INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2025, VOL. 71, NO. 4, PP. 1-7
Manuscript received March 03, 2025; revised September 2025. doi: 10.24425/ijet.2025.155487

Comparative features reduction investigation for Android malware detection on Boolean data

Ary Irawan, Piotr Bilski, and Grzegorz Gwardys

***Abstract*—The aim of this research is to enhance the effectiveness of Android malware detection systems by implementing dimensionality reduction techniques on Boolean data. Algorithms such as Linear Discriminant Analysis (LDA), Principal Component Analysis (PCA), and Multi-Correspondence Analysis (MCA) serve as operations preceding the classification stage. The analysis is carried out using multiple classifiers such as Random Forest Classifier, Logistic Regression, and Support Vector Machines to measure how effective they can detect cyber threats. Results show that the Decision Tree Classifier, implemented without dimensionality reduction, achieved the optimal results with 100% accuracy. Efficient feature selection and rapid computation in the context of malware detection are necessary for real-time mobile cyber environment applications.**

***Keywords*—android; malware; machine learning; algorithms; dimensionality reduction; classification**

I. Introduction

MOBILE devices expand quickly and currently share majority of computers used worldwide. The Android Operating System (OS) is the most prominent software solution in this field. Its popularity makes it an easy target for the malware (similarly to Microsoft Windows in the desktop market). Therefore, it is important to analyze existing threats and find possible countermeasures against them. The Open-Source environment and customizable features of Android provide options for investigating system-level capabilities, app development, and user interaction. Its accessibility and analytical power are greater than in the more restricted environment of the main counterpart, i.e. Apple iOS. The Android ecosystem covers a wide range of devices from different manufacturers, featuring diverse hardware, specifications and software settings. Its development environment (including Android Studio and support for numerous programming languages) is accessible for researchers. Android also provides a higher level of adaptability for personalization and exploration in contrast to iOS. It can be used to carry out experiments, integrate original functionalities, or create software prototypes.

Malware detection (especially exploits, rootkits or spyware) requires usage of reverse engineering. Both static and dynamic analyses provide information extracted from apps' structure and behavior [1]. Static analysis entails inspecting applications' code, resources, and manifest files without executing them. It covers features like requested permissions, API calls, code structure, and resource references [2]. Decompiling bytecode may provide further information. Dynamic analysis involves executing the program in a controlled setting and monitoring its performance. Features are obtained from interactions with the OS, hardware, network, and other programs. Methods such as hooking and runtime modification may be used to collect extra characteristics [3].

Mobile OS applications do not make direct system calls. Instead, they interact with the system through a Virtual Machine (VM) or a managed runtime environment. Because the VM handles the translation of API to system calls, the direct intentions of the program may be obscured. The VM acts as a buffer that masks the application's true behavior from the underlying OS. This makes it challenging to analyze the behavior of the application purely based on system calls, as the actual interactions are abstracted away. Machine learning models can be trained to recognize patterns of normal application behavior. Deviations from the nominal operation can signal potential malicious activity.

Based on the information collected from the analysis, decisions about the nature of the program can be made. The amount of extracted information is usually large and difficult to interpret manually. Therefore Artificial Intelligence (AI) is the standard solution, being able to solve binary classification problems [4]. Feature reduction may be essential here, as it minimizes the amount of data processed both during the classifier training and decision making. It may lead to both increasing accuracy and suppressing time of computations. This is often achieved by filtering attributes with low information capacity or detecting highly correlated features. The goal is to simplify data sets.

The aim of this paper is to investigate efficiency of the data reduction of Boolean-type data for the mobile malware detection. Well-known feature reduction and classification algorithms were combined to find the best solution the task. The framework was deployed using Python language and tested on selected, publicly available data sets.

The content of the paper is as follows. In Section 2 the research gap in the field malware detection is presented. Section 3 presents the general framework applied to the task. Selected feature reduction and classification algorithms are briefly described. In Section 4 the experimental setup and results are discussed. Section 5 contains conclusions and future prospects.

Authors are with Warsaw University of Technology, Poland (01186007@pw.edu.pl, piotr.bilski@pw.edu.pl, grzegorz.gwardys@pw.edu.pl).

II. LITERATURE REVIEW

The problem of Android malware detection may be defined through searching for the minimal set of features allowing for detecting dangerous programs with the highest possible accuracy. Multiple AI methods have been employed so far, including Support Vector Machine (SVM), DT (Decision Tree), RFC (Random Forest Classifier), MLP (Multilayer Perceptron, or Naïve Bayes Classifier (NBC) with the level of accuracy, sensitivity, and precision between 70% and 99% [5-10]. They are in most cases used on full data sets (with all features involved), though dimensionality reduction is also employed.

Numerous features possible to extract, but determining which ones are significant is a challenging task. Therefore, dimension reduction techniques may be used. They include genetic algorithms, Weighted Features Ranking Dimension Reduction, hybrid-based feature selection, correlation-based or community-based feature selection [6-10]. The aim of the operation is to maintain the accuracy while simplifying the data representation and suppress time of computations (especially during training) [5]. The latter means lower costs, especially for cloud services or other pay-per-compute platforms. It may also lead to improved model performance and generalization.

Datasets used for experiments contain different numbers of benign and malware applications with meta-information such as permission-based, control flow, component, and system calls features [5-10]. The total number of programs downloaded from Aptoide is 12,360. Each application was processed with the VirusTotal API -- a tool incorporating a total of 56 antivirus engines to scan for any malicious content. This process confirmed the presence and absence of malware in the applications, thus creating a labeled dataset with two classes: malicious and benign (safe). The dataset contains 8058 cases of 'benignware' and 3418 malware with 167 features. They are classified as continuous, discrete, and Boolean (two-valued only). Continuous types include numerically-valued features such as ratings given by users or metrics touching screen size and CPU type. They facilitate the in-depth analysis of applications. Discrete meta-information is the name of the developer or the lowest SDK version needed which fits into the categorical data. Boolean data include binary variables, most commonly a yes/no or true/false. For example, an application may or may not require certain permission to be used.

Among the most popular dimensionality reduction techniques, PCA has gained attention, since it can handle high-dimensional feature sets, common in Android malware analysis. In [11], its performance varied across different algorithms. For example, the F-measure for PCA combined with K-Nearest Neighbors (KNN) was 0.945, while for NBC it dropped to 0.793, which indicates that probably PCA is not fit for all methods. On the other hand, using LDA with KNN, NBC, Sequential Minimal Optimization (SMO), MLP, RFC, C.45, and LR allowed for improved classification compared to the full feature set. For instance, F-measure for LDA-KNN combination was 0.925, higher than results obtained with PCA [10]. LDA provided a significant gain in speed, with over 1400 times runtime reduction of training and testing compared to all features.

MCA is an extension of the Correspondence Analysis (CA) for the identification of relationships between datasets comprising more than two discrete variables. In the case of the UCI datasets presented in [12], performance metrics have increased in terms of accuracy across different datasets when CA is applied. Results confirm that it boosts classification accuracy on multiple datasets. Selected experiments showed improvement when CA was used. This implies it does sustain a significant amount of information for boosting classification performance while complexity is retained at a lower level. Accuracy varied between 40% to 100% depending on the applied classifier, i.e. Artificial Neural Networks (ANN), C4.5, and KNN algorithms.

Small datasets run the risk of overfitting, where the model learns from insignificant details specific to the particular training examples rather than general patterns, leading to poor generalization on new data [13, 14]. Modern deep learning models, such as Deep Neural Networks (DNN), require large quantities of data. Using small datasets with their conventional training methods often means degraded performances compared to traditional techniques, like MLP and SVM [15, 16, 17]. As DNN require large data sets and significant computational resources, they were excluded from the presented research, though should be considered in the future.

In [18], a new system for detecting malware, based on a DNN was developed. It employs the following combination of features: permissions, intent filters, invalid certificates, presence of APK files in the asset folder, and API calls with 1200 android applications, which included 600 benign apps and 600 malicious apps. The DNN proved to be very effective with an accuracy of 95.31% for classifying benign and malicious applications. These results stemmed from thorough testing with particular sets of features.

The dataset [19] employed in this research was tackled with both static and dynamic analysis. Boolean data cover features that are either present or absent [20]. Reducing dimensionality and improving classification performance has been achieved so far via feature selection and PCA. In this study, scenarios with No Feature Reduction (NF), Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Multiple Correspondence Analysis (MCA) were tested. Their impact on accuracy, sensitivity, specificity, and computing time was evaluated. The general scheme of the approach is in Fig. 1.

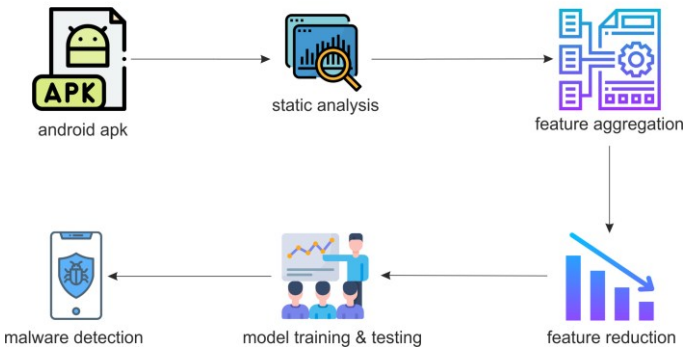

Fig. 1. Scheme of the malware detection using static analysis

In Android malware detection, there is a lack of comprehensive studies that systematically investigate

effectiveness of these techniques for Boolean data, representing existence or lack of the particular detail in the application's profile. Such data play an important role in static analysis, as they represent the raw form of the program, allowing for deep structure and behavior investigation without the code execution. While profiling a mobile application, this method allows for basic checks on the usage of components, like whether broadcast receivers, shared preferences, or background services, were used. For instance, a broadcast receiver existence is denoted by 1, and its absence is marked by 0.

Boolean data allow for easier apps partitioning and detecting anomalies, such as abnormal component and permission levels, and disallowable modifications. Each aspect of the code, including libraries, functions, and data structures is simple to process. Static analysis tools need Boolean data to look for patterns or signatures of known vulnerabilities or malware for effective threat detection. In reverse engineering they help analysts to understand program logic, identify vulnerabilities, and assess security risks. This is especially important in uncovering zero-day or backdoor exploits [21].

III. Malware Detection Algorithms

Machine learning methods are used to analyze program activity via the operating systems' Application Programming Interface (API). This involves integrating Real-Time data and merging third-party services into mobile apps [22-23]. Investigating features based on these call reveals distinctive operational traits of mobile devices, which may help to identify suspicious behavior patterns. The technique entails storing them to detect suspicious activities in new applications, facilitating development of detection algorithms. Security specialists employ command signatures, which are identifiable indicators of hazardous instructions, to analyze and develop detection methods for malicious activities in academic conversations.

A. System Implementation

The system implemented for the research (Fig.2) is divided into two modules. The first one employs feature reduction to minimize the amount of extracted from the application. It uses MCA, PCA, and LDA [21-23]. These are successful in examining relations between categorical variables, identifying variability and trends in data, optimizing class distinction, respectively [24-26]. However, their usefulness is known only in conjunction with the subsequent classifier. Therefore, the second module is responsible for the malware detection, segregation of the analyzed application into one of two categories: malware or "benign", harmless program. This is therefore the binary classification problem. The following section covers description of all methods used in the research.

B. Classification Algorithms

Eight machine learning methods with the rich history of practical implementations were employed in this study: Random Forest Classifier (RFC), Logistic Regression (LR), Gaussian Naïve Bayes (GNB), KNN, SVM, Linear Classifier with Stochastic Gradient Descent (LCSGD), Decision Tree Classifier (DTC), and MLP. They solve the binary classification task, distinguishing between the legitimate (positive category – 0) and dangerous (negative category – 1) software:

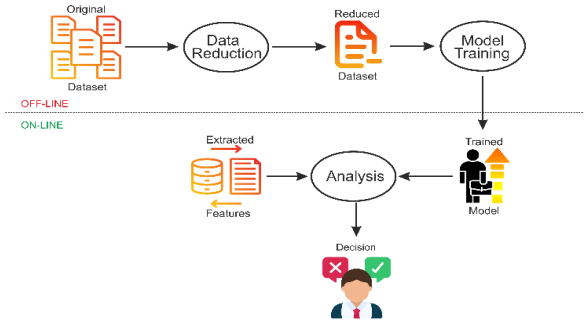

Fig. 2. Malware detection scheme used in the presented research.

Both DTC and RFC are rule-based approaches, knowledge extracted by them during training is readable by the human operator. The tree-like structure allows for the top-down feature vector processing (using tests existing in nodes) until the category in the leaf is reached. Because the single tree has the tendency to excessively adjust the structure to the data, random forests are used as the more generic solution, with the greater generalization capabilities. The feature reduction is performed as the by-product of the decision generation (only part of the features is selected to nodes). However, it is possible to enforce the minimum set of attributes externally.

Statistical/numerical approaches are represented by LR, MLP, SVM and LCSGD. Their aim is to process the input data through the specifically designed function.

LR is a statistical method used to represent the relationship between a dependent variable and one or more independent variables. It involves fitting a straight line to the data in order to make predictions. GNB is a probabilistic classifier that relies on Bayes' theorem KNN predicts the class or value of a data point by considering the majority class or average value of its k nearest neighbors in the feature space.

SVM implements the separating hyperplane in a multidimensional space to divide different categories. The goal is to maximize the distance between them, which is known as the margin. SGD is a linear classifier trained by updating the model parameters using the gradient of the loss function, which is computed on tiny batches of training data. MLP is a specific kind of ANN that consists of numerous layers of interconnected nodes, such as input, hidden, and output layers. All classifiers were used already optimized through grid search [30].

Every classifier uses knowledge of different type, potentially capturing distinct patterns and correlations in data. By comparing performance of these models on the same dataset, it is possible to select the best one. In the end, utilizing several classifiers allows us to construct prediction models for Boolean datasets that are more precise, resilient, and adaptable.

C. Feature Reduction Approaches

1) **Principal Component Analysis** is used to identify the main patterns in a data matrix. It does this by generating a set of score and loading plots that capture the dominant patterns in the matrix [31].
2) **Linear Discriminant Analysis** aims to optimize the ratio of between-class variance to within-class variance in a given dataset, ensuring the highest level of separability. It is commonly used for classification in speech recognition. The fact that LDA can directly handle numerous categories and generate discriminant functions for each

category in comparison to a baseline makes it especially helpful for Boolean data [32]. The algorithm is well-known for its resilience against fluctuations in the data. It focuses on maximizing the separation between classes based on their overall distribution in data, rather than being influenced by individual data points [33, 34].

3) **Multiple Correspondence Analysis** is comparable to PCA for categorical data, while PCA is defined geometrically rather than statistically. It is used to determine original features and redefine rows and columns of a data set as points in a high-dimensional Euclidean space. This allows to discover the primary dimensions that captures the maximum amount of variation [35].

The experiments were performed on a machine with 64 GB RAM and AMD Ryzen™ 3 3200U Processor (2.6 GHz base clock, 3.5 GHz max boost clock, 4 MB L3 cache, 2 cores). On the software side, Python language with scikit learn library were used. Linux Ubuntu version 20.04.6 was running as the operating system.

D. Classifiers' Configuration

Hyperparameters of the applied classifiers were fine-tuned to maximize their accuracy. They are presented in Tab. I and Tab. II, where n_{est} is the number of estimators (classifiers), d_{max} is the maximum tree depth, C is the regularization coefficient, i_{max} is the maximum number of iterations, *smooth* is the decision function smoothing, n_n is the number of nearest neighbors, γ is the width of the Gaussian kernel, *hls* is the number of neurons in the hidden layer, *act* is he activation function inside neurons and *solver* is the optimization task solver.

TABLE I
CONFIGURATION OF CLASSIFIERS USED IN THE EXPERIMENTS

	RFC		LR		GNB	kNN	SVM	
Reduct	n_{est}	d_{max}	C	i_{max}	*smooth*	n_n	C	γ
NF	200	None	1	800	1	3	1	1
PCA	100	10	10	800	10	3	1	1
LDA	100	10	0,1	100	0,1	7	100	1
MCA	300	20	0,01	100	0,1	3	0,1	0,1

TABLE II
CONFIGURATION OF CLASSIFIERS USED IN THE EXPERIMENTS

Reduct	DTC	MLP			
	d_{max}	*hls*	i_{max}	*act*	*solver*
NF	20	125	800	relu	adam
PCA	10	100	800	relu	adam
LDA	50	125	200	tanh	sgd
MCA	20	5	400	tanh	sgd

E. Used Dataset

A dataset from [36] was used in the experiments, covering 12.360. It includes both binary and non-binary data. The former are variables with only two possible values (such as 0 and 1), can simplify the decision-making process for binary classifiers [37]. The features obtained from static analysis were as follows: API calls, Inter-Component Communication (ICC), and Android manifest.

The Android uses ICC or Binder for inter-process communication. Through the use of ICC, an application component has the ability to access data from another component within the same or different application, or a remote service [38]. An instance of a product delivery software may utilize a Map API to determine the geographical coordinates of a device [39].

The Android Manifest file is an archive of data regarding an application's various components, permissions, and configuration. It is frequently monitored to detect possible signs of malicious conduct. The manifest files in Android applications comprise of two sources of information, uses-permissions and uses-features [40]. They are widely acknowledged as the security measure. In order to install any application, it is necessary for the user to provide access permissions [41].

Identifying Android malware based just on the Manifest file is difficult because of advanced evasion techniques and dynamic changes in the malware behavior. Thus, a comprehensive approach that combines Manifest analysis with other detection methods and security mechanisms is crucial for successful malware detection.

IV. MALWARE DETECTION PERFORMANCE

All models were trained and tested on the original and reduced data after removing the numerical (continuous type) features of the dataset. The test scenario was implemented using cross validation with 5 folds [42]. Several measures evaluate the efficiency of the model.

The optimal results obtained for the particular reductions methods are in Fig. 3. In Tab. III-VI, the performance of each model may be analyzed and evaluated considering accuracy, sensitivity, and specificity. They prove that applied classifiers on binary data perform satisfactorily, though their outcomes are worse when data reduction is applied.

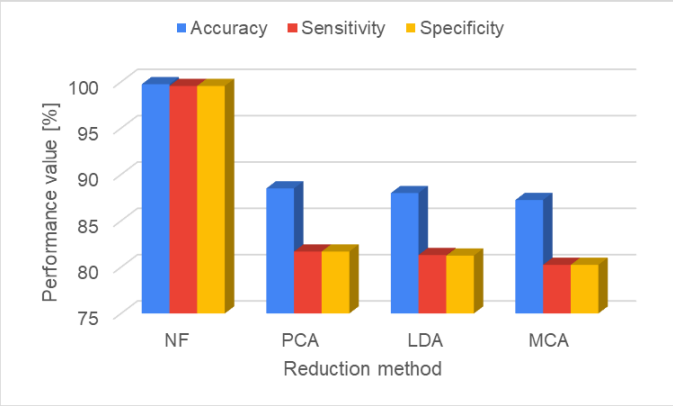

Fig. 3. Dimensionality reduction performance

Among all used methods, the DT model is the best, reaching an accuracy of 100%, with sensitivity and specificity of 100% and low computation time of 180 milliseconds-only, without dimensionality reduction (the NFR case). DT naturally handles high-dimensional data by splitting based on thresholds in features. In the NFR scenario the model preserved all features originally included, thus retaining some of the most critical information which could have otherwise been lost.

In all reduction cases the loss in accuracy is visible. It is acceptable, provided that the minimized set of features is much easier or faster to collect than the original (larger) set. Each method provides a different range of features (their number does not exceed 40), which means the significant speed up in the real-time analysis may be obtained. Also, Boolean data are simpler to describe the vector attack (either the operation or event took place or not), supporting the idea of focusing on such information extracted from the operating system. Besides, RFC and DT operating on the original data may be used as the feature selectors (though the threat of overfitting exists here) with much smaller computational effort.

TABLE III
PERFORMANCE OF CLASSIFIERS FOR THE ORIGINAL DATA SET

	Algorithms (%)							
Measure	RFC	LR	GNB	KNN	SVM	LCSGD	DTC	MLP
acc	100	100	100	99	99	100	100	100
sens	100	100	100	96	97	100	100	100
spec	100	100	100	100	100	100	100	100

TABLE IV
PERFORMANCE OF CLASSIFIERS FOR THE PCA REDUCTION

	Algorithms (%)							
Measure	RFC	LR	GNB	KNN	SVM	LCSGD	DTC	MLP
acc	92	90	90	72	92	90	91	91
sens	78	71	71	5	78	69	78	78
spec	97	97	97	100	97	98	96	97

TABLE V
PERFORMANCE OF CLASSIFIERS FOR THE LDA REDUCTION

	Algorithms (%)							
Measure	RFC	LR	GNB	KNN	SVM	LCSGD	DTC	MLP
acc	91	90	90	72	90	90	91	90
sens	77	71	71	7	71	71	78	73
spec	97	98	98	99	98	98	97	97

TABLE VI
PERFORMANCE OF CLASSIFIERS FOR THE MCA REDUCTION

	Algorithms (%)							
Measure	RFC	LR	GNB	KNN	SVM	LCSGD	DTC	MLP
acc	91	89	89	71	89	89	91	89
sens	78	67	67	6	67	67	77	67
spec	97	99	99	99	99	99	97	99

CONCLUSIONS

This study, demonstrated the importance of dimensionality reduction in combination with machine learning in the improvement of malware detection in Android OS. Application of the presented methods enabled to show their impact on improving classification performance. The DTC was the best performing model with 100% accuracy, sensitivity, specificity, and precision without dimensionality reduction. This means that the data was well-structured and the model captured the underlying patterns of the dataset quite well.

The results have underlined the importance of Boolean data in static analysis for malware detection, as this approach gives knowledge of application behavior without its execution. Also, computational efficiency of these techniques.

Future work must focus on validation of the obtained results on additional data (especially using different datasets, though it should be checked if they contain identical features). Also, comparative analysis between the applied approaches and the explainable AI (including decision trees and random forests) should be performed to verify which approach is more robust. Finally, the ability to implement these techniques into the real-time detection systems should be confirmed.

REFERENCES

[1] P. Benedusi, “Improving reverse engineering models with test-case related knowledge,” *Inf. Softw. Technol.*, vol. 38, no. 11, pp. 711–718, Nov. 1996, https://doi.org/10.1016/0950-5849(96)01119-6

[2] L. Li *et al.*, “Static analysis of android apps: A systematic literature review,” *Inf. Softw. Technol.*, vol. 88, pp. 67–95, Aug. 2017, https://doi.org/10.1016/j.infsof.2017.04.001

[3] H. Binder, K. Krohn, and S. Preibisch, “‘Hook’-calibration of GeneChip-microarrays: Chip characteristics and expression measures,” *Algorithms Mol. Biol.*, vol. 3, no. 1, p. 11, Dec. 2008. https://doi.org/10.1186/1748-7188-3-11

[4] N. Mohapatra, B. Satapathy, B. Mohapatra, and B. K. Mohanta, “Malware Detection using Artificial Intelligence,” in *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Kharagpur, India: IEEE, Oct. 2022, pp. 1–6. https://doi.org/10.1109/ICCCNT54827.2022.9984218

[5] J. Sahs and L. Khan, “A Machine Learning Approach to Android Malware Detection,” in *2012 European Intelligence and Security Informatics Conference*, Odense, Denmark: IEEE, Aug. 2012, pp. 141–147. https://doi.org/10.1109/EISIC.2012.34

[6] A. Fatima, R. Maurya, M. K. Dutta, R. Burget, and J. Masek, “Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning,” in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, Budapest, Hungary: IEEE, Jul. 2019, pp. 220–223. https://doi.org/10.1109/TSP.2019.8769039

[7] M. Z. Mas’ud, S. Sahib, M. F. Abdollah, S. R. Selamat, and R. Yusof, “Analysis of Features Selection and Machine Learning Classifier in Android Malware Detection,” in *2014 International Conference on Information Science & Applications (ICISA)*, Seoul, South Korea: IEEE, May 2014, pp. 1–5. https://doi.org/10.1109/ICISA.2014.6847364

[8] S. K. Smmarwar, G. P. Gupta, and S. Kumar, “A Hybrid Feature Selection Approach-Based Android Malware Detection Framework Using Machine Learning Techniques,” in *Cyber Security, Privacy and Networking*, vol. 370, D. P. Agrawal, N. Nedjah, B. B. Gupta, and G. Martinez Perez, Eds., in Lecture Notes in Networks and Systems, vol. 370. , Singapore: Springer Nature Singapore, 2022, pp. 347–356. https://doi.org/10.1007/978-981-16-8664-1_30

[9] K. Deepa, G. Radhamani, and P. Vinod, “Investigation of Feature Selection Methods for Android Malware Analysis,” *Procedia Comput. Sci.*, vol. 46, pp. 841–848, 2015. https://doi.org/10.1016/j.procs.2015.02.153

[10] A. Bhattacharya and R. T. Goswami, “Community Based Feature Selection Method for Detection of Android Malware:,” *J. Glob. Inf. Manag.*, vol. 26, no. 3, pp. 54–77, Jul. 2018. https://doi.org/10.4018/JGIM.2018070105

[11] D. Ö. Şahin, O. E. Kural, S. Akleylek, and E. Kılıç, “Permission-based Android malware analysis by using dimension reduction with PCA and LDA,” *J. Inf. Secur. Appl.*, vol. 63, p. 102995, Dec. 2021. https://doi.org/10.1016/j.jisa.2021.102995

[12] T. R. Payne and P. Edwards, “Dimensionality Reduction through Correspondence Analysis,” Oct. 14, 1999. Accessed: Dec. 06, 2024. [Online]. Available: https://eprints.soton.ac.uk/263091/1/camap.pdf

[13] W. Zhao, “Research on the deep learning of the small sample data based on transfer learning,” presented at the GREEN ENERGY AND SUSTAINABLE DEVELOPMENT I: Proceedings of the International Conference on Green Energy and Sustainable Development (GESD 2017), Chongqing City, China, 2017, p. 020018. https://doi.org/10.1063/1.4992835

[14] L. Brigato and L. Iocchi, "A Close Look at Deep Learning with Small Data," Oct. 25, 2020, *arXiv*: arXiv:2003.12843. https://doi.org/10.48550/arXiv.2003.12843

[15] S. Feng, H. Zhou, and H. Dong, "Using deep neural network with small dataset to predict material defects," *Mater. Des.*, vol. 162, pp. 300–310, Jan. 2019. https://doi.org/10.1016/j.matdes.2018.11.060

[16] J. Jiang, R. Wang, M. Wang, K. Gao, D. D. Nguyen, and G.-W. Wei, "Boosting Tree-Assisted Multitask Deep Learning for Small Scientific Datasets," *J. Chem. Inf. Model.*, vol. 60, no. 3, pp. 1235–1244, Mar. 2020. https://doi.org/10.1021/acs.jcim.9b01184

[17] B. Labbé, R. Hérault, and C. Chatelain, "Learning Deep Neural Networks for High Dimensional Output Problems," in *2009 International Conference on Machine Learning and Applications*, Miami, FL, USA: IEEE, Dec. 2009, pp. 63–68. https://doi.org/10.1109/ICMLA.2009.48

[18] A. Naway and Y. LI, "Using Deep Neural Network for Android Malware Detection," 2019. https://doi.org/10.48550/ARXIV.1904.00736

[19] A. Martín, "ADROIT." Mendeley, Nov. 15, 2017. https://doi.org/10.17632/YR92XBRVGX.2

[20] D. P. Farrington and R. Loeber, "Some benefits of dichotomization in psychiatric and criminological research," *Crim. Behav. Ment. Health*, vol. 10, no. 2, pp. 100–122, Jun. 2000. https://doi.org/10.1002/cbm.349

[21] C. S. Calhoun, J. Reinhart, G. A. Alarcon, and A. Capiola, "Establishing Trust in Binary Analysis in Software Development and Applications," in *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*, Rome, Italy: IEEE, Sep. 2020, pp. 1–4. https://doi.org/10.1109/ICHMS49158.2020.9209473

[22] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, "Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures," in AusDM '11: Proceedings of the Ninth Australasian Data Mining Conference, vol 121, pp. 171-182, December 2011. https://dl.acm.org/doi/10.5555/2483628.2483648

[23] S. Choi, H. Park, H. Lim, and T. Han, "A static API birthmark for Windows binary executables," *J. Syst. Softw.*, vol. 82, no. 5, pp. 862–873, May 2009. https://doi.org/10.1016/j.jss.2008.11.848

[24] B. A. Draper, K. Baek, M. S. Bartlett, and J. R. Beveridge, "Recognizing faces with PCA and ICA," *Comput. Vis. Image Underst.*, vol. 91, no. 1–2, pp. 115–137, Jul. 2003. https://doi.org/10.1016/S1077-3142(03)00077-8

[25] P. Xanthopoulos, P. M. Pardalos, and T. B. Trafalis, "Linear Discriminant Analysis," in *Robust Data Mining*, in SpringerBriefs in Optimization. , New York, NY: Springer New York, 2013, pp. 27–33. https://doi.org/10.1007/978-1-4419-9878-1_4

[26] D. Ayele, T. Zewotir, and H. Mwambi, "Multiple correspondence analysis as a tool for analysis of large health surveys in African settings," *Afr. Health Sci.*, vol. 14, no. 4, p. 1036, Jan. 2015, https://doi.org/10.4314/ahs.v14i4.35

[27] J. Lever, M. Krzywinski, and N. Altman, "Principal component analysis," *Nat. Methods*, vol. 14, no. 7, pp. 641–642, Jul. 2017, https://doi.org/10.1038/nmeth.4346

[28] G. Feng, D. Hu, M. Li, and Z. Zhou, "A Novel LDA Approach for High-Dimensional Data," in *Advances in Natural Computation*, vol. 3610, L. Wang, K. Chen, and Y. S. Ong, Eds., in Lecture Notes in Computer Science, vol. 3610. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 209–212. https://doi.org/10.1007/11539087_23

[29] B. Broeksema, A. C. Telea, and T. Baudel, "Visual Analysis of Multi-Dimensional Categorical Data Sets," *Comput. Graph. Forum*, vol. 32, no. 8, pp. 158–169, Dec. 2013. https://doi.org/10.1111/cgf.12194

[30] G. S and S. Brindha, "Hyperparameters Optimization using Gridsearch Cross Validation Method for machine learning models in Predicting Diabetes Mellitus Risk," in *2022 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, Chennai, India: IEEE, Mar. 2022, pp. 1–4. https://doi.org/10.1109/IC3IOT53935.2022.9768005

[31] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemom. Intell. Lab. Syst.*, vol. 2, no. 1–3, pp. 37–52, Aug. 1987, doi: https://doi.org/10.1016/0169-7439(87)80084-9

[32] S. T. Mueller, "Psychology and Human Factors in R II," Advanced Statistical Analysis & Design II. Accessed: Jan. 17, 2024. [Online]. Available: https://pages.mtu.edu/~shanem/psy5220/index.html

[33] C. Vidaurre, M. Kawanabe, P. Von Bünau, B. Blankertz, and K. R. Müller, "Toward Unsupervised Adaptation of LDA for Brain–Computer Interfaces," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 3, pp. 587–597, Mar. 2011. https://doi.org/10.1109/TBME.2010.2093133

[34] F. Tang and H. Tao, "Fast linear discriminant analysis using binary bases," *Pattern Recognit. Lett.*, vol. 28, no. 16, pp. 2209–2218, Dec. 2007. https://doi.org/10.1016/j.patrec.2007.07.007

[35] M. Greenacre, "From Correspondence Analysis to Multiple and Joint Correspondence Analysis," *SSRN Electron. J.*, 2005. https://doi.org/10.2139/ssrn.847664

[36] A. Mahindru, "Android permissions dataset, Android Malware and benign Application Data set (consist of permissions and API calls)." Mendeley, Mar. 04, 2020. https://doi.org/10.17632/B4MXG7YDB7.3

[37] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognit.*, vol. 44, no. 8, pp. 1761–1776, Aug. 2011. https://doi.org/10.1016/j.patcog.2011.01.017

[38] J. Jenkins and H. Cai, "Dissecting Android Inter-component Communications via Interactive Visual Explorations," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Shanghai: IEEE, Sep. 2017, pp. 519–523. https://doi.org/10.1109/ICSME.2017.74

[39] V. Sihag, A. Swami, M. Vardhan, and P. Singh, "Signature Based Malicious Behavior Detection in Android," in *Computing Science, Communication and Security*, vol. 1235, N. Chaubey, S. Parikh, and K. Amin, Eds., in Communications in Computer and Information Science, vol. 1235. , Singapore: Springer Singapore, 2020, pp. 251–262. https://doi.org/10.1007/978-981-15-6648-6_20

[40] F. Shen, "Android Security via Static Program Analysis," in *Proceedings of the 2017 Workshop on MobiSys 2017 Ph.D. Forum*, Niagara Falls New York USA: ACM, Jun. 2017, pp. 19–20. https://doi.org/10.1145/3086467.3086469

[41] B. Sanz *et al.*, "MAMA: MANIFEST ANALYSIS FOR MALWARE DETECTION IN ANDROID," *Cybern. Syst.*, vol. 44, no. 6–7, pp. 469–488, Oct. 2013. https://doi.org/10.1080/01969722.2013.803889

[42] H. A. Martens and P. Dardenne, "Validation and verification of regression in small data sets," *Chemom. Intell. Lab. Syst.*, vol. 44, no. 1–2, pp. 99–121, Dec. 1998. https://doi.org/10.1016/S0169-7439(98)00167-1

[43] G. Yan, N. Brown, and D. Kong, "Exploring Discriminatory Features for Automated Malware Classification," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 7967, K. Rieck, P. Stewin, and J.-P. Seifert, Eds., in Lecture Notes in Computer Science, vol. 7967, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 41–61. https://doi.org/10.1007/978-3-642-39235-1_3

[44] J.-S. Hong and G.-S. Hwang, "Interpretability Comparison of Popular Decision Tree Algorithms," *J. Soc. Korea Ind. Syst. Eng.*, vol. 44, no. 2, pp. 15–23, Jun. 2021. https://doi.org/10.11627/jkise.2021.44.2.015

[45] M. Moshkov, "On the depth of decision trees over infinite 1-homogeneous binary information systems," *Array*, vol. 10, p. 100060, Jul. 2021, https://doi.org/10.1016/j.array.2021.100060