Journal of Emerging Computer Technologies
Research Article
Received: *2024-08-06* | ***Reviewing:*** *2024-08-07* & *2024-09-03*| ***Accepted:*** *2024-09-04* | ***Online:*** *2024-09-04* | ***Issue Date:*** *2024-12-31*
Doi: 10.57020/ject.1528965

Android Ransomware Detection System using Feature Selection with Bootstrap Aggregating MARS

Kerem GENCER
Afyon Kocatepe University
Faculty Of Engineering, Computer Engineering
Afyonkarahisar, Türkiye
kgencer@aku.edu.tr
0000-0002-2914-1056
(Corresponding Author)

Fatih BAŞÇİFTÇİ
Selcuk University
Faculty Of Technology, Computer Engineering
Konya, Türkiye
basciftci@selcuk.edu.tr
0000-0003-1679-7416

***Abstract*— Android ransomware has become one of the most dangerous types of attack that have occurred recently due to the increasing use of the Android operating system. Generally, ransomware is based on the idea of encrypting the files in the victim's device and then demanding money to provide the decryption password. Machine learning techniques are increasingly used for Android ransomware detection and analysis. In this study, Android ransomware is detected using Bootstrap Aggregating based Multivariate Adaptive Regression Splines (Bagging MARS) for the first time in feature selection. A feature matrix with 134 permissions and API calls in total was reduced to 34 features via the proposed Bagging MARS feature selection technique. Multi-Layer Perceptron (MLP), one of the classification techniques, produced the best accuracy with 90.268%. Additionally, the proposed feature selection method yielded more successful results compared to the filter, wrapper, and embedded methods used. Thus, this method, which was used for the first time to detect the common features of Android Ransomware, will enable the next Android Ransomware detection systems to work faster and with a higher success rate.**

Keywords*— Bagging, feature selection, machine learning, MARS, ransomware, static analysis

I. INTRODUCTION

Ransomware may infect a victim's computer when the victim opens an e-mail attachment sent to him/her, visits a website, downloads an unsecure file, or installs unapproved software. After infecting the victim's computer, the malicious software begins the first encryption phase by deleting the unencrypted original files, thereby preventing the victim from access ing their files. The attacker does not allow access to the original files until he/she is paid a ransom via a payment method such as virtual money.

If the ransom is not paid in the time specified by the attacker, he/she destroys the encryption key and permanently deletes the data [1]. In the beginning, the majority of ransomware victims were Windows desktop users. After a while, ransomware emerged on different platforms including Android, iOS and other mobile operating systems. In recent years, types of attack shifted towards ransomware of a particular type [2]. Ransomware attacks continued at the beginning of 2020 with a range of rapidly spreading ransomware including Maze, Sodinokibi, DoppelPaymer, Nemty, Nefilim, CLOP and Sekhmet. According to the cybersecurity firm Emsisoft, attackers published the stolen data on their own websites in case a payment was not made.

Emsisoft announced that the aim of ransomware groups is, in general, selling the stolen data to rivals, using the stolen data to attack business partners of the victim and publishing the victim's private information on their webpages for everyone to see.Some attackers have exploited COVID-19 with reventive measures to prevent future incidents [3].

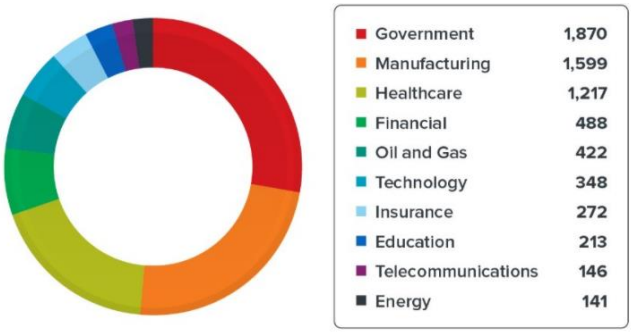

Figure 1. Industries Affected by Ransomware

As seen in Figure 1, ransomware victims range across a broad spectrum including industries such as production, health and education to municipal management [4]. Mobile operating systems used around the globe are 74.25% Android, 25.15% iOS, 0.23% Samsung and 0.37% other operating systems [5]. According to Chebyshev (2020) [6], there were 60.176 mobile ransomware attacks in 2018 worldwide, while this number rose to 68.362 in 2019. Victims of the largest 11 ransomware attacks in 2020 have spent 144,2 million dollars up to now on various costs ranging from investigating the attack, reinstating networks and restoring backups, paying computer hackers and applying according to these numbers, there is a 13.6% increase in mobile ransomware attacks, annually. The distribution of recent ransomware attacks across countries is given in Figure 2 [7]. Ransomware hides in uncertified source downloads, Google Play applications and exploitation kits using security vulnerabilities that are yet unpublicized, and spread from there. There are two ransomware types, crypto-ransomware and screen locker ransomware. Crypto-ransomware encrypts the user's data and files. The key used in the encryption process is required to decrypt them.

Screen locker ransomware is activated by the user downloading a fake application from Google Play store or a

Cite (APA): Gencer, K., Başçiftçi, F. (2024). Android Ransomware Detection System using Feature Selection with Bootstrap Aggregating MARS. *Journal of Emerging Computer Technologies*, 4(1), 38-45. Doi: 10.57020/ject.1528965

third-party market, and the mobile device then being rendered inoperable by locking the user screen. In general, a ransomware attack has three main stages: Earning the execution privilege, preventing access, and finally notifying the victim with the ransom message [8].

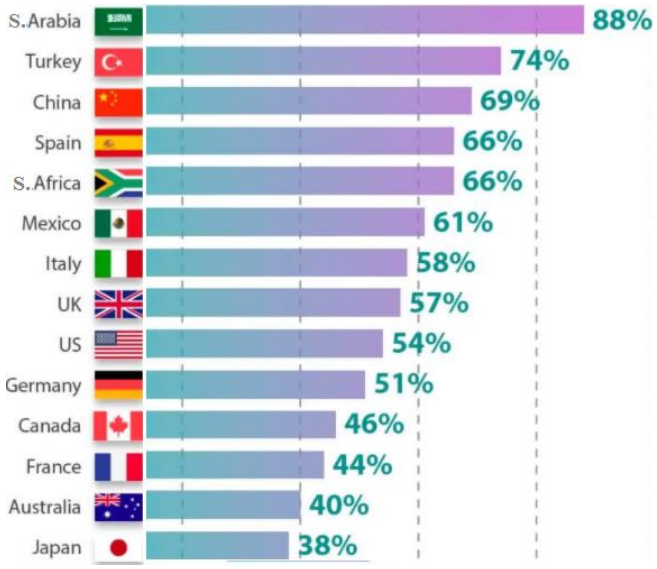

Figure 2. Distribution of Ransomware Attacks across Countries

Ransomware is a one of the type of malware. However, it has its own characteristics. Since they encrypt and store the files on the infected system, cleaning them from the system does not save the victim's information. Ransomware's work differently than other malware requires the use of different operating features specific to ransomware in their detection. Some of these features are given below. In recent years, ransomware attackers have found many methods to avoid detection by antivirus software and security utilities. However, ransomware does have some common properties that can allow the detection tools to reveal them. In the Android environment, ransomware has been found to have some common behavior and features, as follows [9-10].

Obtaining administrator privileges

- Detecting and disabling working anti-viruses
- Encrypting user files on the device
- Stealing contact information
- Starting device camera and taking photos
- Locking or unlocking the device
- Turning ring and notification tones off
- Showing threatening short messages

As a result, the use of unique features in the detection of Ransomware will increase the detection speed and accuracy. For this reason, it has become necessary to work specifically for Ransomware. Although there have been various studies to detect malicious ransomware, there are not many studies on Android ransomware detection. Automated malicious software analysis can be performed via two traditional methods, static and dynamic. Static analysis is basically the investigation of resources obtained via re-compilation before executing the code. Static analysis is a investigation simple and fast method. Dynamic analysis, on the other hand, is a method that investigates behavior emerging from the execution of the software on either a real machine or a virtual environment. In this study, static analysis is used for ransomware investigation. In this study, a dataset of 2990 samples in total was used for feature selection. As a result, Bagging-MARS was more efficient than the other feature selection methods it was compared with. The proposed method considerably exceeded Random Forest, one of the most commonly used and best-known embedded methods in the literature.

Also, the accuracy rate of Bagging-MARS surpasses Random Forest in all machine learning techniques used. In Section 2 of this study, a literature review is performed on related work. Section 3 covers some basic concepts related to the theoretical substructure of the proposed technique. In Section 4, information about the dataset and the evaluation metrics used are given, and the experimental results of the proposed technique are presented. In Section 5, the application results from the Bagging-MARS technique are discussed and interpreted in the conclusions.

II. Related Work

Recent research on Android ransomware detection has developed various approaches to make detection and prevention of such malware more effective. In this article, studies by Kirubavathi and Anne (2024) [11], Rahima et al. (2024) [12], and Li et al. (2024) [13] have contributed to significant developments in this field by highlighting different techniques and methods. Kirubavathi and Anne used behavioral analysis and machine learning techniques in the detection of Android ransomware in their study. This study focuses on identifying specific behavioral patterns of ransomware, thus enabling the detection of malware. The proposed model optimizes the detection process by monitoring the behaviors of ransomware on Android devices. With this approach, early warning systems are being developed to prevent the spread of ransomware. Rahima and his colleagues presented an approach based on hamming distance, a new feature selection method, for the detection of Android ransomware. This study aimed to increase the accuracy of the detection model by using this new method in the feature selection phase. Rahima and his team managed to make the detection of ransomware more sensitive and faster with this method, and especially emphasized the benefits of hamming distance. Li and his colleagues developed an Android ransomware detection framework called ARdetector. This framework detects ransomware using both static and dynamic analysis methods. The study aims to provide a multi-layered defense mechanism against ransomware attacks, preventing attacks from being hidden in various ways. ARdetector has been presented as a powerful tool in the analysis of Android applications and has achieved high accuracy rates in ransomware detection.

Also, the feature selection stage is a critical step in obtaining an efficient classifier model because, along with the effect of input data on designing a strong classifier, it is directly affected by long execution times and classifier accuracy.

Feature selection and population classification has caught the attention of many researchers in statistics, machine learning, neural networks and data mining for years. Fast Correlation-Based Filtering (FCBF) was proposed by Deisy et al. (2007) [14] to remove both irrelevant and unnecessary features using symmetric similarity. When the recent studies on feature selection were investigated, Yıldız and Doğru (2019) [15] proposed a method to detect malicious Android software via feature selection with the Genetic Algorithm (GA). Three different classification techniques consisting of separate feature subsets chosen by GA were applied to detect and comparatively analyze malicious Android software.

Chakravarty, (2020) [16] investigated the most effective permission recognition using feature reduction. They used Gain Ratio and J48 to evaluate the features selected and features used for feature reduction in Random Forest, Multi-Layer Perceptron, Sequential Minimal Optimization (SMO) and Randomizable Filtered classifiers. Experimental results showed that five permissions may provide almost complete feature accuracy and therefore optimize the malicious detection software system.

Varma et al. (2020) [17] contributed to finding the minimal feature set for malicious software detection using a series of importance values of dependent features along with Ant Colony Optimization (ACO) as a heuristic search technique. The malicious software dataset called claMP, which has both integrated and raw features, was accepted as the comparison dataset for this study. Analytical results proved that claMP can achieve 97.15% and 92.8% data storage optimization with minimum loss of accuracy for integrated and raw datasets, respectively.

In published literature, there are few studies proposing static and/or dynamic approaches to detect Android ransomware. Some studies using static analysis – Andronio et al. (2015) [4]– investigated the common properties and analysis results for existing mobile ransomware families. They proposed HellDroid, which is a fast, effective and fully automated approach that recognizes harmless software, known and unknown malicious software and examples of them.

The working principle of Helldroid is generally based on detecting the building blocks necessary for mobile ransomware to work. It detects if an application is trying to lock or encrypt the device without user permission and if ransom requests are shown on screen. HellDroid shows almost zero false positives in a large dataset consisting of hundreds of apk files, including harmless/malicious software and ransomware.

Zheng et al. (2016) [18] proposed GreatEatlon, which is a next generation mobile ransomware detector. As a preventive countermeasure, they envisioned the deployment of GreatEatlon in the application store. In essence, GreatEatlon uses static program analysis techniques to extract the correct information dataflow necessary to resolve reflection-based, anti-analysis attempts and detect malicious usage of the device management API and cryptographic APIs.

Mercaldo et al. (2016) [19] proposed a method that can identify the characteristic properties of a ransomware program inside malicious software and detect the ransomware. According to experimental results, they claimed that the proposed method can be the correct way to develop commercial solutions that successfully detect ransomware and prevent their effects.

Maiorca et al. (2017) [10] utilized the information extracted from API packets that enabled characterizing applications without any special information regarding user-defined content such as the application language. Results obtained from the data showed that it is possible to detect Android ransomware and distinguish them from malicious software in general with a very high accuracy. Moreover, they correctly distinguished true ransomware from the false positives using R-PackDroid to identify applications detected as ransomware with a very low confidence by the Virustotal service.

Cimitille et al. (2017) [20] proposed a technique based on formal methods to detect malicious ransomware in Android devices. They made the method usable by implementing it in a tool called Talos. Results obtained showed that Talos was quite effective in recognizing ransomware even if it was hidden, and detected them with a 99% accuracy.

In some studies using dynamic analysis, Song et al. (2016) [21] proposed an effective method to prevent mutated ransomware attacks exploiting vulnerabilities in existing systems against novel ransomware patterns on Android platforms. The proposed technique was based on investigating these processes by processor usage, memory usage and input/output ratio based statistical methods to detect those processes exhibiting abnormal behavior. If a suspicious ransomware-executing process was detected, the proposed method stopped the process and the user was requested to delete the programs related to that process. The high detection speed was thanks to the method being applicable to Android source code instead of the mobile application.

Chen et al. (2018) [22] gathered 2,721 ransomware samples covering most of the existing Android ransomware families and proposed a method characterized systematically by various aspects including time diagrams and malicious intent features. Moreover, since the detection results of existing anti-virus tools are quite unsuccessful, they proposed a new real-time detection system called RansomProber to detect ransomware extorting users by encrypting data. RansomProber can deduce if file encryption services have been started manually or not by analyzing the user interface widgets of related activities and the user's finger movement coordinates. Experimental results showed that RansomProber could effectively detect encrypting ransomware with high accuracy and acceptable execution time performance.

On the other hand, there are studies that perform both dynamic and static analyses. When some of these studies were investigated, Ferrante et al. (2017) [23] proposed a hybrid method that could actively withstand ransomware. The proposed method consisted of a dynamic approach that first observed the execution time behavior of applications that were going to be used on a device before installation with a static approach, and then determined if the system was under attack or not. While they used the frequency of process codes in their static ransomware detection, the dynamic detection takes the CPU, memory and network usages and system call statistics

into account. They evaluated the performance of their hybrid detection technique in a dataset with both ransomware and legal applications. Results showed that although both the static and dynamic detection methods provided good performance in ransomware detection, the hybrid method showed the best performance, detecting ransomware with 100% sensitivity and having a false positive rate below 4%.

Gharib et al. (2017) [24] proposed the DNA-Droid which has a two-layer detection framework. They used a dynamic analysis layer on top of a static analysis layer as a complementary layer. In DNA-Droid, they used new features and a deep neural network to obtain a series of features with a powerful capacity to distinguish ransomware and harmless samples. Furthermore, sequence ordering techniques were used to profile ransomware families. A web system was developed to extract dynamic features for researchers. DNA-Droid has been tested against thousands of samples. Example results showed that it has a high precision and recall even in detecting unknown ransomware samples while keeping the false negative rate below 1.5%.

III. Theoretical Backround

A. Android Ransomware Permissions and Application Programming Interface Calls

A permission for an Android application developer is a constraint that limits access to documents, a part of the code, or data in the device. It is applied to protect critical data that can be limited, and protect the code. Every permission is defined with a unique label. In general, the label indicates the limited action.

Today, every Android application developed has a related AndroidManifest.xml file. The manifest file contains all the necessary details needed for the Android platform to execute the application since its compilation. Moreover, it gives information about application components such as services and activities.

Android applications are compressed files with an ".apk" extension. Android is developed via an Application Programming Interface (API) and consists of four types of components: activities, services, broadcast receivers and content providers. Android software interacts with applications using these components. In application packages, instead of multiple class files, all classes are packaged into a single file with a ".dex" extension. Android application packages are jar files that contain the application byte code, local code libraries, application resources and the AndroidManifest. The AndroidManifest is an XML file that contains information like the application package name and the application permissions. It is written in a human-readable XML format and transformed into binary XML during compilation [25].

Feature selection is a method used to select the most appropriate features that can be more easily classified into a specific class (malicious or harmless). In this study, the permissions and the API Calls parameters are used as features that can be helpful in determining if the Android software is malicious or not. To find these features, the Apktool reverse engineering tool is used. The AndroidManifest.xml and the smali files are investigated for permission requests and API Calls, respectively. These features were obtained by writing a bash script in the Ubuntu operating system. As a result, the feature matrix with a total of 134 permissions and API Calls was transformed into a feature matrix with the best 34 features.

B. The Proposed Method

Feature selection techniques identify and remove irrelevant and redundant information, enhancing the effectiveness of data mining algorithms [26]. As computer and information technologies have advanced, the analysis of multi-dimensional databases has become inevitable. The curse of dimensionality decreases classifier performance on high-dimensional datasets due to increased complexity, training requirements, and computation times. Reducing the number of features is essential to address this issue. Dimensionality reduction can be achieved through feature extraction or feature selection. Feature extraction compresses or transforms original features but lacks interpretability, while feature selection removes irrelevant and redundant features to choose the best subsets. Three feature selection algorithms are: Filter technique: Features are ranked and selected based on this ranking for model evaluation.Wrapper method: Features are tested with a machine learning algorithm to select the subset that improves model performance using heuristic methods like forward and backward selection.Embedded method: Combines filter and wrapper techniques to rank and select the highest-ranking features, enhancing classifier performance [27].In this study, the Bagging-MARS method is proposed for Android ransomware detection. This method achieved higher accuracy with 34 features compared to other techniques, as shown in Figure 3.

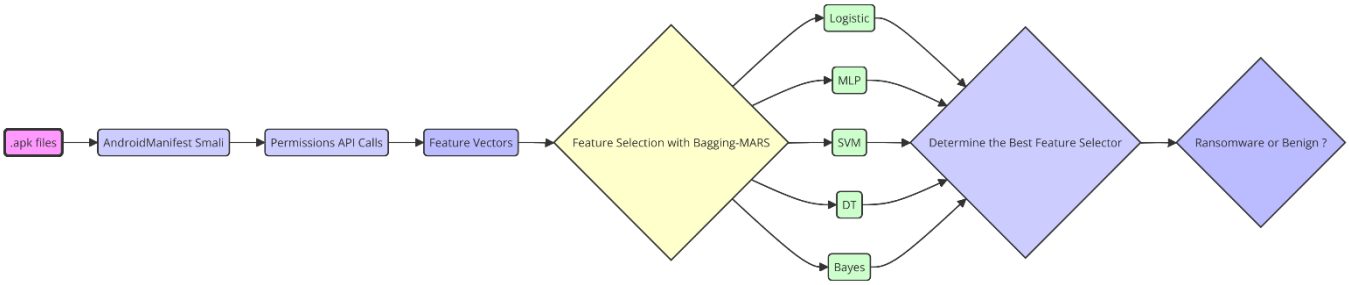

Figure 3. Architecture of the Proposed Technique

C. Bootstrapping

Bootstrap algorithms are used to create and resample large datasets. The bootstrap method reevaluates the statistical inferences of some parameters. Numerous resampling iterations are performed to make this process more reliable. Variance estimates are successfully obtained using the bootstrap method, which is frequently used for variance estimation. Additionally, the bootstrap method is superior when the sample distribution is not normal or when variance analysis is conducted on very small datasets. Bootstrapping is a method that is quite easy to understand and can be used with limited assumptions, without the need for intensive mathematical formulas [28].

A small number of samples with a high-dimensional feature space causes a decline in classifier performance in machine learning, statistics, and data mining systems. The results of feature selection methods significantly impact the success of data mining processes, especially when datasets are large. To obtain the best results from feature selection methods, it is necessary to conduct a comprehensive search in the search space and ensure accuracy in classification by checking each subset of features. It is not necessary to check each subset of features to achieve the same performance; it demonstrates that only a very small combination subset needs to be checked to achieve the same performance with a comprehensive approach.

The success of data mining algorithms stems from different factors. For instance, the quality of the input data is one of these factors. If the data contains irrelevant or redundant information or noise, the learning process across the search space will be more difficult. Feature selection techniques allow for the identification and removal of some irrelevant and redundant information. Such a process depends on the selection of a subset of optimal features that maximizes the efficiency of the data mining algorithm on the initial data. In the context of classification, bootstrapping repeats the entire classification experiment many times to obtain estimation accuracy from repeated experiments. Therefore, many bootstrap resamples are generated by (randomly) replicating each original sample to estimate the error rate in a few samples [29]. A sample of size m is taken from the original sample through resampling. Sampling with replacement means that some data points may be skipped [30].

D. Multivariate Adaptive Regression Splines (MARS)

MARS is a non-parametric regression method developed in the early 1990s by Jerome H. Friedman [31]. Designed for both binary and continuous outcome variables, MARS is known for its flexibility, accuracy, and speed. Unlike linear methods, MARS considers subsets of variables by dividing the predictor variable space into overlapping regions to form spline functions called basis functions [32, 33]. This method effectively handles non-monotonic relationships between predictor variables, making it superior in interpreting complex relationships in high-dimensional data compared to other linear and parametric methods [34, 35].

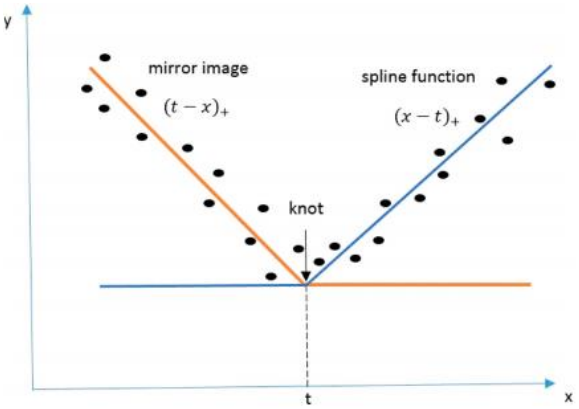

Figure 4. Inflection Point of the Spline Function

Friedman (1991) [36] recommended including the main region while determining sibling subregions, as this makes future splits more effective (Lewis and Stevens, 1991) [37]. MARS is used for prediction when the output variable is continuous and for classification when the output variable is categorical, demonstrating its wide applicability as a highly flexible, accurate, and fast technique [38]. Figure 4 shows the inflection point of a spline function [39].

In constructing the MARS model, the initial maximum model is refined using a backtracking algorithm that eliminates the least effective variables one by one. submodels generated during this process are compared using the Generalized Cross-Validation (GCV) criterion to determine the best-approximating submodel.

$$GCV = \frac{\sum_{i=1}^{N}\left(y_i - \hat{f}_\lambda(x_i)\right)^2}{(1-M(\lambda)/N)^2} \quad (1)$$

Here, $M(\lambda)$ and N show the effective parameter count and the number of observations, respectively. In this expression, $M(\lambda)$ is found by $M(\lambda) = r + cK$ where r and K denote the number of independent basis functions and the number of nodes selected in the incremental part, respectively. Bagging-MARS Pseudo Code is as follows.

Input: Dataset D (n samples, m features)
Output: Final Bagging-MARS Model
1. Determine hyperparameters:
- B: Number of bootstrap samples
- M: Maximum number of iterations for MARS model
- P: Penalty parameter
- Nk: Maximum number of nodes
2. Create an empty model list: Model_List = []
3. For b = 1 to B:
a. Create bootstrap sample: D_b = Bootstrap sample(D)
b. Train MARS model:
Model_b = MARS (D_b, Max_Iteration=M, Penalty=P, Nk=Nk)
c. Add model to model list: Model_List.append(Model_b)
End For
4. Make predictions for final model:
a. Make predictions on test data X:
Predictions = []
For each Model in Model_List:
Prediction = Model.predict(X)
Predictions.append(Prediction)
End For
b. Calculate the final prediction (average or majority vote):
Final_Prediction = Aggregate (Predictions)
5. Evaluate the final model:
a. Final_Accuracy = Evaluate (Final_Prediction, Actual Values)
b. Final_F_Measure = Calculate_F_Measure(Final_Prediction, Actual Values)
6. Return the final model and performance metrics as output.

IV. Experiment and Discussion

To evaluate the performance of machine learning methods for classifying Android ransomware, several metrics are used as follows:

$$ACC = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (2)$$

$$TPR = \frac{TP}{(TP+FN)} \quad (3)$$

$$TNR = \frac{TN}{(TN+FP)} \quad (4)$$

$$P = \frac{TP}{(TP+FP)} \quad (5)$$

$$F - M = \frac{(2\times P\times TPR)}{(P+TPR)} \quad (6)$$

In this study, 990 benign data points from the Google Play Store [40] and 2000 malicious data points from Virustotal [41], Andrototal, and Ransommobi [42] were collected as detailed in Table 1.

TABLE I. Android Datasets

Dataset	Source	Number of Samples	Features
Benign (Non-Malicious)	Google Play Store	990	Application permissions, API calls
Malicious	Virustotal, Andrototal, Ransommobi	2000	Application permissions, API calls

The study utilized thirteen feature selection methods, encompassing Bagging-MARS, Random Forest, AdaBoost, Naive Bayes, J48, Logistic Regression, Information Gain, Gain Ratio, Chi-Square, Correlation, OneR, and RRelief. For the classification task, five machine learning techniques were employed: Logistic Regression, Multi-Layer Perceptron (MLP), Support Vector Machines (SVM), Decision Trees (DT), and Bayesian methods.

TABLE II. Performance Evaluation Results (Precision)

	Machine Learning Tecniques				
Feature Selection Techniques	**Logistic**	**MLP**	**SVM**	**DT**	**Bayes**
Logistic Regression	0.670	0.681	0.660	0.665	0.663
Information Gain	0.660	0.604	0.660	0.660	0.783
Gain Ratio	0.660	0.604	0.660	0.660	0.660
Chi-Square	0.739	0.608	0.723	0.718	0.747
Correlation	0.842	0.849	0.842	0.852	0.856
One R	0.736	0.727	0.744	0.749	0.747
RRelief	0.760	0.754	0.760	0.760	0.760
J48	0.848	0.828	0.859	0.845	0.816
Naive Bayes	0.848	0.828	0.859	0.845	0.816
SVM	0.848	0.858	0.856	0.857	0.793
Random Forest	0.875	0.877	0.876	0.875	0.841
AdaBoost	0.591	0.616	0.587	0.586	0.649
Prop. Bagging-MARS	0.904	0.905	0.897	0.898	0.887

The precision performance results, as illustrated in Table 2, indicate that the Bagging-MARS method achieved the highest precision, with a value of 0.905. This was followed by Random Forest with a precision of 0.877, SVM with 0.858, and J48 with 0.828. In contrast, Information Gain and Gain Ratio exhibited the lowest precision, both scoring 0.604. Overall, it was observed that filter methods generally underperformed during the feature selection process for ransomware detection.

TABLE III. Performance Evaluation Results (F-Measure)

	Machine Learning Techniques				
Feature Selection Techniques	**Logistic**	**MLP**	**SVM**	**DT**	**Bayes**
Logistic Regression	0.640	0.682	0.643	0.667	0.665
Information Gain	0.643	0.604	0.643	0.643	0.565
Gain Ratio	0.643	0.603	0.643	0.643	0.643
Chi-Square	0.568	0.608	0.567	0.567	0.558
Correlation	0.796	0.796	0.796	0.795	0.795
One R	0.705	0.707	0.710	0.712	0.557
RRelief	0.753	0.752	0.753	0.753	0.753
J48	0.825	0.857	0.828	0.860	0.744
Naive Bayes	0.816	0.827	0.811	0.818	0.750
SVM	0.794	0.802	0.797	0.803	0.777
Random Forest	0.875	0.876	0.842	0.874	0.778
AdaBoost	0.554	0.621	0.572	0.571	0.651
Prop. Bagging-MARS	0.901	0.903	0.897	0.898	0.868

The F-Measure values, as presented in Table 3, reveal that the Bagging-MARS technique outperformed all other feature selection methods, achieving the highest F-Measure of 0.903. This was followed by Random Forest with an F-Measure of 0.876, and SVM with an F-Measure of 0.802. In contrast, Information Gain and Gain Ratio recorded the lowest F-Measure values among the methods evaluated.

TABLE IV. Performance Evaluation Results (Accuracy)

	Machine Learning Tecniques				
Feature Selection Techniques	**Logistic**	**MLP**	**SVM**	**DT**	**Bayes**
Logistic Regression	69.264	68.327	66.889	69.495	66.655
Information Gain	66.889	65.050	66.889	66.889	57.525
Gain Ratio	66.889	65.016	66.889	66.889	66.889
Chi-Square	68.127	65.217	68.026	67.993	56.488
Correlation	81.638	81.806	81.638	81.739	81.839
One R	69.699	69.966	70.167	70.334	56.388
RRelief	76.689	76.220	76.689	76.689	76.689
J48	83.712	85.652	84.281	86.154	74.114
Naive Bayes	83.077	83.110	82.943	83.144	74.248
SVM	81.605	82.308	81.973	82.375	79.197
Random Forest	87.424	87.559	85.451	87.391	77.191
AdaBoost	66.488	64.816	66.890	66.923	65.284
Prop. Bagging-MARS	90.033	90.268	89.715	89.765	86.421

The accuracy performance results, as detailed in Table 4, demonstrate that the Bagging-MARS technique achieved the highest accuracy, with a value of 90.268%. This was followed by Random Forest, which attained an accuracy of 87.559%, and SVM, with an accuracy of 82.308%. In contrast, Information Gain and Gain Ratio produced the lowest accuracy values among the evaluated methods. Notably, the method introduced in this study for the first time outperformed the widely recognized Random Forest method, which is frequently cited in the literature. Additionally, AdaBoost, another well-known ensemble method, recorded an accuracy of 64.816%. The Bagging-MARS method clearly outperformed AdaBoost in this context.

V. CONCLUSIONS

In this study, a MARS method based on Bagging feature selection was proposed. It has been shown that this technique has advantages compared to other existing methods for improving classification. To conduct a comparative analysis for the detection of unknown Android ransomware, thirteen feature selection methods and five machine learning methods were investigated. To observe the effectiveness of feature selection, standard Android permissions and permissions selected by Bagging-MARS were used with classifiers. The models were built from static analysis based on the investigation of application permissions. Experimental results showed that the best performance was produced by MLP with 90.268% accuracy and an F-Measure of 0.903. Bagging-MARS is the most effective feature selection method for improving classification techniques in this comparison. The proposed method significantly outperformed Random Forest, one of the most commonly used and well-known embedded methods in the literature. The accuracy rate of Bagging-MARS surpasses Random Forest across all machine learning techniques used. Bagging-MARS achieved 90.268% accuracy, while Random Forest remained at 87.559% accuracy.

In conclusion, this study clearly demonstrates the high efficiency of the MARS method based on Bagging feature selection in detecting Android ransomware. The detailed analysis thoroughly examined the impact of various feature selection methods and machine learning techniques on correct classification performance. In this context, the Bagging-MARS method shows a significant advantage over other alternatives, proving to be an extremely valuable tool, especially in the context of Android ransomware detection.

The results may encourage future security-focused studies to consider such feature selection and machine learning approaches more in-depth for ransomware detection and prevention on the Android platform. The larger-scale application of such methods can help develop more effective protection mechanisms against rapidly evolving ransomware threats. This study paves the way for further advancements in research on Android ransomware detection.

CONFLICT OF INTEREST

The author declares that there is no conflict of interest.

REFERENCES

[1] Rajput, T. S. (2017). Evolving threat agents: Ransomware and their variants. *International Journal of Computer Applications*, 164, 28–34.

[2] Uma, E., & Kannan, A. (2014). Improved cross site scripting filter for input validation against attacks in web services. *Kuwait Journal of Science*, 41(2).

[3] Nowinson, M. (2020). The biggest ransomware attack of 2020. CRN. https://www.crn.com/slide-shows/security/the-11-biggest-ransomware-attacks-of-2020-so-far

[4] Jesus, M. D., Malubay, M. & Ramos, A.C. (2020). Ransomware report: Avaddon and new techniques emerge, industrial sector targeted. TrendMicro. https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/ransomware-report-avaddon-and-new-techniques-emerge-industrial-sector-targeted

[5] Statcounter. (2020). Mobile operating system market share worldwide. Statcounter. https://gs.statcounter.com/os-market-share/mobile/worldwide

[6] Chebyshev, V. (2020). Mobile malware evolution 2019. Securelist. https://securelist.com/mobile-malware-evolution-2019/96280/

[7] C. E. (2020). Ransomware facts, trends & statistics for 2020. Safety Detectives. https://www.safetydetectives.com/blog/ransomware-statistics/

[8] Alsoghyer, S., & Almomani, I. (2019). Ransomware detection system for Android applications. *Electronics*, 8, 868.

[9] Andronio, N., Zanero, S., & Maggi, F. (2015). Heldroid: Dissecting and detecting mobile ransomware. In *Research in Attacks, Intrusions, and Defenses: 18th International Symposium, RAID 2015, Kyoto, Japan, November 2-4, 2015. Proceedings 18* (pp. 382-404). Springer International Publishing.

[10] Maiorca, D., Mercaldo, F., Giacinto, G., Visaggio, C. A., & Martinelli, F. (2017, April). R-PackDroid: API package-based characterization and detection of mobile ransomware. In *Proceedings of the symposium on applied computing* (pp. 1718-1723).

[11] Kirubavathi, G., & Anne, W. R. (2024). Behavioral-based detection of Android ransomware using machine learning techniques. *International Journal of System Assurance Engineering and Management*, 1–22.

[12] Manzil, H. H. R., & Naik, S. M. (2024). Android ransomware detection using a novel hamming distance-based feature selection. *Journal of Computer Virology and Hacking Techniques*, 20(1), 71–93.

[13] Li, D., Shi, W., Lu, N., Lee, S. S., & Lee, S. (2024). ARdetector: Android ransomware detection framework. *The Journal of Supercomputing*, 80(6), 7557–7584.

[14] Deisy, C., Subbulakshmi, B., Baskar, S., & Ramaraj, N. (2007). Efficient dimensionality reduction approaches for feature selection. *In 2007 International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007).*

[15] Yildiz, O., & Doğru, I. A. (2019). Permission-based Android malware detection system using feature selection with genetic algorithm. International *Journal of Software Engineering and Knowledge Engineering*, 29, 245–262.

[16] Chakravarty, S. (2020, June). Feature selection and evaluation of permission-based android malware detection. In *2020 4th International conference on trends in electronics and informatics (ICOEI)(48184)* (pp. 795-799). IEEE.

[17] Varma, R. K., Akhila, K., & Mallidi, S. K. R. (2020). Feature reduction and optimization of malware detection system using ant colony optimization and rough sets. *International Journal of Information Security and Privacy*, 14(3), 95–114.

[18] Zheng, C., Dellarocca, N., Andronio, N., Zanero, S., & Maggi, F. (2017). GreatEatlon: Fast, static detection of mobile ransomware. *In Security and Privacy in Communication Networks* (pp. 136–156). Springer International Publishing.

[19] Mercaldo, F., Nardone, V., & Santone, A. (2016). Ransomware inside out. *In 2016 11th International Conference on Availability, Reliability and Security* (pp. 628–637).

[20] Cimitile, A., Mercaldo, F., Nardone, V., Santone, A., & Visaggio, C. A. (2018). Talos: No more ransomware victims with formal methods. *International Journal of Information Security*, 17(6), 719–738.

[21] Song, S., Kim, B., & Lee, S. (2016). The effective ransomware prevention technique using process monitoring on Android platform. *Mobile Information Systems,* 2016, 2946735.

[22] Chen, J., Wang, C., Zhao, Z., Chen, K., Du, R., & Ahn, G. (2018). Uncovering the face of Android ransomware: Characterization and real-time detection. *IEEE Transactions on Information Forensics and Security*, 13(5), 1286–1300.

[23] Ferrante, A., Malek, M., Martinelli, F., Mercaldo, F., & Milosevic, J. (2017). Extinguishing ransomware - A hybrid approach to Android ransomware detection. *In Proceedings of the 10th International Symposium on Foundations and Practice of Security* (pp. 49–64). Springer International Publishing.

[24] Gharib, A., & Ghorbani, A. (2017). DNA-Droid: A real-time Android ransomware detection framework. *In Proceedings of the 11th International Conference on Network and System Security* (pp. 256–272). Springer International Publishing.

[25] Rastogi, V., Chen, Y., & Jiang, X. (2013). DroidChameleon: Evaluating Android antimalware against transformation attacks. *In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security* (pp. 329–334).

[26] Diaz-Diaz, N., Aguilar-Ruiz, J. S., & Nepomuceno, J. A. (2005). Feature selection based on bootstrapping. *In Proceedings of the 2005 ICSC Congress on Computational Intelligence Methods and Applications.*

[27] Ilham, S., Abderrahim, G., & Abdelhakim, B. A. (2018). Permission based malware detection in Android devices. *In Proceedings of the 3rd International Conference on Smart City Applications* (pp. 83). Association for Computing Machinery.

[28] Simon, J. L., & Bruce, P. C. (1991). Resampling: A tool for everyday statistical work. *Chance*, 4(1), 22–32.

[29] Efron, B. (1983). Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the American Statistical Association*, 78(382), 316–331.

[30] Pokhriyal, A. (2021). What is bootstrap sampling in statistics and machine learning? Analytics Vidhya. https://medium.com/analytics-vidhya/what-is-bootstrapping-in-machine-learning-777fc44e222a

[31] Banks, D. (2001). Exploratory data analysis: Multivariate approaches (Nonparametric regression). *In International Encyclopedia of the Social & Behavioral Sciences* (pp. 5087–5092). Elsevier.

[32] Muñoz, J., & Felicísimo, Á. M. (2004). Comparison of statistical methods commonly used in predictive modelling. *Journal of Vegetation Science*, 15(2), 285–292.

[33] Put, R., Xu, Q. S., Massart, D. L., & Vander Heyden, Y. (2004). Multivariate adaptive regression splines (MARS) in chromatographic quantitative structure–retention relationship studies. *Journal of Chromatography A*, 1055(1), 11–19.

[34] Olecka, A. (2007). Beyond classification: Challenges of data mining for credit scoring. In *Knowledge Discovery and Data Mining: Challenges and Realities* (pp. 139-161). IGI Global.

[35] Xu, Q. S., Daeyaert, F., Lewi, P. J., & Massart, D. L. (2006). Studies of relationship between biological activities and HIV reverse transcriptase inhibitors by multivariate adaptive regression splines with curds and whey. *Chemometrics and Intelligent Laboratory Systems*, 82(1–2), 24–30.

[36] Friedman, J. H. (1991). Multivariate adaptive regression splines (with discussion). *The Annals of Statistics*, 19(1), 1–141.

[37] Lewis, P. A. W., & Stevens, J. G. (1991). Nonlinear modeling of time series using multivariate adaptive regression splines (MARS). *Journal of the American Statistical Association*, 86(416), 864–877.

[38] Mukhopadhyay, A., & Iqbal, A. (2009). Prediction of mechanical property of steel strips using multivariate adaptive regression splines. *Journal of Applied Statistics*, 36(1), 1–9.

[39] Ağraz, M., & Purutçuoğlu, V. (2019). Extended lasso-type MARS (LMARS) model in the description of biological network. *Journal of Statistical Computation and Simulation*, 89(1), 1–14.

[40] Google. (2020). Google. Access date:2021. http://play.google.com/store

[41] Virustotal. (2020). Virustotal. Access date:2021. https://www.virustotal.com

[42] Ransommobi. (2020). Ransommobi. Access date:2021. https://www.ransommobi.com