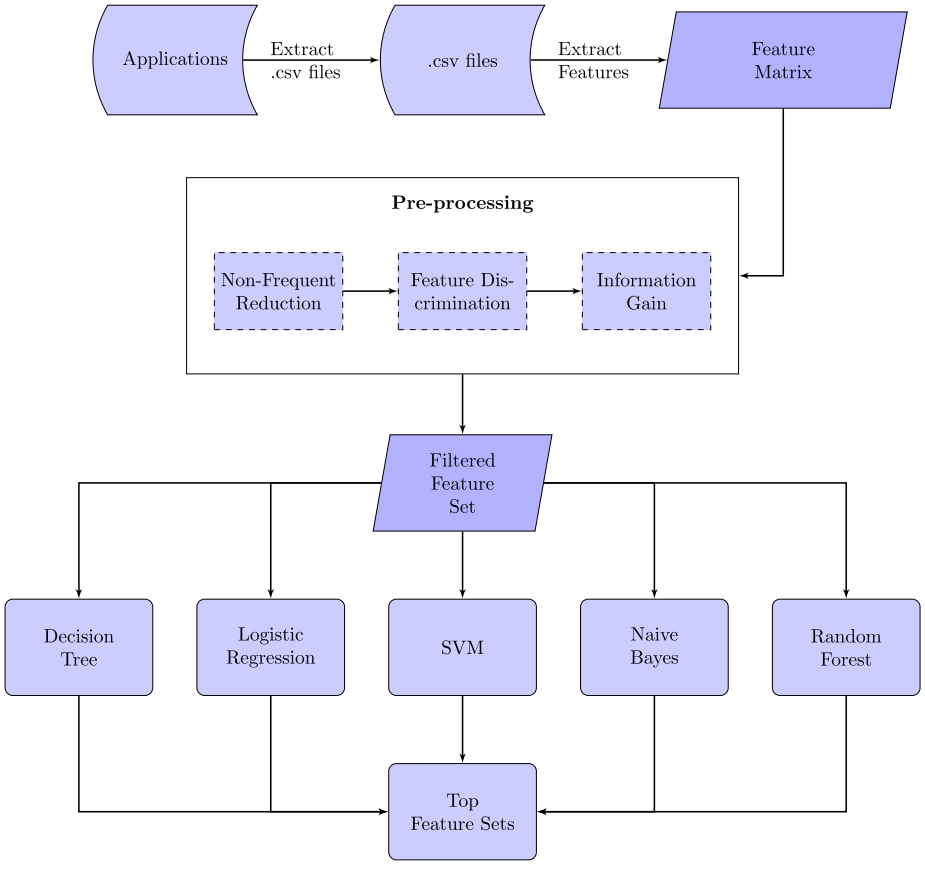

Fig. 2. Flowchart of proposed approach.

Table 1
Subsets of dataset.

S. No.	Dataset	Malicious(M)	Benign(B)
1	D_a	1320	1543
2	D_b	2640	1543
3	D_c	1320	3085

most used feature amongst static features for malware detection. The second most used feature is the java code, which consists of API calls. Another feature set used in this study is Intent (Feizollah et al., 2015; Feizollah et al., 2017). The proposed method uses these three feature sets for malware detection in the static environment.

Permissions

Android is a Linux-based OS so, to gain access to anything in the system, applications need to mention Permission to access that component in the manifest file separately. Three categories of Android Permissions given to the applications are install-time, runtime, and special Permissions. These Permissions define the scope of data accessed and the actions allowed to perform on that data. An application requesting more than the minimal number of Permissions required indicates malicious activity (A. Developer, 2021).

API-Calls

API calls are a set of rules used for calling functions present in a package already installed. Researchers use this feature for the behavioral analysis of Android applications by observing the pattern of the API calls. This information acts as a signature as it is unique for every application. Any deviation from the regular patterns may indicate malicious activity. Researchers prominently use API calls as features for the classification of Android applications (Fernando, 2016).

Intent

Another salient feature is Intent, which researchers use for malware analysis. Android developers use Intents to communicate between application components. Its main use is to launch activities associated with the method. Each Intent carries a pair of information, a) action, b) data on which it has to act. ACTION_ VIEW, ACTION_ DIAL, ACTION_ EDIT are some examples of action data pairs (A. Team, Intents, Android Developer, 2019). Hence, Intents can provide information about the actions that the application is going to perform. Application developers provide details of Intent that the application would receive in the manifest file. By analyzing the pattern of these Intents, it is possible to discover malicious applications.

3.3. Feature extraction

We used an open-source tool (AndroPyTool et al., 2019) to convert Android files into JavaScript Object Notation (JSON) files, and a python code further extracted features out of JSON files into CSV files. The JSON file represents an object's fields as key-value pairs. Each pair shows whether a particular feature is present or not in the application (JSON, 1999). The tool then takes these CSV files as input to perform feature reduction. The first step is to disassemble the.apk files to extract static features from them. The approach stores the extracted features in the feature matrix Fig. 3. Feature matrix consists of the applications as rows and features as columns. The cell value denotes the occurrence of the feature in the respective application.

$$\begin{pmatrix} & F_1 & F_2 & F_3 & F_4 & F_5 & F_6... & F_N \\ Apk_1 & 11 & 15 & 19 & 6 & 0 & 8... & 20 \\ Apk_2 & 21 & 13 & 12 & 18 & 9 & 78... & 22 \\ Apk_3 & 22 & 11 & 30 & 12 & 16 & 12... & 14 \\ . & & & & & & & \\ . & & & & & & & \\ Apk_M & 0 & 9 & 22 & 15 & 12 & 34... & 13 \end{pmatrix}$$

Fig. 3. An instance of a feature matrix.

3.4. Feature reduction

The first phase of the proposed approach consists of a three-step feature reduction approach to filter out uninformative feature sets.

The first step removes the features whose frequency is less than a minimum threshold value of 0.8 3.4.1. The threshold value is selected using the brute force approach. The less frequent features are noise and affect the classification. The next step calculates the feature discrimination score. This step removes the features that are nearly equal in a ratio in malicious and benign files. Such features do not help in classifying application files in between two classes (benign and malign) 3.4.2. To further improve the classification performance, the approach applies Information Gain to remove less informative features 3.4.3.

The stepwise reduction of features strategically removes redundant and insignificant features. And also increases the processing time and lowers the efficiency of the classifier. The resultant features are used along with machine learning approaches to frame an efficient malware detection approach.

3.4.1. Filtering non frequent features

In the first step, the approach filters out features present in a negligible number in both malicious and benign applications. These features are insignificant and won't have any positive effect on the detection technique due to their low frequency.

P, I, and A are three feature sets where 1:

$$\begin{aligned} P &= \text{All the Permissions in the dataset D} \\ I &= \text{All the Intents in the dataset D} \\ A &= \text{All the API calls in the dataset D} \end{aligned} \tag{1}$$

Dataset D represents all the Android applications. P', I', and A' are the resultant feature set 2 3 consisting of features whose frequency is greater than the minimum threshold($Min_{(thresh)}$) value selected.

$$P' \subset P, I' \subset I, A' \subset A \tag{2}$$

$$P' \cup I' \cup A' \subseteq P \cup I \cup A \tag{3}$$

F represent feature itemset 4. All the elements in F have frequency greater than $Min_{(thresh)}$.

$$F = \{f_1, f_2, f_3, ...f_n\}, f_i \in P' \cup I' \cup A' \tag{4}$$

An optimum threshold value of 0.8 is selected. This step filtered out the features that are not present in at least 20% of the malicious or benign database.

3.4.2. Feature discrimination

This step applies the feature discrimination (Li et al., 2019) method to the resultant feature set P', I', and A' to filter out the features that are indiscriminate between the two classes, i.e., malware and benign. It means this method will score the features based on their distribution pattern between malicious and benign applications. This method gives a high score to feature present more or less in one of the classes and a low score to feature with somewhat equal presence in both. Since features with equal distribution in both the classes will not be very informative for the detection technique. The formula to calculate the feature discrimination score for each feature f_i is:(5), (6)

$$Score(f_i) = 1 - \frac{min(f_{ib}, f_{im})}{max(f_{ib}, f_{im})} \quad (5)$$

Where,

$$f_{ib} = \frac{total\ number\ of\ benign\ files\ with\ feature\ f_i}{total\ number\ of\ benign\ files}$$
$$f_{im} = \frac{total\ number\ of\ malicious\ files\ with\ feature\ f_i}{total\ number\ of\ malware\ files} \quad (6)$$

f_{ib} represents frequency of feature f_i in benign files and, f_{im} represents frequency of feature f_i in malicious files

$Score(f_i) \in \{0, 1\}$

When:

$Score(f_i)$ = 0 {equal frequency of occurrence in both the classes; no discrimination}

$Score(f_i) \approx 0$ {low frequency of occurrence in either of the classes; worst discriminating feature}

$Score(f_i) \approx 1$ {high frequency of occurrence in either of the classes; best discriminating feature}

Calculate the degree of discrimination or score for each feature f_i, the elements with the highest score from each feature set P', I', and A', are selected. Finally, the elements in the resultant feature set P'', I'' and A'' 7 have a high frequency of occurrence in either of the two classes(benign or malicious).

$$P'' \cup I'' \cup A'' \subseteq P' \cup I' \cup A' \quad (7)$$

This step resulted in a nearly 80% reduction of features from initial numbers.

3.4.3. Information Gain

The last feature reduction technique used is Information Gain. It is a filtering method that filters the features based on the classification information it has. Information gain is also known as a variable selection measure. It provides attributes (features) that best discriminates between two classes (malicious and benign) in a given training dataset.

The information here is entropy. Information Gain, also known as Kullback–Leibler divergence or relative entropy, is the amount of statistical information acquired about a target variable from a predictor variable. Information Gain is a feature selection method used to remove irrelevant features. It is independent of the machine learning algorithm used and needs less time to calculate the values of variables. It plays a pivotal role in improving the classification performance of machine learning algorithms. The approach helps in filtering out the redundant as well as irrelevant feature vectors from the dataset (Quinlan, 1986).

The approach calculates Information Gain for the remaining elements in the resultant set P'', I'', and A''. A higher score indicates the feature is informative and, a lower score shows it is less informative, hence irrelevant for classification.

Information Gain(IG) value for a feature is calculated as (8)

$$IG(f_i) = E(Ismalware) - E(Ismalware, f_i) \quad (8)$$

Larger value of $IG(f_i)$ indicates that f_i is of utmost importance for classification.

E(Ismalware)- Entropy of the Ismalware (class) is calculated as (9):

$$E(Ismalware) = -\sum_{i=1}^{c} p_i \log_2 p_i \quad (9)$$

where c = Total number of classes, in this case c =2 (benign and malware) &

p_i= Probability of occurrence of the class.

E(Ismalware,f_i)- Conditional entropy of the Ismalware (class) given feature f_i is calculated as (10):

$$E(Ismalware, f_i) = p(f_i)E(Ismalware \mid f_i) + p(\overline{f_i})E(Ismalware \mid \overline{f_i}) \quad (10)$$

The value of IG varies from 0 to 0.5.

The approach selects seven features with the highest gain score from each type of feature set. Table 2 provides a list of the twenty-one top features. This feature set 11 consists of the best features from a diverse feature pool. The combination of different types of features will provide an in-depth and reliable analysis.

$$P''' \cup I''' \cup A''' \subseteq P'' \cup I'' \cup A'' \quad (11)$$

The first phase filters twenty-one best features out of 21,149 features using three feature reduction methods. In the next section analysis of the results and the most significant feature set are presented.

4. Evaluation Metrics

For evaluating the performance of the approach following, evaluation metrics are used (Bhat and Dutta, 2021):

Accuracy 12 represents the ratio of correctly classified applications to the total number of benign and malicious applications present in the test dataset.

$$A = \frac{TP + TN}{TP + FP + TN + FN} \quad (12)$$

Recall(Sensitivity)13 also known as detection rate, represents the ratio of correctly classified malicious applications to the total number of malicious applications present in the test dataset (Anael Beaugnon, 2018; Shrivastav, 2020).

$$R = \frac{TP}{TP + FN} \quad (13)$$

Precision 14 represents the ratio of actual malicious files detected to the total number of applications predicted to be malign by the proposed approach.

$$P = \frac{TP}{TP + FP} \quad (14)$$

F1-Score 15 represents the harmonic mean of Precision and Recall.

$$FS = \frac{2 * P * R}{P + R} \quad (15)$$

Specificity(TNR) 16 represents the ratio of correctly classified benign applications.

$$Specificity(TNR) = \frac{TN}{TN + FP} \quad (16)$$

False-negative rate 17 represents the ratio of malicious applications misclassified as benign to the total number of malicious applications in the test dataset.

$$False\ Negative\ Rate(FNR) = \frac{FN}{TP + FN} \quad (17)$$

False-positive rate 18 represents the ratio of benign applications misclassified as malicious to the total number of benign applications in the test dataset.

$$False\ Positive\ Rate(FPR) = \frac{FP}{FP + TN} \quad (18)$$

Matthews correlation coefficient (MCC) 19 is the best measure to evaluate the performance when the test dataset is unequal. Value of MCC ranges between $(+1, -1)$.

Table 2
Top 21 features with highest Information Gain (IG) score.

N	*Features*	*IG Score*	*Abbreviation*
1	android.permission.SEND_ SMS	0.359	P1
2	android.permission.RECEIVE_ SMS	0.321	P2
3	android.permission.READ_ SMS	0.297	P3
4	android.permission.READ_ PHONE_ STATE	0.267	P4
5	android.permission.WRITE_ SMS	0.233	P5
6	com.google.android.c2dm.permission.RECEIVE	0.163	P6
7	android.permission.MOUNT_ UNMOUNT _ FILESYSTEMS	0.155	P7
8	com.android.vending.INSTALL_ REFERRER	0.251	I1
9	com.google.firebase.INSTANCE _ ID _ EVENT	0.237	I2
10	com.google.android.c2dm.intent.RECEIVE	0.201	I3
11	android.provider.Telephony.SMS _ RECEIVED	0.196	I4
12	android.intent.action.USER _ PRESENT	0.144	I5
13	com.google.firebase.MESSAGING_ EVENT	0.116	I6
14	android.provider.Telephony.SMS_ DELIVER	0.106	I7
15	java.util.TimeZone.getAvailableIDs	0.519	A1
16	android.view.ViewGroup.indexOfChild	0.517	A2
17	android.content.pm.PackageManager.getPackageInstaller	0.507	A3
18	android.os.ParcelFileDescriptor	0.498	A4
19	java.nio.BufferOverflowException	0.495	A5
20	android.view.View.getDisplay	0.495	A6
21	android.support.v7.view.ActionMode.getTitle	0.490	A7

Where, +1= perfect prediction.
0 = random prediction
−1 =disagreement between observation and prediction (Matthews, 1975)

$$Matthews\ correlation\ coefficient(MCC) = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (19)$$

where,

True Positive(TP)-Represents the number of malicious applications correctly classified as malicious.

True Negative(TN)-Represents the number of benign applications correctly classified as benign.

False Positive(FP)-Represents the number of benign applications incorrectly classified as malicious.

False Negative(FN)-Represents the number of malicious applications incorrectly classified as benign.

Table 3 represents the confusion matrix for each machine learning algorithm in the case of D.

5. Results and Analysis

This section discusses the results obtained using the selected features with multiple machine learning classification approaches to figure out top feature sets. The method takes all 21 features arrived in the previous step and runs all possible combinations of 10 features on five machine learning approaches.

Hardware used to perform this experiment is a Core i5 processor with 8 GB RAM and a hard disk of 512 GB storage capacity. Considering the limited hardware resources and as $^{21}C_{10}$ results in 3,52716 combinations, the figure of 10 features is selected so that the number is neither too low nor too high. In future experiments can be done using the proposed framework with a different number of features. Tables 4–7 shows results for all the subsets.

5.1. Decision Tree

The Decision Tree is a supervised classification method. It splits the data into partitions using binary recursive partitioning. In the output tree, the nodes represent the classification test, and the branches represent the outcome of this test. These branches connect to the next node or the leaf node that represents the outcome of classification (Pedregosa et al., 2011; Sharma and Kumar, 2016). The Decision Tree shows a 96.28% Accuracy rate and 97.40% Recall rate. The value of FNR and FPR is 3.86% and 12.24%, respectively.

The execution time for the Decision Tree to classify and find the best possible combination of feature sets was 90 s.

5.2. Logistic Regression

The third classification model used in this study is Logistic Regression. It is a supervised classification method used for predictive analysis that uses a linear regression equation to provide a discrete binary output (Karthi et al., 2015). The Logistic Regression classification approach resulted in 95.30% Accuracy and 97.92% Recall rate. Also, the method achieves a 5.82% FNR and 13.82% FPR rate. The execution time for Logistic Regression was 734 s.

5.3. Support Vector Machine

The second classification model used in this paper is SVM. SVM is a widely used supervised classification method in malware detection research.

SVM relies on statistical learning theory (Evgeniou and Pontil, 2001). SVM was able to achieve an Accuracy rate of 95.30% and 97.08% Recall rate. Also, the value of FNR is 5.30% and, FPR is 10.83%. It took 3,111 s for SVM to classify and find the best possible feature set.

5.4. Random Forest

Another classification model used in this study is Random Forest. Leo Breiman proposed the Random Forest in the 2000's "to build a predictor ensemble with a set of Decision Trees" (Biau, 2012). It uses randomly selected subspaces of data from the dataset to grow Decision Trees, and the final prediction is obtained by aggregating over the ensemble. The Random Forest classification approach shows a 96.28% Accuracy, 97.92% Recall, 3.47% FNR, and 12.74% FPR rate with a 100 optimal number of trees. The execution time was 454 s.

5.5. Naive Bayes

The Naive Bayes classifier is the final classification model used in the study. It is a simplified bayesian probability model based on "the assumption that the features are independent of each

Table 3
Confusion matrix for each machine learning algorithm(D).

Decision Tree

		Actual		
		Positive	Negative	Total
Predicted	Positive	2717	367	3084
	Negative	109	2632	2741
	Total	2826	2999	N

Support Vector Machine

		Actual		
		Positive	Negative	Total
Predicted	Positive	2770	314	3084
	Negative	155	2586	2741
	Total	2925	2900	N

Logistic Regression

		Actual		
		Positive	Negative	Total
Predicted	Positive	2671	413	3084
	Negative	165	2576	2741
	Total	2836	2989	N

Random Forest

		Actual		
		Positive	Negative	Total
Predicted	Positive	2698	386	3084
	Negative	97	2644	2741
	Total	2795	3030	N

Naive Bayes

		Actual		
		Positive	Negative	Total
Predicted	Positive	2715	369	3084
	Negative	183	2558	2741
	Total	2898	2927	N

Table 4
Performance comparison of various machine learning algorithms using D.

ML	*Accuracy*	*Recall*	*Precision*	*F1 – Score*	*TNR*	*FNR*	*FPR*	*MCC*
DT	96.18%	97.40%	89.20%	93.12%	87.76%	3.86%	12.24%	0.840
LR	95.30%	97.92%	88.75%	93.11%	86.18%	5.82%	13.82%	0.804
SVM	95.30%	97.08%	89.53%	93.15%	89.17%	5.30%	10.83%	0.840
RF	96.28%	97.92%	89.40%	93.47%	87.26%	3.47%	12.74%	0.838
NB	95.30%	95.91%	90.06%	92.89%	87.39%	6.31%	12.61%	0.812

Table 5
Performance comparison of various machine learning algorithms using D_a.

ML	*Accuracy*	*Recall*	*Precision*	*F1 – Score*	*TNR*	*FNR*	*FPR*	*MCC*
DT	96.19%	97.72%	94.22%	95.94%	93.04%	4.84%	6.9%	0.882
LR	95.14%	97.95%	92.8%	95.31%	91.30%	5.32%	8.70%	0.861
SVM	95.4%	96.8%	96.8%	95.73%	93.31%	5.30%	5.30%	0.881
RF	96.15%	98.02%	93.21%	95.56%	91.32%	4.85%	6.68%	0.885
NB	95.21%	96.66%	93.53%	95.07%	92.21%	5.71%	7.09%	0.865

Table 6
Performance comparison of various machine learning algorithms using D_b.

ML	*Accuracy*	*Recall*	*Precision*	*F1 – Score*	*TNR*	*FNR*	*FPR*	*MCC*
DT	95.40%	94.23%	93.26%	93.74%	96.08%	4.11%	3.92%	0.901
LR	93.97%	94.53%	93.00%	94.18%	95.84%	5.81%	4.16%	0.871
SVM	95.45%	93.44%	94.29%	93.87%	96.64%	5.56%	3.36%	0.902
RF	94.10%	96.10%	95.10%	93.72%	96.40%	3.50%	3.84%	0.906
NB	93.53%	93.07%	94.42%	91.51%	96.60%	6.50%	3.39%	0.864

other," implying that the probability of occurrence of one of the features is independent of the probability of occurrence of other features (Mukherjee and Sharma, 2012). The Naive Bayes is known to give good results in practical situations. The Naive Bayes classification method achieved an Accuracy and Recall rate at 95.30% and 95.91%, respectively. Also, FNR and FPR rates are 6.31% and 12.61%. It took Naive Bayes 89 s to complete the execution.

Furthermore, we use an SMS malware dataset (D_{sms}) (Mahdavifar et al., 2020) to analyze the performance of the approach for specific malware categories. It is a publically available dataset (CICMalDroid2020) having 3904 SMS malware and 1795 benign files. Table 8 shows that the proposed method achieved the highest Accuracy rate of 95.02% and MCC value of 0.900 with Random Forest.

Table 7
Performance comparison of various machine learning algorithms using D_c.

ML	*Accuracy*	*Recall*	*Precision*	*F1 – Score*	*TNR*	*FNR*	*FPR*	*MCC*
DT	93.96%	97.51%	93.96%	95.70%	86.06%	2.49%	11.94%	0.857
LR	93.77%	96.81%	94.42%	95.60%	85.75%	3.19%	13.25%	0.850
SVM	95.43%	95.80%	95.25%	95.73%	88.03%	2.85%	10.20%	0.860
RF	94.10%	97.45%	94.22%	95.81%	86.56%	2.55%	13.44%	0.859
NB	93.22%	97.16%	93.25%	95.17%	86.55%	2.85%	13.45%	0.840

The Accuracy, Recall, Precision, and Specificity(TNR) rate achieved in all the cases is high. Also, the false-negative and the false-positive rate is low. The execution time includes the time taken to run all possible combinations as well. Naive bayes perform best in terms of time complexity. The time taken by SVM is high as compared to other classification approaches. The execution time of Random Forest is high compared to Logistic Regression and lower than the Decision Tree. The Random Forest outperforms as it has a higher Accuracy rate.

The value of MCC is close to 1 in all the cases. It implies that the approach performs well even when the dataset is unbalanced, for example, D_b and D_c. D_a represents an almost balanced dataset, which is an ideal case. Further, this work discusses the results of dataset D in detail.

Table 9, represents some of the features that are common for multiple approaches showing their importance. The results also show that a few features are ubiquitous and they are significant for malware detection. Table 10 represents the best feature set with five approaches; hence, we have termed these features as MIFs (Most Important Features). These can be used with other classification approaches to achieve a highly accurate and reliable malware detection approach.

A platform-independent approach will be of much significance as malicious attacks are not limited to the Android operating system. The researchers are working in the area of malware detection on different operating systems. Haddadpajouh et al. (2018), Cimitile et al. (2017) have applied Information Gain for feature selection and machine learning algorithms for malware detection to Mac OS and iOS. In the future, researchers can use the proposed multi-tiered feature selection approach on different platforms to improve the malware detection and analyze the performance.

6. Comparative Analysis

Fig. 4 and 5 show the comparative analysis of results obtained using various feature sets for all the subsets of the dataset. Best Accuracy was achieved at 96.28% (D) using Random Forest classification. One of the prime reasons Random Forest yields stable and accurate results is that it is an ensemble approach that merges the predictions of correlated Decision Trees. The method generates multiple correlated Decision Trees and applies the bagging method for training. Further, a majority voting scheme amalgamates the prediction of individual trees to generate the final output (Donges, 2021).

Random forest and Logistic Regression provided the highest Recall at 97.92% (D). Further, Naive Bayes achieved the highest Precision score at 90% (D). Classification approaches show similar results for D_b when malicious files are more than benign files.

Fig. 6 and 7 represent the comparative analysis between D and D_{sms}. The Accuracy and Recall value with Random Forest is higher for dataset D while the value of MCC is nearly equal to 1 for D_{sms}. Also, the value of the F1-Score for D_{sms} is 95.21%, which is moderately greater than D(93.47%) with Random Forest.

Overall results show that the approach performs well for all the cases, i.e., when there is a biased dataset and even for a particular class of malware (D_{sms}).

Table 11 provides a comparative analysis of performance with features discovered in this work and related work. The proposed Optimal Static Feature Set (OSFS) can detect malware with higher Accuracy. These results justify that the OSFS discovered using the proposed approach can be used in malware detection to improve its performance.

7. Discussion and Future Work

This section briefly discusses the emerging threats against the Android platform and future research direction to improve the performance of the existing Android detection mechanism.

The rapid evolution of zero-day-malware families has become a challenging security issue for Android. The detection mechanism needs to analyze the cyber code and extract information to protect the system from attacks. Coulter et al. (2020) propose a data-driven cyber security (DDCS) framework to analyze the cyber code. The work provides a guideline for improving the machine learning algorithms to protect the system from attacks like zero-day vulnerabilities. The first component of the framework is data processing. The paper emphasizes how a balanced and well-defined dataset impacts the performance of the machine learning algorithms. Our approach uses a well-defined and labeled dataset, Drebin. The correctness of the dataset is also verified using the VirusTotal tool, as suggested in the paper. Also, the proposed feature reduction model eliminates unnecessary and irrelevant data. The second component is feature engineering, which focuses on using essential features for classification and feature representation. In our approach, we use three features, permission, API calls, and intent for analysis. A feature matrix represents the occurrence of these features in the applications.

In the future, we will use techniques like Abstract Syntax Tree (AST) (Lin et al., 2017) and Extreme Programming (Holzinger et al., 2005) for representing programs. Such methods will help to analyze code and detect zero-day attacks. The third component of the framework is cyber modeling that discusses various metrics

Table 8
Performance comparison of various machine learning algorithms using D_{sms}.

ML	*Accuracy*	*Recall*	*Precision*	*F1 – Score*	*TNR*	*FNR*	*FPR*	*MCC*
DT	94.08%	95.92%	92.45%	94.15%	92.26%	7.74%	4.08%	0.882
LR	93.15%	92.58%	93.26%	93.41%	93.77%	6.23%	7.42%	0.862
SVM	94.08%	93.47%	95.17%	94.31%	94.75%	5.25%	6.53%	0.881
RF	95.02%	94.36%	96.07%	95.21%	95.74%	4.26%	5.64%	0.900
NB	92.09%	91.44%	93.15%	92.28%	92.79%	7.21%	8.56%	0.841

Table 9
Top 10 features with five machine learning algorithm (D).

ML	*F1*	*F2*	*F3*	*F4*	*F5*	*F6*	*F7*	*F8*	*F9*	*F10*
DT	P2	P3	P4	P7	I8	I3	A1	A4	A6	A7
LR	P1	P2	P4	P7	I2	A2	A3	A5	A6	A7
SVM	P1	P2	P4	P7	I1	I2	I7	A1	A6	A7
RF	P2	P4	P6	P7	I1	I5	A1	A4	A6	A7
NB	P1	P2	P4	P6	I1	I3	A1	A4	A6	A7

Table 10
Most Important Features(MIFs) (D).

SNo	*Features*	*Abbreviation*
1	android.permission.RECEIVE_ SMS	P2
2	android.permission.READ_ PHONE_ STATE	P4
3	android.permission.MOUNT_ UNMOUNT _ FILESYSTEMS	P7
4	java.util.TimeZone.getAvailableIDs	A1
5	android.view.View.getDisplay	A6
6	android.support.v7.view.ActionMode.getTitle	A7

used to evaluate the performance of models. In the proposed work, other than TP, FP, TN, and FN metrics, TPR and FPR values are also considered for analyzing the performance of machine learning models using various performance metrics. To address the issue of an imbalanced dataset, we use subsets of the dataset to analyze the performance of algorithms. Also, the value of MCC evaluation metrics shows how the multi-tiered feature reduction model improves the performance of the algorithms even when the dataset is imbalanced.

Machine learning algorithms work efficiently on a smaller to medium and labeled dataset. The models are easy to interpret and has limited resource requirement. The processing and training time ranges from seconds to hours. Such models are well suited for lightweight applications. However, the massive amounts of data make it challenging to secure the system from potential unknown attacks.

Deep learning algorithms will play a key role in analyzing and interpreting a large volume of data. Qiu et al. (2020) discusses various deep learning algorithms that can abstract complex behavior from a large number of Android applications. Deep learning algorithms can efficiently protect the system from a previously unknown malware as these algorithms can interpret from an unlabelled dataset. Qiu et al. (2019) proposes a method A3CM (Automatic Capability Annotation for Android Malware) that automatically detects and annotates previously unknown malware. The approach extracts semantic features and uses numerical vectors to represent the features. For classification method uses SVM, DT, and DNN (Deep Neural Network), DNN shows higher performance with A3CM for multi-class malware family classification. Their approach relies on a limited dataset of 6,899 annotated malicious samples. Also, tuning the parameters of DNN lowers the performance of the system.

The expanse of Android usage for handling the security of IoT (Internet of Things) devices and CPS (Cyber-Physical Systems) has also increased the rate of sophisticated malicious attacks against Android devices. To enhance the security of such devices requires analyzing the behavior of Android applications in a real-time environment. SideNet, a deep learning model proposed by Ma et al. (2021), analyzes sensitive Android application behaviors. The model is a combination of Encoder (Universal Encoder for Time

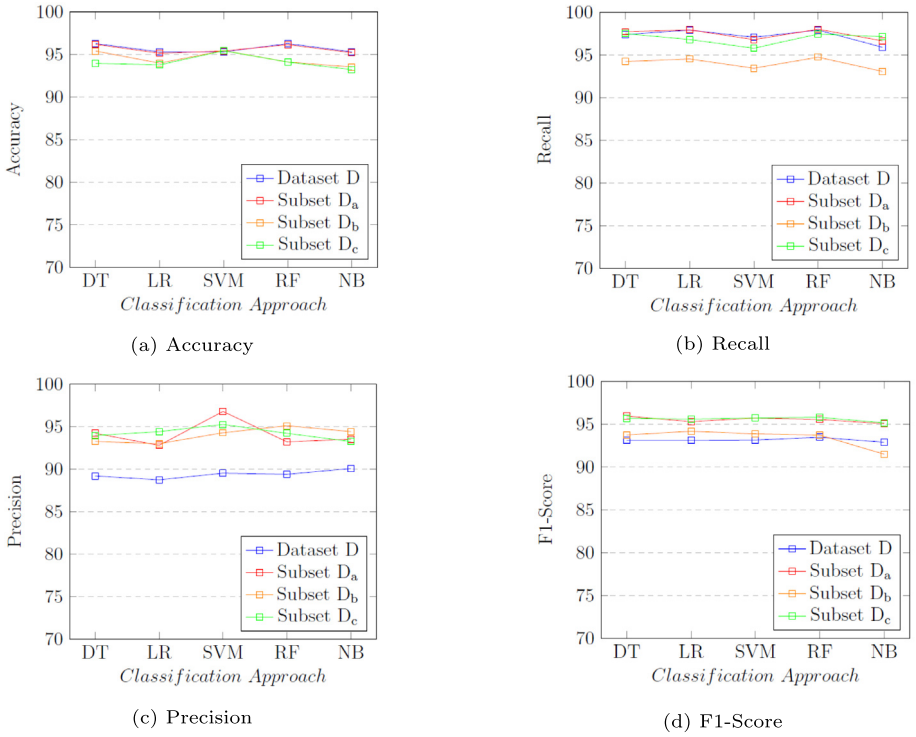

Fig. 4. Comparative analysis of Accuracy, Recall, Precision and, F1-Score of D & three subsets.

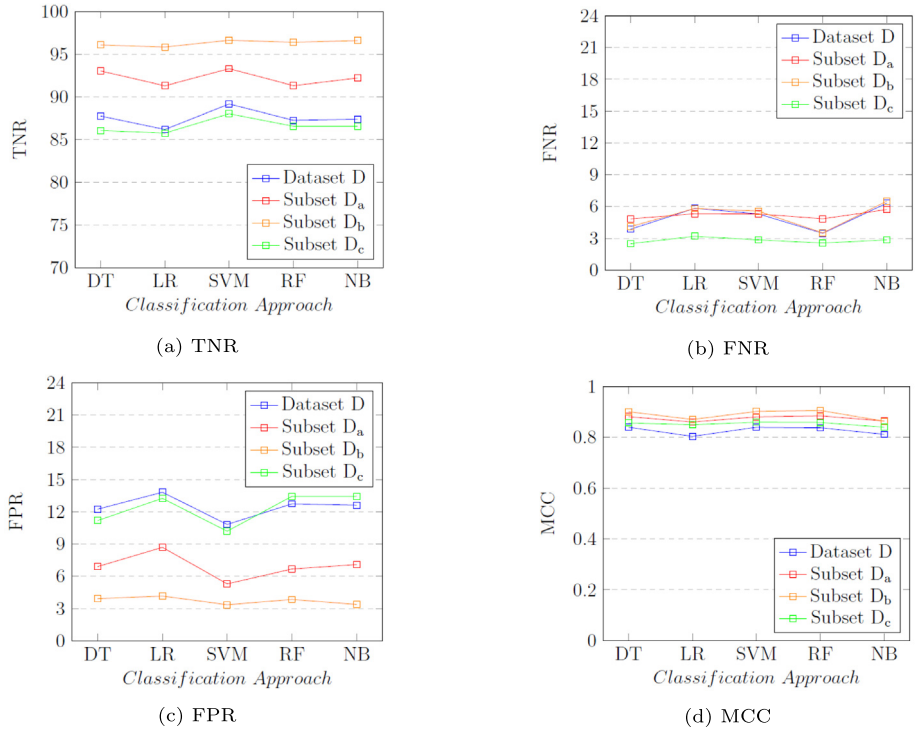

Fig. 5. Comparative analysis of Specificity, FNR, FPR and, MCC of D & three subsets.

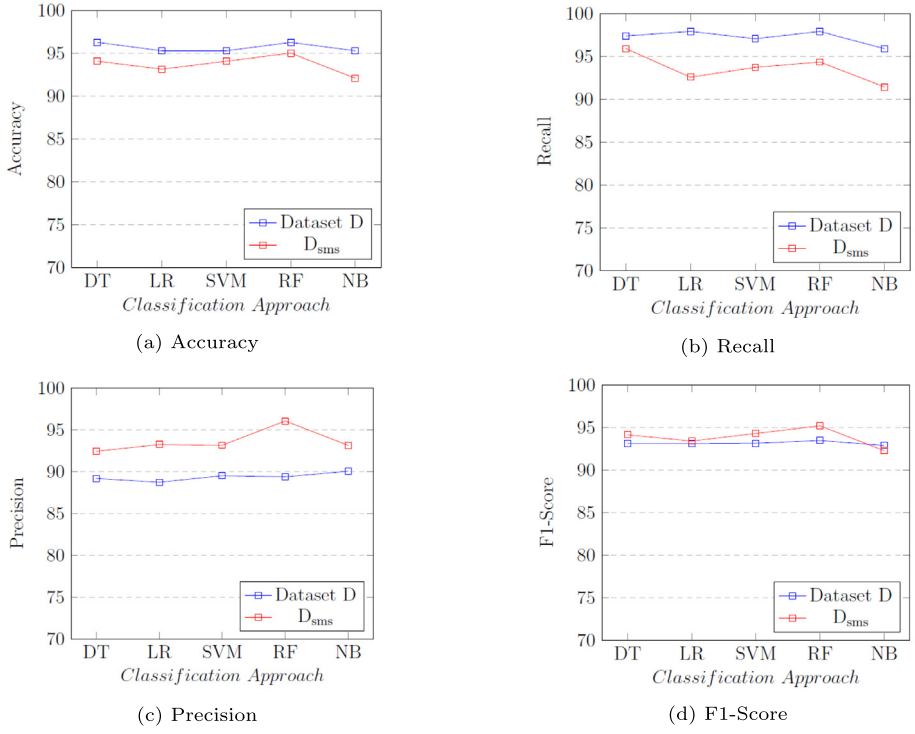

Fig. 6. Comparative analysis of Accuracy, Recall, Precision and, F1-Score of D & D_{sms}.

Series) (Serrà et al., 2018) and ResNet (residual learning framework) (He et al., 2015). The sparse learning approach is employed to reduce the number of parameters and the training time of the model (Dettmers and Zettlemoyer, 2019). The author uses 15 real-time applications to analyze the performance of the SideNet. The model showed a significant increase in the accuracy rate and efficiency in comparison to the existing models. The model requires offline devices (side-channel observer) and a customized Android Open Source Project (AOSP) to collect sensitive information for analysis. The SideNet requires a considerable number of

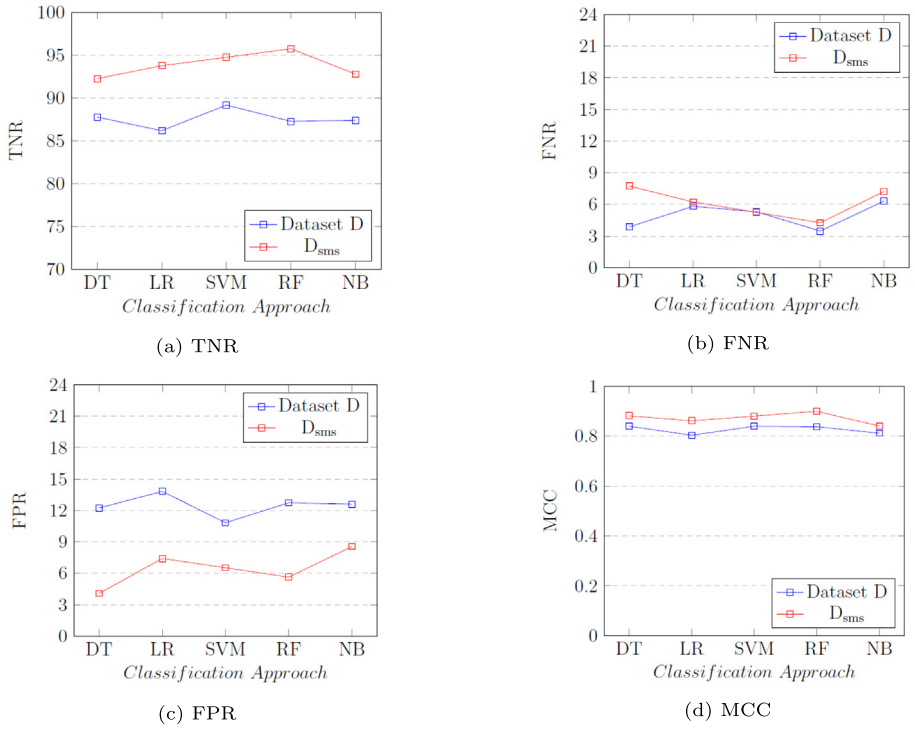

Fig. 7. Comparative analysis of Specificity, FNR, FPR and, MCC of D & D_{sms}.

Table 11
Comparative analysis with related work.

Author	*Features*	*Performance*	*Dataset*	*Remarks*
Sahs et al. (2012)	Permissions, Bytecode	Precision and F1-score decreases with the increase in benign files	2081 benign and 91 malicious applications	Biased dataset; High false negative rate.
Sanz et al. (2013)	Permissions	86.41% Accuracy rate	1811 benign and 249 malicious applications	Biased dataset; Low Accuracy rate compared to the proposed approach.
Liang and Du (2014)	Permission Combination	96% Accuracy rate; 88% benign files recognition rate	741 benign and 1260 malicious applications	Limited dataset; High false positive ratio.
Sharma et al. (2014)	Permissions, API calls	95.1% Accuracy rate	800 benign and 800 malicious applications	Limited dataset
Verma et al. (2016)	Permissions, Intent	94% Accuracy rate	850 benign and 620 malcious applications	Smaller dataset for a reliable classification.
Feizollah et al. (2017)	Permissions, Intent	91% Accuracy rate using Android intent and 83% using permission	1846 benign and 5560 malicious applications	Biased dataset
Razak et al. (2018)	Permissions	95.6% Sensitivity rate	3500 benign and 8000 malicious applications	Low F1_ score compared to the proposed approach.
Shang F. et al. (2018)	Permissions	86.54% detection rate with Naive Bayes and 97.59% with weighted Naive Bayes	945 benign and 1725 malicious applications	Limited dataset; Low detection rate compared to the proposed approach.
Li et al. (2019)	Permissions, API calls	88.69% Accuracy rate	1000 benign and 1000 malicious applications	Limited dataset; Low Accuracy rate compared to the proposed approach.
Guan et al. (2020)	Permissions, Intent	96.6% Accuracy rate	1200 benign and 1280 malicious applications	Limited dataset; Using large number of features with PSO results in increase in time complexity.
Jung et al. (2021)	Permissions, API calls	96.51% Accuracy rate	27,041 benign and 26,276 malicious applications	Balanced dataset; Slightly high Accuracy and low Recall rate compared to the proposed approach.
Sahin et al. (2021)	Permissions	96.1% F1-Score rate	1000 benign and 1000 malicious applications	Limited dataset
OSFS	Permissions, Intent and API calls	96.28% Accuracy rate and 97.92% Recall rate	6170 benign and 5279 malicious applications	High false positive rate

resources to operate, which increases the cost and complexity of the system. Also, a deep learning approach to work efficiently requires a large dataset for analysis, increasing the time complexity of the system. The hardware dependency and a large amount of data increase the overall cost of the system and impact the performance.

Researchers employ deep learning models to detect a wide range of attacks against the Android platform but at the cost of increased complexity, cost and time. There is a need to address certain factors while employing deep learning algorithms for analysis. The dataset should be large enough for an algorithm to come to an interpretation. The deep learning model requires optimizing sev-

eral parameters that slow down the training process and increase the processing time of the deep-learning model. A large dataset will increase the number of features to be analyzed (Flair, 2018; Naumenko, 2020). In the future, we will employ the proposed feature reduction approach on a deep learning model to remove the number of insignificant features. Reducing the number of features to train a deep-learning model will significantly decrease the training time and improve the performance.

8. Conclusion

This study proposes a method to discover features that provide high accuracy in malware detection. The proposed method uses feature reduction techniques to weed out irrelevant features.

The stepwise reduction ensured that only informative and significant features are remaining. These top features are used with multiple classification approaches to test the effectiveness of the resultant features. The method achieves an Accuracy as high as 96.28%, Recall 97.92%, and F1-Score of 93.47% using the Random Forest machine learning approach. The comparison with related work shows that the proposed work with a reduced feature set provides high Accuracy. These results surmise that Permissions and API calls can play an essential part in static malware detection approaches. There is a variation in the best feature set for each classification method. The results show that the malware detection tools cannot focus on any of the single aspects independently. An equal emphasis must be placed on features and classification approaches as both seem dependent on each other.

In future work, it is possible to experiment with more features. Researchers can use the feature set discovered in this paper for better insights into the efficacy of the results.

Funding

This work is the results of the research funded by the Ministry of Human Resource Development

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

A. Developers, Permissions on android, Permissions, Available: URL:https://developer.android.com/guide/topics/permissions/overview (2021).

Anael Beaugnon, A.H., 2018. Pierre Collet, Detection performance metrics, Detection Rate, Available: URL:https://anssi-fr.github.io/SecuML/miscellaneous.detection_perf.html#::text=The%20confusion%20matrix%20allows%20to,PTP%2BFN.

AndroPyTool, Andropytool, GitHub, Available:URL:https://github.com/alexMyG/AndroPyTool (2019).

Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., Rieck, K., 2014. Drebin: Effective and explainable detection of android malware in your pocket, Symposium on Network and Distributed System Security (NDSS) (02 2014). doi:10.14722/ndss.2014.23247.

Artyom Skrobov, S.M., 2019. Advanced sms phishing attacks against modern android-based smartphones, Check Point Software Technologies LTD, Available: URL:https://research.checkpoint.com/advanced-sms-phishing-attacks-against-modern-android-based-smartphones/.

A. Team, Intents, Android Developer, Available: URL:https://developer.android.com/reference/android/content/Intent (2019).

Bhat, P., Dutta, K., 2019. A survey on various threats and current state of security in android platform. ACM Comput. Surveys 52, 1–35. https://doi.org/10.1145/3301285.

Bhat, P., Dutta, K., 2021. Cogramdroid-an approach towards malware detection in android using opcode ngrams, Concurrency and Computation: Practice and Experience n/a (n/a) e6332. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.6332, doi: 10.1002/cpe.6332. URL:https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.6332.

Biau, G., 2012. Analysis of a random forests model, J. Mach. Learn. Res. 13 (1) (2012) 1063–1095, URL:http://dl.acm.org/citation.cfm?id=2503308.2343682.

Cebuc, E., 2020. How are we doing with android's overlay attacks in 2020?, F-Secure, Available: URL:https://thehackernews.com/2020/05/stranhogg-android-vulnerability.html.

Cimitile, A., Martinelli, F., Mercaldo, F., 2017. Machine learning meets ios malware: Identifying malicious applications on apple environment (02 2017).

Coulter, R., Han, Q.-L., Pan, L., Zhang, J., Xiang, Y., 2020. Code analysis for intelligent cyber systems: A data-driven approach, Information Sciences 524 (03 2020). doi:10.1016/j.ins.2020.03.036.

Crussell, J., Gibler, C., Chen, H., 2012. Attack of the clones: Detecting cloned applications on android markets. In: Foresti, S., Yung, M., Martinelli, F. (Eds.), Computer Security – ESORICS 2012. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 37–54.

Dettmers, T., Zettlemoyer, L., 2019. Sparse networks from scratch: Faster training without losing performance, CoRR abs/1907.04840 (2019). arXiv:1907.04840. URL:http://arxiv.org/abs/1907.04840.

Donges, N., 2021. A complete guide to the random forest algorithm, All you need to know about the random forest model., Available: URL:https://builtin.com/data-science/random-forest-algorithm (2021).

Evgeniou, T., Pontil, M., 2001. Support vector machines: Theory and applications, in: Machine Learning and Its Applications, Springer-VerlagBerlin, Heidelberg, 2001.

Feizollah, A., Anuar, N.B., Salleh, R., Wahab, A.W.A., 2015. A review on feature selection in mobile malware detection. Digit. Investig. 13 (C), 22–37. https://doi.org/10.1016/j.diin.2015.02.001.

Feizollah, A., Anuar, N., Salleh, R., Suarez-Tangil, G., Furnell, S., 2017. Androdialysis: Analysis of android intent effectiveness in malware detection. Computers Security 65, 121–134. https://doi.org/10.1016/j.cose.2016.11.007.

Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D., 2011. Android permissions demystified. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 627–638. https://doi.org/10.1145/2046707.2046779.

Fernando, J., 2016. What is an api? how to call an api from android?, DroidMentor, Available: URL:https://droidmentor.com/api-call-api-android/.

Flair, D., 2018. Deep learning vs machine learning - demystified in simple words, Blog, Available: URL:https://data-flair.training/blogs/deep-learning-vs-machine-learning/ (2018).

Guan, J., Mao, B., Jiang, X., 1634 (2020). The feature selection based on AndroidManifest.xml. J. Phys: Conf. Ser. https://doi.org/10.1088/1742-6596/1634/1/012027 012027.

Haddadpajouh, H., Dehghantanha, A., Khayami, R., Choo, K.-K.R., 2018. Intelligent os x malware threat detection with code inspection, J. Computer Virology and Hacking Techniques 14 (08 2018). doi:10.1007/s11416-017-0307-5.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition, CoRR abs/1512.03385 (2015). arXiv:1512.03385. URL:http://arxiv.org/abs/1512.03385.

Holzinger, A., Errath, M., Searle, G., Thurnher, B., Slany, W., 2005. From extreme programming and usability engineering to extreme usability in software engineering education (xp+ue/spl rarr/ xu), in: 29th Annual International Computer Software and Applications Conference (COMPSAC'05), Vol. 2, 2005, pp. 169–172 Vol. 1. doi:10.1109/COMPSAC.2005.80.

JSON, ECMA-404 The JSON Data Interchange Standard, Available: URL: https://www.json.org/ (1999).

Jung, J., Park, J., je Cho, S., Han, S., Park, M., Cho, H.-H., 2021. https://jit.ndhu.edu.tw/article/view/2500Feature engineering and evaluation for android malware detection scheme, J. Internet Technology 22 (2) (2021) 423–440. URL:https://jit.ndhu.edu.tw/article/view/2500.

Kapratwar, A., Troia, F.D., Stamp, M., 2017. Static and dynamic analysis of android malware, in: Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ForSE, (ICISSP 2017), INSTICC, SciTePress, pp. 653–662. doi:10.5220/0006256706530662.

Mary Gladence, L., Karthi, M., Anu, M., 2015. A statistical comparison of logistic regression and different bayes classification methods for machine learning, ARPN J. Eng. Appl. Sci. 10 (2015) 5947–5953.

Khandelwal, S., 2019 New attack lets android apps capture loudspeaker data without any permission, The Hacker News, Available: URL:https://thehackernews.com/2019/07/android-side-channel-attacks.html.

Kumar, M., 2020. New android flaw affecting over 1 billion phones let attackers hijack apps, The Hacker News, Available: URL:https://thehackernews.com/2020/05/stranhogg-android-vulnerability.html.

Liang, S., Du, X., 2014. Permission-combination-based scheme for android mobile malware detection, IEEE International Conference on Communications, ICC 2014, Sydney, Australia, June 10–14, 2014 (2014) 2301–2306 doi:10.1109/ICC.2014.6883666. URL:https://doi.org/10.1109/ICC.2014.6883666.

Li, J., Wu, B., Wen, W., 2019. Android malware detection method based on frequent pattern and weighted naive bayes. In: Yun, X., Wen, W., Lang, B., Yan, H., Ding, L., Li, J., Zhou, Y. (Eds.), Cyber Security. Springer Singapore, Singapore, pp. 36–51.

Lin, G., Zhang, J., Luo, W., Pan, L., Xiang, Y., 2017. Poster: Vulnerability discovery with function representation learning from unlabeled projects. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Association for Computing Machinery, New York, NY, USA, pp. 2539–2541. https://doi.org/10.1145/3133956.3138840.

Ma, H., Tian, J., Qiu, K., Lo, D., Gao, D., Wu, D., Jia, C., Baker, T., 2021. Deep-learning-based app sensitive behavior surveillance for android powered cyber-physical systems. IEEE Trans. Industr. Inf. 17 (8), 5840–5850. https://doi.org/10.1109/TII.2020.3038745.

Mahdavifar, S., Abdul Kadir, A.F., Fatemi, R., Alhadidi, D., Ghorbani, A.A., 2020. Dynamic android malware category classification using semi-supervised deep learning (2020) 515–522 doi:10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00094.

Mandal, A., Panarotto, F., Cortesi, A., Ferrara, P., Spoto, F., 2019. Static analysis of android auto infotainment and on-board diagnostics ii apps. Software: Practice and Experience 49, 1131–1161.

Matthews, B., 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme, Biochimica et Biophysica Acta (BBA) -. Protein Struct. 405 (2), 442–451. https://doi.org/10.1016/0005-2795(75)90109-9. URL:http://www.sciencedirect.com/science/article/pii/0005279575901099.

Mukherjee, S., Sharma, N., 2012. Intrusion detection using naive bayes classifier with feature reduction, Procedia Technology 4 (2012) 119–128, 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25–26, 2012. doi: 10.1016/j.protcy.2012.05.017.

Naumenko, V., 2020. Guide on machine learning vs. deep learning vs. artificial intelligence, Trends Engineering, Available: URL:https://jelvix.com/blog/ai-vs-machine-learning-vs-deep-learning (2020).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, Y. Wang, Y. Xiang, A3cm: Automatic capability annotation for android malware, IEEE Access PP (2019) 1–1. doi:10.1109/ACCESS.2019.2946392.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., Xiang, Y., 2020. A survey of android malware detection with deep neural models. ACM Comput. Surv. 53 (6), (Dec. https://doi.org/10.1145/3417978.

Quinlan, J.R., 1986. Induction of decision trees. Mach. Learn 1, 81–106.

Razak, M.F.A., Anuar, N.B., Othman, F., Firdaus, A., Afifi, F., Salleh, R., 2018. Bio-inspired for features optimization and malware detection. Arabian J. Sci. Eng. 43, 6963–6979.

Sahin, D., Kural, O., Akleylek, S., Kilic, E., 2021. A novel permission-based android malware detection system using feature selection based on linear regression. Neural Comput. Appl., 1–16 https://doi.org/10.1007/s00521-021-05875-1.

Sahs, J., Khan, L., 2012. A machine learning approach to android malware detection. In: Proceedings - 2012 European Intelligence and Security Informatics Conference, EISIC 2012, pp. 141–147. https://doi.org/10.1109/EISIC.2012.34.

Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P.G., Álvarez, G., 2013. Puma: Permission usage to detect malware in android. In: Herrero, Á., Snášel, V., Abraham, A., Zelinka, I., Baruque, B., Quintián, H., Calvo, J.L., Sedano, J., Corchado, E. (Eds.), International Joint Conference CISIS'12-ICEUTE'12-SOCO'12 Special Sessions. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 289–298.

Serrà, J., Pascual, S., Karatzoglou, A., 2018. Towards a universal neural network encoder for time series, CoRR abs/1805.03908 (2018). arXiv:1805.03908. URL: http://arxiv.org/abs/1805.03908.

Shang, F., Li, Y., Deng, X., He, D., 2018. Android malware detection method based on naive bayes and permission correlation algorithm. Cluster Computing 21, 1–12. https://doi.org/10.1007/s10586-017-0981-6.

Sharma, A., Dash, S.K., 2014. Mining api calls and permissions for android malware detection, 191–205.

Sharma, H., Kumar, S., 2016. A survey on decision tree algorithms of classification in data mining, International Journal of Science and Research (IJSR).

Shrivastav, N., 2020. Confusion matric, TPR, Available: URL:https://medium.datadriveninvestor.com/confusion-matric-tpr-fpr-fnr-tnr-precision-recall-f1-score-73efa162a25f.

Statcounter, Mobile operating system market share worldwide, GlobalStats, Available: URL:https://gs.statcounter.com/os-market-share/mobile/worldwide (2021).

Verma, S., Muttoo, S., 2016. An android malware detection framework-based on permissions and intents. Defence Science J. 66, 618. https://doi.org/10.14429/dsj.66.10803.

VirusShare, Virusshare team, Dataset, Available: URL:https://virusshare.com/ (2019).

Virustotal, Virustotal antivirus team, Antivirus, Available: URL:https://www.virustotal.com/gui/home/upload (2019).

Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., Wu, K.-P., 2012. Droidmat: Android malware detection through manifest and api calls tracing. In: Proceedings of the 2012 7th Asia Joint Conference on Information Security AsiaJCIS 2012, pp. 62–69. https://doi.org/10.1109/AsiaJCIS.2012.18.

Yerima, S., Muttik, I., Sezer, S., 2015. High accuracy android malware detection using ensemble learning, IET Inform. Security (04 2015). doi:10.1049/iet-ifs.2014.0099.