Hindawi
Complexity
Volume 2022, Article ID 6760920, 18 pages
https://doi.org/10.1155/2022/6760920

WILEY | Hindawi

*Research Article*

# An Empirical Evaluation of Supervised Learning Methods for Network Malware Identification Based on Feature Selection

**C. Manzano**,[1] **C. Meneses**,[2] **P. Leger**,[1] **and H. Fukuda**[3]

[1]*Escuela de Ingeniería, Universidad Católica Del Norte, Antofagasta, Chile*
[2]*Departamento de Ingeniería de Sistemas y Computación, Universidad Católica Del Norte, Antofagasta, Chile*
[3]*Shibaura Institute of Technology, Tokyo, Japan*

Correspondence should be addressed to P. Leger; pleger@ucn.cl

Received 15 November 2021; Revised 6 February 2022; Accepted 5 March 2022; Published 7 April 2022

Academic Editor: Giacomo Fiumara



Malware is a sophisticated, malicious, and sometimes unidentifiable application on the network. The classifying network traffic method using machine learning shows to perform well in detecting malware. In the literature, it is reported that this good performance can depend on a reduced set of network features. This study presents an empirical evaluation of two statistical methods of reduction and selection of features in an Android network traffic dataset using six supervised algorithms: Naïve Bayes, support vector machine, multilayer perceptron neural network, decision tree, random forest, and K-nearest neighbors. The principal component analysis (PCA) and logistic regression (LR) methods with $p$ value were applied to select the most representative features related to the time properties of flows and features of bidirectional packets. The selected features were used to train the algorithms using binary and multiclass classification. For performance evaluation and comparison metrics, precision, recall, F-measure, accuracy, and area under the curve (AUC-ROC) were used. The empirical results show that random forest obtains an average accuracy of 96% and an AUC-ROC of 0.98 in binary classification. For the case of multiclass classification, again random forest achieves an average accuracy of 87% and an AUC-ROC over 95%, exhibiting better performance than the other machine learning algorithms. In both experiments, the 13 most representative features of a mixed set of flow time properties and bidirectional network packets selected by LR were used. In the case of the other five classifiers, their results in terms of precision, recall, and accuracy, are competitive with those obtained in related works, which used a greater number of input features. Therefore, it is empirically evidenced that the proposed method for the selection of features, based on statistical techniques of reduction and extraction of attributes, allows improving the identification performance of malware traffic, discriminating it from the benign traffic of Android applications.

## 1. Introduction

Malware is short for malicious software, it is a generic term widely used to name all the different types of unwanted software programs [1]. There are various types of malware such as viruses, scareware, ransomware, ad-ware, spyware, smsware, etc. [2]. Cybercriminals have used malware as a network attack weapon to encrypt and hijack personal computer data, steal confidential information from information systems, penetrate networks, bring down servers, and cripple critical infrastructure [2]. These attacks often cause serious damage and generate significant economic losses [3].

According to a June 2020 report delivered by Kaspersky Lab, the number of malware attacks from 2018 to 2019 increased by 37% and reached 1,169,153 new cases at the end of last year. Also, McAfee Labs observed that during the first quarter of 2020 the number of malware threats to mobile applications was 375 per minute [4]. Today, one of the mobile platforms most affected by malware attacks is Android [5]. Generating new solutions that allow the detection and identification of new types of malware is a challenge that cybersecurity research communities must address to prevent the exploitation and misuse of current systems.

In the literature, to support the detection and identification of malware, three analysis techniques are proposed:

static analysis, dynamic analysis and network analysis [6]. Static analysis is mainly based on the study of malware source codes and is easily bypassed through code obfuscation [7]. Dynamic analysis focuses on using operating system calls to extract reliable information from malware execution traces [8]. The main disadvantage of dynamic analysis is to find the exact traceability of the behavior of the malware by being in a controlled environment called sandbox [9].

Unlike static and dynamic analysis techniques, which are based on the recognition of malware code and behavior within a host [10], network analysis allows the recognition of malware behavior according to the direct or passive features of conversations of a network flow [6]. The flow of the network can be seen as a set of conversations that is represented as a statistical summary of the network traffic between a source IP (Internet Protocol) and destination IP [11]. Network analysis has raised additional challenges such as data encryption and port obfuscation in network malware behavior [12]. One of the network analysis techniques to identify malware is the classification of network traffic with machine learning [13]. In the empirical works of [14–16], the network traffic classification method with machine learning has shown good results in the identification of malware. However, a common problem with this method is to adapt, on certain occasions, to high-dimensional datasets with irrelevant and redundant features to accurately classify and identify the types of malware ([17, 18]).

Feature reduction is a critical activity within the data preprocessing stage of a machine learning project [19] and especially for a network traffic classification problem due to the emergence of new network service traffic patterns and to the great demand for bandwidth [20]. The goal of feature reduction is to obtain a reduced representation of the original dataset that has not been processed. Wavelet Transform, PCA (Principal Component Analysis), Clustering, sampling, and traditional feature selection techniques such as Wrapper, Embedded and Filter, are methods used within the feature reduction phase at the stage of data preprocessing of a machine learning project [21]. Reducing or selecting a minimum number of features to represent the behavior of network traffic is a key task to achieve good performance in the malware detection and identification process [22].

Recently, in [21, 23] it is evidenced that researchers have adopted statistical methods of reduction and selection of features in order to improve the performance of detection and identification of malware. This is the case of [24], where the PCA statistical method is applied to reduce the features of the Application Programming Interface (API) type of the MamaDroid malware detection system. The authors in [24] initially worked with 116,281 features and managed to reduce their dimension to 10 main components. MamaDroid scored a good 99.9% performance for F-measure, and averaged over 90% accuracy and recall for all its malware detection experiments. In [25], experimental work was performed using the support vector machine (SVM) classifier to detect malware. In [25] 20 features of OpCode (operation code) were used and it was possible to reduce the initial dimension to 8 components by means of the PCA statistical method. The PCA method, applied in [25], managed to represent 99.5% of the total variance of its components. In [25], K-nearest neighbors (KNN) performed well at 83.41% accuracy, and 4.2% false negatives (FN) in detecting malware. Another study carried out in [26], applied the Sparse Logistic Regression (SLR) method to discriminate the less significant features of the model and improve the classification of malware attacks with its intrusion detection system (IDS). The SLR method was able to discriminate 4 features from the initial dataset of 20 features. In [26] a $p$-value of 0.5 was used, and overfitting and feature redundancy were controlled by simultaneously selecting and classifying them. SPLR achieved good malware detection performance of 97.6% overall accuracy and a total of 0.34% false positives (FP). In [27], a method called 4-LFE (L1-L2-LR-LDA Feature Extraction) is presented, composed of statistical techniques such as L1-L2 Penalty, logistic regression and linear discriminant analysis (LDA), to reduce the dimension of features and detect malware. The experimental results of the 4-LFE method show that it managed to classify the malware with 99.99% accuracy.

This paper presents the results of an empirical comparison of the performance of shallow learning algorithms, such as Naïve Bayes, support vector machine, multilayer perceptron neural network, decision tree, random forest, and K-nearest neighbors, to identify malware traffic. Two statistical techniques, PCA and logistic regression with $p$-value, were considered to reduce and select the most significant features related to the time of flows and bidirectional packets of the dataset with CICAndMal2017 Android network traffic. This work seeks to contribute through:

(1) The proposed feature selection methodology based on a combination of statistical and computational methods.

(2) The comparative analysis of different machine learning algorithms when applied to the identification of malware traffic based on different sets of preselected features. This provides empirical evidence that a feature selection method based on statistical and computational techniques generates better predictive results in relation to the use of all features without prior selection, particularly in the domain of identifying malware versus benign traffic.

The rest of the work is structured as follows. The following section discusses material and methods used in this study. Then, this paper describes the dataset and the methods used to perform the feature selection. After, we explain the four-phase methodology proposed for this work. Using this methodology, this paper presents performance of the experiments with the associated results. Then, the results are compared to those obtained in related work, regarding the identification of malware in network traffic considering methods of reduction and selection of features. Finally, the conclusions and future work are presented.

## 2. Methods and Materials

In this section, the CICAndMal2017 dataset used is first described, and second, the data preprocessing for binary and multiclass classification is explained. Finally, the feature

selection methods used as part of the methodology proposed in this work are presented.

*2.1. Dataset.* The CICAndMal2017 dataset is made up of a combination of more than 80 flow time and network packet features to detect and identify malware traffic alongside benign Android applications. This set was built by Lashkari et al. [28] of the Canadian Cybersecurity Institute (CIC). CICAndMal2017 offers 2,126 files in CSV (comma separated values) format and more than 20 gigabytes in PCAP (packet capture) files with malware traffic conversations and benign Android mobile network applications, captured in the years 2015–2017. Network traffic of benign applications are labeled "benign." Malware traffic is labeled into four categories: adware, ransomware, scareware, and smsware. Each category of malware consists of different families as presented in Table 1 [11]. Originally both sets of CSV and PCAP files are structured with more than 80 Android network traffic features.

*2.2. Feature Selection Methods.* Two statistical methods of feature selection are described below, known as PCA and Logistic Regression, which were selected and used for their good performance in features selection work in the detection and identification of malware [26, 27, 29–33]. The PCA and logistic regression methods were used to select the most representative features of network traffic from the input data of the CICAndMal2017 dataset.

*2.3. Feature Selection Based on Principal Component Analysis (PCA).* PCA is a method used to reduce the dimensionality of a large dataset to a smaller one, containing a large part of the information from the original set [24]. Reducing the number of features in a dataset sometimes means losing valuable information, but it also means simplifying the problem, since it is easier to explore and visualize data in small sets [34]. The PCA method therefore allows condensing the information provided by multiple variables into only a few components and having the value of the original variables to calculate these components [35]. Therefore, PCA decomposes a dataset into eigenvectors and eigenvalues. An eigenvector is a direction, for example $(x, y)$, and an eigenvalue is a number that represents the value of the variance in that direction [34]. The main component will be the eigenvector with the highest eigenvalue. There are as many eigenvector/eigenvalue pairs in a dataset as there are dimensions. The eigenvectors do not modify the data, but rather allow us to see them from a different point of view, more related to the internal structure of the data, and with a much more intuitive view of them [30]. Once the eigenvalues, which are a measure of the variance of the data, have been ordered, it is necessary to decide which is the smallest number of eigenvectors or principal components to maintain. To do this, a metric known as explained variance is used, which shows how much variance can be attributed to each of these principal components. Furthermore, as defined in [35], the principal components can be conceptualized as new axes that offer a new coordinate system to evaluate the data, making the differences between the observations in the dataset more visible. The PCA tries to put as much information as possible in the first component, then as much information as possible in the second component, and so on. This process is done until you have a total of principal components equal to the original number of features. As mentioned in [35], there is no single answer or method that allows identifying the optimal number of principal components to select. A very widespread way of proceeding consists of evaluating the proportion of accumulated explained variance and selecting the minimum number of components beyond which the increase is no longer substantial.

In other words, PCA corresponds to a linear transformation that takes the input data to a new space of orthogonal axes. In this new space, the axes are ordered such that the first axis captures the largest variance of the original data (called the first principal component), and the last axis captures the smallest variance. Formally [36], let $X$ be a data matrix of dimensions $n \times p$, where each column of data is previously normalized to have zero mean. Here $n$ and $p$ correspond to the number of observations and the number of columns or features of the data set, respectively. In mathematical terms, PCA defines a set of $l$ vectors of weights or coefficients $w_k$, each of dimension $p$, which transforms each row vector $x_i$ from matrix $X$ to a new vector $t_{k,i}$ in the space represented by the $l$ principal components. The transformation of each $x_i$ into a new vector $t_{k,i}$ is calculated as defined in equation (1).

$$t_{k,i} = x_i \cdot w_k, \tag{1}$$

where $i = 1, \ldots, n$ and $k = 1, \ldots, l$. Each of the principal components successively captures the maximum possible variance from the original data in matrix $X$. In order to reduce dimensionality, $l < p$ is usually considered.

The data matrix $X$ is decomposed by PCA as $T = XW$, where $W$ is a weight matrix, of dimensions $p \times p$, and its column vectors correspond to the eigenvectors of the matrix $X^T X$. These eigenvectors turn out to be proportional to the covariance matrix obtained from the data set $X^T$. In other words, PCA diagonalizes the covariance matrix obtained from the data sample. In matrix terms, this can be stated as $Q \propto X^T X = W \Lambda W^T$, where $\Lambda$ is the diagonal matrix of eigenvalues of $X^T X$.

Notably, PCA transforms a data vector $x_i$, of dimension $p$, into new $p$ variables that are uncorrelated in this new space. Given the different levels of variance captured by each component, not all of them need to be preserved. For example, keeping only the first $L$ components (eigenvectors), this results in a truncated version of the transformation $T_L = XW_L$, where $T_L$ is a matrix of $n$ rows but with only $L$ columns.

*2.4. Feature Selection Based on Logistic Regression with p Value.* The Logistic Regression method is generally used to test the importance or estimate the relationship between a dependent variable, dichotomous binary response, as a

Table 1: Category and family of malware.

| Category | Family type | | | |
|---|---|---|---|---|
| Adware | Edwin | Koodous | Kemoge | Dowgin |
| | Mobidash | Youmi | Feiwo | Selfmite |
| Ransomware | R Shuanet | | Gooligan | |
| | Charger | Pletor | LockerPin | Jisut |
| | RansomBO | Svpeng | PornDroid | Koler |
| Scareware | WannaLocker | | Simplocker | |
| | AndroidDefender | FakeAV | FakeApp.AL | FakeJobOff |
| | AVforAndroid | Penetho | VirusShield | FakeApp |
| Smsware | FakeTaoBao | AVpass | AndroidSpy.277 | |
| | BeanBot | Jifake | FakeNotify | Biige |
| | Nandrobox | FakeInst | Mazarbot | Plankton |
| | FakeMart | | SMSsniffer | |

function of a single quantitative variable (called Univariate Regression) or of a set of continuous independent variables (called Multivariate Regression) [37]. Regression analysis is a popular statistical process used for modeling and data analysis, indicating significant relationships and impact between the predicted target and the features under study [38]. In a logistic regression model, the evaluation of the fulfillment of the null hypothesis is based on the degree of relationship between the class attribute and each independent attribute of the model, determined by the level of significance and quantified by the $p$-value [39]. In our work, the null hypothesis corresponds to the nonassociation between the network traffic features and the malware class.

In general, the level of significance quantifies the possibility of accepting an erroneous conclusion, that is, of determining that there is an association when in fact there is not [33]. For example, a significance level (usually denoted by $\alpha$) of 0.05 establishes a 5% risk of accepting a relationship when there is none.

In other words, this represents a 95% certainty that the association we are studying is not due to chance. Therefore, if we want to work with a 99% safety margin, it has an implicit $p$-value of less than 0.01. Therefore, $p$ value $\leq\alpha$ indicates that the association is statistically significant. If $p$-value $\geq\alpha$, the association is not statistically significant [40].

The formal mathematics underpinning the Logistic Regression method is briefly described in [41] and summarized in the following paragraphs.

Let $y_i$ and $x_{i,j}$ be the value of the dependent variable and the value of the $j$-th independent variable ($j = 1, \ldots, k$), for the $i$-th observed data, respectively. The variable $y_i$ denotes a binary variable, which determines whether or not the $i$-th data observed belongs to a given group, being $y_i = 1$ when the data belongs to a group, and $y_i = 0$ in the case not belong to that group. The probability that $y_i = 1$ corresponds to $p_i$. All these variables are formally related as

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n. \quad (2)$$

In (2) the odds is given by $p/(1-p)$ and it represents the likelihood that the event will occur. In this context, the natural logarithm of $p_i/(1-p_i)$ is equal to the log odds, which allows us to transform a probability in the range 0 to 1 into a value in the range $(-\infty, +\infty)$. In order to isolate the value of $p$, we raise both sides of the (2) to $e$, which eliminates the natural logarithm of the left side:

$$\left(\frac{p}{1-p}\right) = e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\ldots+\beta_n x_n}. \quad (3)$$

This expression can be manipulated to isolate the value of p:

$$p = \left(\frac{1}{1+e^{-x}}\right), \quad (4)$$

where $x$ stands for $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n$. This expression turns out to be the Sigmoid or Logistic Function, given by equation (5).

$$Sigmoid(x) = \left(\frac{1}{1+e^{-x}}\right). \quad (5)$$

## 3. Methodology

The work methodology consists of a sequence of four phases that are part of a standard machine learning project [21]:

(1) Analysis and preprocessing
(2) Feature selection
(3) Classifier selection and training
(4) The evaluation of the classifier

Figure 1 shows these phases. In summary, the methodology first selects two types of datasets, one for each method, then evaluates the different identification algorithms with these datasets. Finally, these results are compared. Each phase of this methodology is described in detail below.

*3.1. Analysis and Preprocessing.* The conversations of network traffic of malware and of benign Android applications correspond to the dataset called CICAndMal2017 in CSV format [28]. The idea behind the network conversation level approach delivered by CICAndMal2017 is to present the behavior patterns of network traffic between two or more hosts on the network.

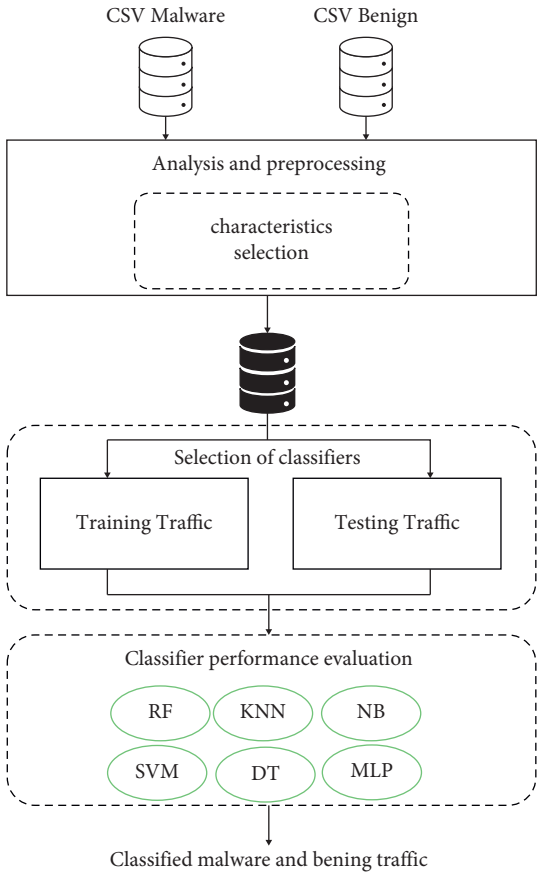


Figure 1: Four phases that are part of a standard machine learning project.

Using an application programmed with the use of the Sklearn Python library, the network traffic conversations from the CSV datasets of malware and CSV of benign applications were combined (See Figure 1). The total size of the consolidated dataset for this work is 37.8 MB corresponding to 245,138 observations of network traffic for ransomware, adware, scareware, and benign software. In addition, 15 features of network traffic conversations defined in [42] were initially separated (See Table 2). No normalization was applied to the data, since this process was initially carried out in [28]. Also, packets containing TCP (Transmission Control Protocol) retransmissions or other errors are discarded in [28]. The size of our dataset corresponds to 4.4% of the total CICAndMal2017 set.

A classification can be categorized according to the number of classes to be discriminated. In this form, we can deal with binary classification (two classes, one positive and the other the negative one) or multiclass classification (when the number of classes is more than two). Several issues arise when dealing with multiple classes in classification tasks, mainly the problem of imbalanced classes ([43–47]).

Our experiments include data preprocessing for binary and multiclass classification tasks. Specifically, binary classification (malware detection) preprocessing do not require the use of data balancing techniques, because the number of malware and benign application network traffic observations are evenly distributed in the CICAndMal2017 dataset (See Table 3). The total malware class traffic corresponds to the aggregation of ransomware, ad-ware and scareware traffic observations.

For the data preprocessing in the multiclass classification task, the CICAndMal2017 data set is divided into four types of classes, that is, the positive classes to be identified as "scareware", "ransomware" and "adware"; together with the negative class named "benign software". For the positive classes it is not necessary to balance them since their amounts of network traffic observations are approximately evenly distributed in the data. Due to the fact that the focus of the learning is to discriminate between the different types of malware and, despite the fact that the negative class "benign software" has a ratio of 3 : 1 with respect to each of the positive classes (See Table 3), it is decided not to under sample the negative class for balance it with positive classes.

*3.2. Feature Selection.* The selection of features is a fundamental stage in the process of recognition and enumeration of machine learning algorithms patterns, since the vast majority of these algorithms lack metrics that allow them to evaluate the relevance of an attribute for the prediction of the class attribute. Without this prior "filter," these algorithms can be confused by irrelevant attributes, notoriously deteriorating their performance.

The PCA and Logistic Regression (LR) methods, presented in the section Methods and Materials, were used to

Table 2: Features of network traffic conversations [42].

| No. | Feature | Description |
|---|---|---|
| 1 | Flow duration | Duration of the flow in microsecond |
| 2 | Flow byts/s | Number of flow bytes per second |
| 3 | Tot fwd pkts | Total packets in the outgoing |
| 4 | Tot bwd pkts | Total packets in the incoming |
| 5 | Fwd pkt len min | Minimum size of packet in outgoing |
| 6 | Fwd pkt len max | Maximum size of packet in outgoing |
| 7 | Fwd pkt len mean | Mean size of packet in outgoing direction |
| 8 | Fwd pkt len std | Standard deviation size of packets in outgoing |
| 9 | Bwd pkt len min | Minimum size of packet in incoming |
| 10 | Bwd pkt len max | Maximum size of packet in incoming |
| 11 | Bwd pkt len mean | Mean size of packet in incoming direction |
| 12 | Bwd pkt len std | Standard deviation size of packets in incoming |
| 13 | Tot len fwd pkts | Minimum length of a flow |
| 14 | Tot len bwd pkts | Maximum length of a flow |
| 15 | Mean len pkts/s | Mean length of a flow |
| 16 | Label | Type of malware |

Table 3: Distribution of observations that were used for all classifiers.

| Type | Malware traffic | | | Benign traffic |
|---|---|---|---|---|
| | Ransomware | Adware | Scareware | Benign |
| Total samples | 41,100 | 40,866 | 39,672 | 123,500 |
| Train samples | 32,880 | 32,694 | 31,738 | 98,800 |
| Test samples | 8,220 | 8,172 | 7,934 | 24,700 |

select the most representative features of network traffic from the input data of the CICAndMal2017 dataset. PCA and LR used the initial 15 network traffic features to create a new subset of mixed features between incoming and outgoing packets and network time streams.

*3.3. Selection of Classifiers and Training.* Six supervised machine learning algorithms were chosen for the classification of network traffic, with the aim of identifying the traffic of malware and benign Android applications. In the literature, the algorithms Random Forest, K-Nearest Neighbors, Decision Tree, Naïve Bayes, Multilayer Perceptron Neural Network and Support Vector Machine, have shown good performance in the classification of network traffic with features of time properties and packet flow (e.g., [11, 28, 42, 48, 49]). In the following, a brief explanation is provided regarding the machine learning algorithms used in this work to estimate the predictive performance of each set of features and for each method used to reduce the dimensionality of the dataset.

*3.3.1. Random Forest.* Random Forest was proposed by Braimanis in [50]. Random forest is a classifier consisting of a collection of tree-structured classifier defined as [51]:

$$\{h(x, \theta_k), k = 1, 2, \ldots i, \ldots\}, \tag{6}$$

where $h$ represents the random forest classifier, the $\theta_k$ are independent identifically distributed random vectors and each tree casts a unit vote for the most popular class at input $x$ [52]. Random forest generates an ensemble of decision trees to classify a new object from an input vector. The input vector is run down each of the trees in the forest. Each tree gives a classification and each tree votes for the class. Regarding the training data, a subset of the data is created for each tree of the forest by using bootstrapping sampling. The chance of overfitting is significantly reduced in comparison to individual decision tree, and there is no need to prune the trees.

K-nearest neighbor is a nonparametric classification and regression method [53, 54], where the input considers the $k$ closest training examples in a dataset. In k-NN classification, an input object is classified by a plurality vote of its $k$ nearest neighbors ($k > 0$ and integer), while in k-NN regression the output is the average of the values of $k$ nearest neighbors. k-NN is a lazy method, where the function is locally approximated and computation is delayed until function evaluation. k-NN relies on the distance computation, where common distance functions can be Euclidean, Manhattan or Minkowski (equations (7)–(9), respectively).

Euclidean equation:

$$\sqrt{\sum_{i=1}^{n} (X_i - Y_i)^2}. \tag{7}$$

Manhattan equation:

$$\sum_{i=1}^{n} |X_i - Y_i|. \tag{8}$$

Minkowski equation:

$$\left( \sum_{i=1}^{k} \left( |X_i - Y_i| \right)^q \right)^{1/q}. \tag{9}$$

The training data are vectors in a multidimensional feature space, each one with an associated class label. During training, the algorithm only stores the feature vectors and class labels. During classification, where $k$ is a user-defined constant, an unlabeled $x$ vector (named a query or test point) is classified by assigning the most frequent label among the $k$ nearest training examples (neighbors) to the given query point. In the case of discrete (nominal or ordinal) variables, Hamming distance can be used. In other domains (e.g. gene expression microarray data) correlation coefficients may be used as a distance metric (e.g., Pearson and Spearman correlation coefficients [55]).

*3.3.2. The Decision Tree.* The Decision Tree algorithm is a supervised learning approach that builds a predictive model that has a graphical representation. The tree is built by choosing features at the nodes of the tree and arcs associated with the values of the attributes used in the decision tree. In general, the generation of a decision tree is carried out in three stages: selection of features, construction of the tree (its nodes and arcs), and a final stage of pruning the resulting tree [56]. For the experimental process of the decision tree algorithm, the Gini coefficient was used for selection of features [57].

The C4.5 algorithm [58] used in this work bases its operation on determining at each step the most predictive attribute with respect to the class attribute, creating a node in the tree for this attribute and dividing the data based on the values of this selected attribute. The division criteria based on this attribute is calculated in the following five steps:

First, the expected information required to classify an observation in a data set $D$, is determined according to the expression shown in equation (10).

$$Info(D) = \sum_{i=1}^{m} p_i \log_2(p_i), \tag{10}$$

where $p_i$ represents the probability that an observation in the data set $D$ corresponds to the class $C_i$. In this case, $m$ represents the cardinality of the class attribute.

Second, the expected information needed to classify an observation by partitioning the data set $D$ by the $v$ values of an attribute $A$, is determined according to the expression shown in equation (11).

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j). \tag{11}$$

The $j$-th partition ($j = 1, \ldots, v$) has a weight represented by the term ($|D_j|/|D|$).

Third, the information gain when the attribute $A$ is used to partition the data set $D$, is determined according to the expression of equation (12).

$$Gain(A) = Info(D) - Info_A(D). \tag{12}$$

Fourth, calculate the split information of attribute $A$ with $v$ values, as shown in (13):

$$SplitInfo_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right). \tag{13}$$

Fifth, calculate the gain ratio of attribute $A$, as shown in (14):

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}. \tag{14}$$

In each node of the tree under construction, the attribute with either the highest information gain or gain ratio is selected.

*3.3.3. The Naïve Bayes.* The Naïve Bayes classifier is based on Bayes' theorem assuming conditional independence between the independent or predictor variables, given a value of the class attribute (dependent variable). Despite its simplicity, it often shows surprisingly good performance and is widely used, in some cases improving the classification results obtained with more sophisticated methods. Bayes' theorem provides a method to calculate the posterior probability of the class to which the object to be classified belongs. The Naïve Bayes classifier assumes that the effect of the value of one predictor variable is independent of the values of another predictor variable, given one class value. This assumption is called conditional independence of the class [59].

Mathematically, given a vector of features $X = (x_1, x_2, x_3, \ldots, x_n)$ and a class variable $y$, Bayes theorem states that [60]

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}. \tag{15}$$

Thus, the posterior probability $P(y|X)$ is calculated from the likelihood $P(X|y)$, the prior probability $P(y)$ and evidence $P(X)$. Then, the term $P(X|y)$ can be decomposed and simplified, using the chain rule and the conditional independence assumption, resulting in the expression shown in equation (16).

$$P(y|X) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(X)}. \tag{16}$$

In practice, there is interest only in the numerator of the fraction in (16), because the denominator $P(X)$ does not depend on $y$ and can be considered constant.

The Naïve Bayes classifier combines this probability model with a decision rule, in order to select the hypothesis that is most probable, which is known as the Maximum A Posterior (MAP) decision rule. The Bayes classifier applies the function that assigns a class label $c = C_k$, for the $k$ value that maximizes the expression shown in equation (17).

$$c = \text{argmax}_k p(C_k) \prod_{i=1}^{n} p(x_i|C_k), \tag{17}$$

with $k = 1, \ldots, K$

*3.3.4. A Multilayer Perceptron Neural Network (MPNN).* A multilayer perceptron neural network (MPNN) refers to a directed neural network formed by several consecutive levels

[61]. In an MPNN, during the training process, the input information is propagated from the input layer to the hidden unit layer, and finally reaches the output units to calculate the predicted value. An MPNN seeks to approximate an unknown function, denoted by $f^*$, such that $y = f^*(x)$, where $x$ is the input data and $y$ is the output value calculated by the network. In other words, an MPNN through an iterative process of parameter tuning (parameters denoted by $\theta$) optimizes a loss function to find a mapping $f$ such that $y = f(x;\theta)$, that is the function $f$ that minimizes the error associated with the loss function. In each unit (neuron), the MPNN performs the calculation indicated by equation (18).

$$y = \sigma(W \cdot x + b), \tag{18}$$

where $y$ corresponds to the output computed by the neuron, $x$ denotes the vector of input values, $W$ represents the vector of weights of input connections to the neuron, and $b$ corresponds to the bias. $\sigma(.)$ denotes the activation function used, usually a nonlinear function. Popular activation functions include the following:

(i) Sigmoid (or logistic): $\text{sigmoid}(x) = 1/1 + e^{-x}$

(ii) Hyperbolic tangent: $\tanh(x) = e^{x} - e^{-x}/e^{x} + e^{-x}$

(iii) Linear unit: $\text{Re}LU(x) = \max(x, 0)$

(iv) Leaky ReLU: $Leaky\text{Re}LU(x) = \max(\alpha x, x)$, being $\alpha$ a small constant, e.g., 0.1

In this experimental study, the rectified linear unit function (ReLU) is used as activation function.

*3.3.5. Support Vector Machines (SVM).* In support vector machines (SVM), a hyperplane that maximizes the margin between two classes in the training data is calculated to perform the classification process. The margin is defined as the minimum perpendicular distance between two points of each class to the separating hyperplane; this hyperplane is fitted during the learning process with the training data or predictors. From these predictors, the vectors that define the hyperplane are selected, which are called support vectors. The optimal hyperplane corresponds to the one that minimizes the training error and, at the same time, has the maximum margin of separation between the two classes. To generalize the cases where the decision limits are not linearly separable, Support Vector Machine projects the training data into another space of higher dimensionality; if the dimensionality of the new space is high enough, the data will always be linearly separable. To avoid having to carry out an explicit projection in a larger dimensional space, a kernel function is used, which implicitly transforms the data to this larger dimensional space to make the linear separation of the classes possible. The kernel function can be polynomial, Gaussian radial basis or sigmoidal perceptron, among others [62].

Formally, SVMs are based on the construction of a decision boundary, which takes the form of a hyperplane. In the case of input data that is not clearly linearly separable, kernel functions are used to transform the input data to a new multidimensional space, where a linear decision boundary can be constructed. In either case, the decision function for separating positive from negative classes takes the form of the equation of a hyperplane, as defined by (19) [63].

$$D(x) = w\phi(x) + b, \tag{19}$$

where $w$ and $b$ represent the parameters to be found to find the hyperplane that best separates positive from negative examples. Here, $\phi(x)$ represents the application of the kernel function to transform the original data represented by the vector $x$ into a new space of dimension $M$. Additionally, $D(x)/\|w\|$ represents the distance between the hyperplane and the data pattern $x$. Solving algebraically from (19), the values for the parameters $w$ and $b$ are obtained as indicated in the expressions defined in (20) and (21):

$$w = \sum_{k} \alpha_k y_k x_k, \tag{20}$$

$$b = (y_k - w * x_k). \tag{21}$$

The coefficients $\alpha_k$ are nonzero for the support vectors. It follows from these equations that the parameter $w$ is computed as a linear combination of the training data $x_k$, and the value $b$ is computed as an average of the nonzero $\alpha_k$.

*3.4. Classifier Performance Evaluation.* Usually, the confusion matrix is used to evaluate the performance of the classifiers, since it allows us to analyze and decompose the errors and successes for each value of the class attribute. Three fundamental metrics can be derived from a confusion matrix: precision (P), recall (R) and F-measure (F). These metrics are defined in terms of: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In particular, for the network traffic identification process, TP and TN corresponds the number of observations that correctly predict whether it is ransomware or benign application, respectively. On the other hand, FP and FN corresponds to the number of observations that are incorrectly predicted as ransomware or benign application, respectively.

(i) Precision (P) is defined as the ratio of all predicted samples as ransomware traffic that are actually ransomware, and it is computed as shown in equation (22).

$$P = \frac{TP}{TP + FP}. \tag{22}$$

(ii) Recall (R) is defined as the ratio of all ransomware traffic samples that are expected to be actually ransomware, and it is computed as shown in equation (23).

$$R = \frac{TP}{TP + FN}. \tag{23}$$

(iii) F-score ($F_1$): the $F_1$ value corresponds to the harmonic mean of precision and recall values, and therefore it may be better for evaluating performance than overall accuracy. It is computed for the expression shown in equation (24).

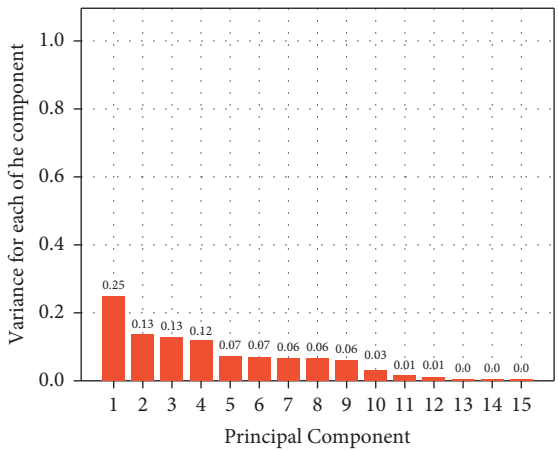


Figure 2: The variance explained by each component computed by the PCA method.

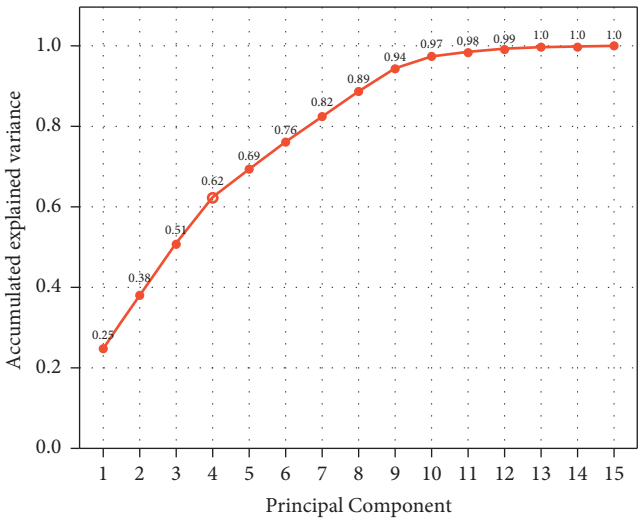


Figure 3: The accumulated explained variance associated to the PCA components.

$$F_1 = \frac{2 \times P \times R}{P + R}. \tag{24}$$

In addition, the area under the curve (AUC) evaluation metric was used for the receiver operating characteristics (ROC curve). The ROC curve evaluation metric is a graph that shows the performance of a ranking model across all ranking thresholds. A ROC curve represents true positives versus false positives at different classification thresholds [64]. The AUC value corresponds to the two-dimensional area under the entire ROC curve. Thus, the AUC metric provides an aggregated measure of performance at all possible classification thresholds [64], and is calculated as shown in equation (25).

$$AUC = \frac{1}{2}\left(\frac{TP}{TP + FP} + \frac{TN}{TN + FP}\right). \tag{25}$$

Table 4: Features that are most representative of the network traffic of malware and benign for the PCA method.

| NO. | Feature | Description |
|---|---|---|
| 1 | Flow duration | Duration of the flow in microsecond |
| 2 | Tot fwd pkts | Total packets in the outgoing |
| 3 | Tot bwd pkts | Total packets in the incoming |
| 4 | Fwd pkt len min | Minimum size of packet in outgoing |
| 5 | Fwd pkt len max | Maximum size of packet in outgoing |
| 6 | Bwd pkt len min | Minimum size of packet in incoming |
| 7 | Bwd pkt len max | Maximum size of packet in incoming |
| 8 | Tot len fwd pkts | Minimum length of a flow |
| 9 | Tot len bwd pkts | Maximum length of a flow |
| 10 | Label | Type of malware |

Table 5: Summary of the results obtained from the execution of the first experiment with $p$-value $\leq 0.05$.

| Feature | Coef | Odds ratio | $p$ value |
|---|---|---|---|
| Const | −0.4324 | 0.65282 | 0.001 |
| Flow_Duration | 0.00000 | 1.00000 | 0.001 |
| Tot_Fwd_Pkts | 0.00000 | 1.00000 | 0.001 |
| Tot_Bwd_Pkts | −0.07704 | 0.92585 | 0.001 |
| TotLen_Fwd_Pkts | 0.00016 | 1.00016 | 0.001 |
| TotLen_Bwd_Pkts | 0.00005 | 1.00005 | 0.001 |
| Fwd_Pkt_Len_Min | 0.00649 | 1.00652 | 0.001 |
| Bwd_Pkt_Len_Min | −0.00009 | 0.99991 | 0.192 |
| Fwd_Pkt_Len_Max | −0.00217 | 0.99784 | 0.001 |
| Bwd_Pkt_Len_Max | −0.00027 | −0.00027 | 0.001 |
| Fwd_Pkt_Len_Mean | −0.00798 | 0.99205 | 0.001 |
| Bwd_Pkt_Len_Mean | 0.00039 | 1.00039 | 0.001 |
| Fwd_Pkt_Len_Std | 0.01084 | 1.01090 | 0.001 |
| Bwd_Pkt_Len_Std | 0.00020 | 1.00020 | 0.100 |
| Mean len Pkts/s | 0.00000 | 1.00000 | 0.001 |
| Flow_Byts/s | 0.00000 | 1.00000 | 0.001 |

Table 6: Features that are most representative of the network traffic of malware and benign for logistic regression.

| NO. | Feature | Description |
|---|---|---|
| 1 | Flow duration | Duration of the flow in microsecond |
| 2 | Flow Byts/s | Number of flow bytes per second, |
| 3 | Tot fwd pkts | Total packets in the outgoing |
| 4 | Tot bwd pkts | Total packets in the incoming |
| 5 | Fwd pkt len min | Minimum size of packet in outgoing |
| 6 | Fwd pkt len max | Maximum size of packet in outgoing |
| 7 | Fwd pkt len mean | Mean size of packet in outgoing direction |
| 8 | Fwd pkt len Std | Standard deviation size of packets in outgoing |
| 9 | Bwd pkt len max | Maximum size of packet in incoming |
| 10 | Bwd pkt len mean | Mean size of packet in incoming direction |
| 11 | Tot len fwd pkts | Minimum length of a flow |
| 12 | Tot len bwd pkts | Maximum length of a flow |
| 13 | Mean len Pkts/s | Mean length of a flow |
| 14 | Label | Type of malware |

To obtain the values of the ROC curves in this work, the benign class was replaced with a value of 0 and the malware class with a value of 1 for the binary classification experiments. Likewise, to obtain the values of the ROC curves in

Table 7: Binary classification results without cross validation.

| Model | Precision | Recall | F1 score | Accuracy | AUC |
|---|---|---|---|---|---|
| CDI + DT | 0.94 | 0.94 | 0.94 | 94.03% | 0.94 |
| CDI + NB | 0.56 | 0.51 | 0.38 | 51.40% | 0.64 |
| CDI + RF | 0.95 | 0.95 | 0.95 | 95.30% | 0.97 |
| CDI + SVM | 0.86 | 0.82 | 0.82 | 82.26% | 0.86 |
| CDI + MPNN | 0.78 | 0.75 | 0.74 | 74.63% | 0.65 |
| CDI + KNN | 0.89 | 0.89 | 0.89 | 89.36% | 0.93 |
| CDPCA + DT | 0.91 | 0.91 | 0.91 | 90.90% | 0.93 |
| CDPCA + NB | 0.62 | 0.51 | 0.63 | 51.06% | 0.66 |
| CDPCA + RF | 0.94 | 0.94 | 0.94 | 93.76% | 0.96 |
| CDPCA + SVM | 0.86 | 0.82 | 0.82 | 82.26% | 0.86 |
| CDPCA + MPNN | 0.71 | 0.70 | 0.69 | 69.53% | 0.61 |
| CDPCA + KNN | 0.88 | 0.88 | 0.88 | 87.61% | 0.92 |
| CDLR + DT | 0.94 | 0.94 | 0.94 | 94,02% | 0.94 |
| CDLR + NB | 0.56 | 0.51 | 0.38 | 51.40% | 0.64 |
| CDLR + RF | 0.96 | 0.96 | 0,96 | 96.42% | 0.98 |
| CDLR + SVM | 0.86 | 0.82 | 0.82 | 82.26% | 0.86 |
| CDLR + MPNN | 0.71 | 0.70 | 0.70 | 70.20% | 0.61 |
| CDLR + KNN | 0.89 | 0.89 | 0.89 | 89.36% | 0.93 |

the multiclass classification experiments, the benign class was replaced by the value 0 ($C_0$), the adware class with the value 1 ($C_1$), the scareware class with the value 2 ($C_2$) and the ransomware class with the value 3 ($C_3$).

## 4. Experiments and Results

In this section, the experimental results in feature selection and performance evaluation are presented. Firstly, in selection of features, the results of the experiments carried out by the PCA and Logistic Regression methods are presented, to reduce and select the set of network features most representative of the behavior of malware and benign Android applications traffic (See Table 2).

Secondly, in performance evaluation, the experiments and results of the empirical evaluation of the performance of the following supervised algorithms are presented: Naïve Bayes, support vector machine, multilayer perceptron neural network, decision tree, random forest and K-nearest neighbors, with the purpose of identifying malware traffic from the features selected by the PCA and Logistic Regression statistical methods. All experiments were executed on Microsoft Windows 10 Professional (64 bit) with a second-generation Intel Core i7 2.20 Ghz processor and 16 GB of RAM. The Python 3.7.0 programming language was used to perform the data preprocessing tasks, the selection of features and the construction of the classification models. For the classifiers, the default parameters of Python scikit-learn were utilized.

*4.1. Experimental Results of the Selection of Features.* The experiment carried out with the PCA method calculated the proportion of explained variance for each computed component (See Figure 2.) and the accumulated explained variance (See Figure 3.) derived from the initial dataset (See Table 2). The 94% total variability is explained by the first 9 PCA components. The results of the PCA technique discarded 6 of 15 components related to the flow in bytes and

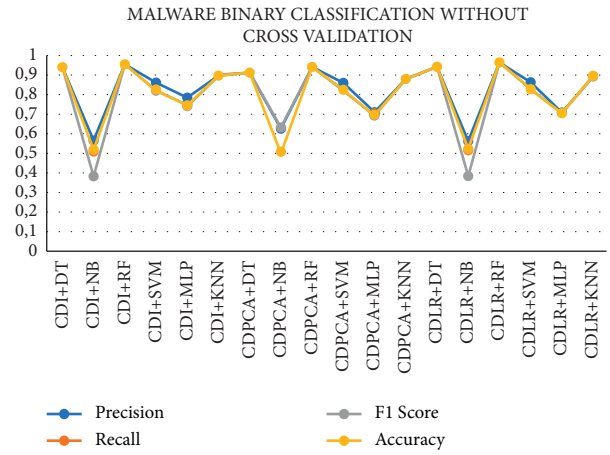


Figure 4: Experimental result of malware binary classification without cross-validation.

Table 8: Binary classification results using cross validation with $N = 10$.

| Model | Precision | Recall | F1 score | Accuracy (%) | AUC |
|---|---|---|---|---|---|
| CDI + DT | 0.94 | 0.94 | 0.94 | 94.05 | 0.94 |
| CDI + NB | 0.56 | 0.51 | 0.38 | 51.39 | 0.64 |
| CDI + RF | 0.95 | 0.95 | 0.95 | 95.29 | 0.97 |
| CDI + SVM | 0.87 | 0.87 | 0.87 | 86.86 | 0.86 |
| CDI + MPNN | 0.71 | 0.58 | 0.51 | 58.37 | 0.45 |
| CDI + KNN | 0.89 | 0.89 | 0.89 | 89.36 | 0.93 |
| CDPCA + DT | 0.94 | 0.94 | 0.94 | 93.98 | 0.94 |
| CDPCA + NB | 0.56 | 0.51 | 0.38 | 51.39 | 0.64 |
| CDPCA + RF | 0.94 | 0.94 | 0.94 | 93.76 | 0.96 |
| CDPCA + SVM | 0.87 | 0.87 | 0.87 | 86.86 | 0.86 |
| CDPCA + MPNN | 0.68 | 0.65 | 0.64 | 65.34 | 0.55 |
| CDPCA + KNN | 0.88 | 0.88 | 0.88 | 87.61 | 0.92 |
| CDLR + DT | 0.94 | 0.94 | 0.94 | 93.99 | 0.94 |
| CDLR + NB | 0.56 | 0.51 | 0.38 | 51.39 | 0.64 |
| CDLR + RF | 0.96 | 0.96 | 0.96 | 96.41 | 0.98 |
| CDLR + SVM | 0.86 | 0.82 | 0.82 | 82.26 | 0.86 |
| CDLR + MPNN | 0.79 | 0.78 | 0.77 | 77.50 | 0.79 |
| CDLR + KNN | 0.89 | 0.89 | 0.89 | 89.41 | 0.93 |

flow mean length packets per second (Flow Byts/s, Mean Len Pkts/s), the average size of input and output packets (Bwd Pkt Len Mean, Fwd Pkt Len Mean), and the standard input and output packet size (Bwd Pkt Len Std, Fwd Pkt Len Std). Therefore, the PCA method presents 9 features that are representative of the network traffic of malware and benign Android applications from the initial dataset (See Table 4).

Regarding the results obtained by the Logistic Regression method, Table 5 presents a summary of the results obtained from the execution of the first experiment with $p$-value of 0.05, where the minimum length features of input packets (Bwd_Pkt_Len_Min) and the standard deviation of the input packet length (Bwd_Pkt_Len_Std) presented a statistically nonsignificant association with respect to the class of the Logistic Regression model. For the second Logistic Regression experiment, the two variables with the lowest significance (Bwd_Pkt_Len_Min and Bwd_Pkt_Len_Std)

obtained from the first experiment with $p$-value $\leq 0.05$ were removed, and the significance value restriction was lowered to $p$-value $\leq 0.01$. The results of the second experiment did not find differences with respect to the model of features selected by the first experiment of $p$-value $\leq 0.05$. Table 6 presents the 13 most representative features of the network traffic of malware and benign Android applications selected by the Logistic Regression method.

*4.2. Experimental Results for the Performance Evaluation of Supervised Algorithms.* With the network traffic features already selected by the PCA and Logistic Regression methods, and considering the initial dataset, two experimental scenarios were defined to evaluate the performance of the six supervised algorithms for the task of classifying malware traffic. These are binary classification and multiclass classification. The binary classification scenario includes observations of network traffic with class-tagged malware and benign software. The multiclass classification scenario includes four types of classes: scareware, ransomware, adware, and benign software. For these two scenarios, the ratio of the training and testing set is 80 : 20. For both scenarios, experiments with and without N-fold cross-validation were performed. In N-fold cross-validation the dataset is randomly partitioned between N observations and the evaluations are executed by N iterations. In each iteration, N-1 sets of samples are selected for training and the other one is left to validate the precision of the classifier [65]. $N = 10$ was selected to carry out the experiments, according to the N-fold performance obtained in studies related to the detection and identification of malware ([28, 57, 66, 67]). Likewise, for both scenarios, the initial dataset (CDI), the dataset with features selected by PCA (CDPCA) and the dataset with features selected by Logistic Regression (CDLR) were considered. Tables 7–11, and 12 present the experimental results obtained through the combination of the initial features, the features selected by the PCA and LR

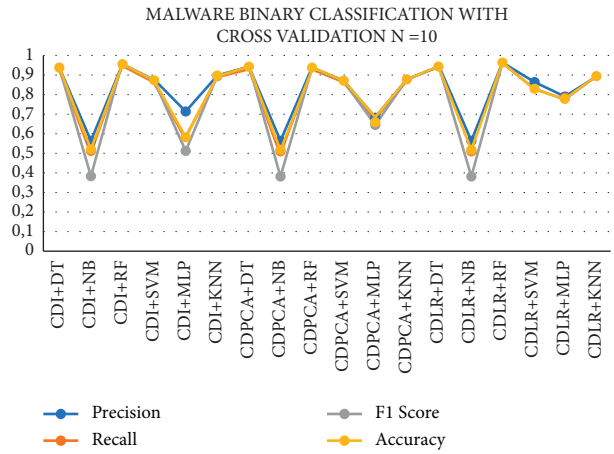


Figure 5: Experimental result of malware binary classification with cross-validation $N = 10$.

Table 9: Results for the multiclass classification scenario without cross validation.

| Model | Precision | Recall | F1 score | Accuracy (%) |
|---|---|---|---|---|
| CDI + DT | 0.86 | 0.86 | 0.85 | 85.7 |
| CDI + NB | 0.46 | 0.50 | 0.35 | 49.76 |
| CDI + RF | 0.86 | 0.86 | 0.86 | 86.93 |
| CDI + SVM | 0.78 | 0.78 | 0.77 | 77.53 |
| CDI + MPNN | 0.61 | 0.55 | 0.56 | 58.4 |
| CDI + KNN | 0.62 | 0.63 | 0.62 | 62.59 |
| CDPCA + DT | 0.85 | 0.86 | 0.85 | 85.67 |
| CDPCA + NB | 0.47 | 0.51 | 0.35 | 50.63 |
| CDPCA + RF | 0.86 | 0.86 | 0.86 | 85.63 |
| CDPCA + SVM | 0.78 | 0.78 | 0.77 | 77.53 |
| CDPCA + MPNN | 0.58 | 0.59 | 0.62 | 58.91 |
| CDPCA + KNN | 0.61 | 0.63 | 0.62 | 62.52 |
| CDLR + DT | 0.85 | 0.86 | 0.85 | 85.66 |
| CDLR + NB | 0.46 | 0.50 | 0.35 | 49.76 |
| CDLR + RF | 0.87 | 0.87 | 0.87 | 87.06 |
| CDLR + SVM | 0.78 | 0.78 | 0.77 | 77.53 |
| CDLR + MPNN | 0.55 | 0.59 | 0.54 | 59.45 |
| CDLR + KNN | 0.63 | 0.62 | 0.63 | 62.53 |

Table 10: The ROC curve final mixture of the performance results with multiclass without cross validation.

| Model | $C_0$ | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|---|
| CDI + DT | 0.97 | 0.96 | 0.94 | 0.96 |
| CDI + NB | 0.62 | 0.64 | 0.61 | 0.72 |
| CDI + RF | 0.97 | 0.98 | 0.95 | 0.97 |
| CDI + SVM | 0.55 | 0.77 | 0.56 | 0.67 |
| CDI + MPNN | 0.55 | 0.77 | 0.56 | 0.67 |
| CDI + KNN | 0.72 | 0.94 | 0.73 | 0.77 |
| CDPCA + DT | 0.97 | 0.98 | 0.95 | 0.97 |
| CDPCA + NB | 0.65 | 0.66 | 0.63 | 0.74 |
| CDPCA + RF | 0.96 | 0.97 | 0.94 | 0.97 |
| CDPCA + SVM | 0.55 | 0.77 | 0.56 | 0.67 |
| CDPCA + MPNN | 0.72 | 0.82 | 0.70 | 0.72 |
| CDPCA + KNN | 0.73 | 0.93 | 0.70 | 0.79 |
| CDLR + DT | 0.97 | 0.96 | 0.94 | 0.96 |
| CDLR + NB | 0.62 | 0.64 | 0.61 | 0.72 |
| CDLR + RF | 0.97 | 0.98 | 0.95 | 0.97 |
| CDLR + SVM | 0.55 | 0.77 | 0.56 | 0.67 |
| CDLR + MPNN | 0.97 | 0.98 | 0.95 | 0.97 |
| CDLR + KNN | 0.72 | 0.94 | 0.73 | 0.77 |

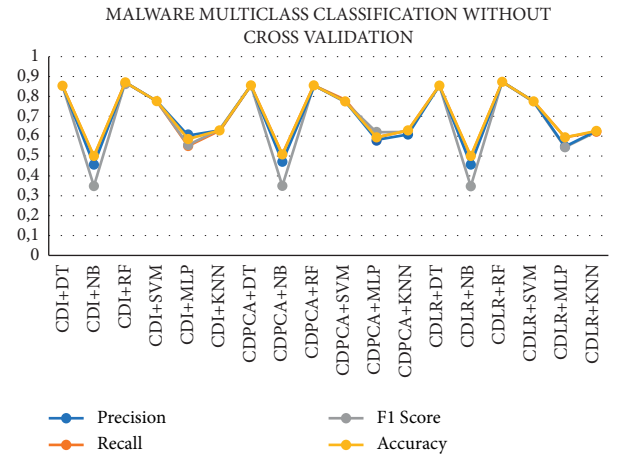


Figure 6: Experimental result of malware multiclass classification without cross-validation.

Table 11: Results for the multiclass classification with cross validation with $N = 10$.

| Model | Precision | Recall | F1 score | Accuracy (%) |
|---|---|---|---|---|
| CDI + DT | 0.86 | 0.86 | 0.85 | 85.7 |
| CDI + NB | 0.42 | 0.50 | 0.35 | 49.60 |
| CDI + RF | 0.87 | 0.87 | 0.87 | 86.92 |
| CDI + SVM | 0.78 | 0.78 | 0.77 | 77.53 |
| CDI + MPNN | 0.52 | 0.57 | 0.48 | 56.57 |
| CDI + KNN | 0.71 | 0.71 | 0.71 | 71.46 |
| CDPCA + DT | 0.85 | 0.85 | 0.85 | 85.47 |
| CDPCA + NB | 0.47 | 0.51 | 0.35 | 50.54 |
| CDPCA + RF | 0.86 | 0.86 | 0.86 | 85.63 |
| CDPCA + SVM | 0.78 | 0.78 | 0.77 | 77.53 |
| CDPCA + MPNN | 0.55 | 0.59 | 0.53 | 58.73 |
| CDPCA + KNN | 0.70 | 0.71 | 0.70 | 71.15 |
| CDLR + DT | 0.85 | 0.86 | 0.85 | 85.50 |
| CDLR + NB | 0.42 | 0.50 | 0.35 | 49.60 |
| CDLR + RF | 0.87 | 0.87 | 0.87 | 87.05 |
| CDLR + SVM | 0.78 | 0.78 | 0.77 | 77.53 |
| CDLR + MPNN | 0.56 | 0.59 | 0.51 | 58.57 |
| CDLR + KNN | 0.71 | 0.71 | 0.71 | 71.43 |

Table 12: The ROC curve final mixture of the performance results with multiclass cross validation with $N = 10$.

| Model | $C_0$ | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|---|
| CDI + DT | 0.97 | 0.96 | 0.94 | 0.96 |
| CDI + NB | 0.64 | 0.67 | 0.63 | 0.70 |
| CDI + RF | 0.97 | 0.98 | 0.95 | 0.97 |
| CDI + SVM | 0.64 | 0.67 | 0.63 | 0.70 |
| CDI + MPNN | 0.55 | 0.77 | 0.56 | 0.67 |
| CDI + KNN | 0.72 | 0.94 | 0.73 | 0.77 |
| CDPCA + DT | 0.97 | 0.98 | 0.95 | 0.97 |
| CDPCA + NB | 0.65 | 0.70 | 0.65 | 0.71 |
| CDPCA + RF | 0.96 | 0.97 | 0.94 | 0.97 |
| CDPCA + SVM | 0.55 | 0.77 | 0.56 | 0.67 |
| CDPCA + MPNN | 0.72 | 0.82 | 0.70 | 0.72 |
| CDPCA + KNN | 0.73 | 0.93 | 0.70 | 0.79 |
| CDLR + DT | 0.97 | 0.96 | 0.94 | 0.96 |
| CDLR + NB | 0.64 | 0.67 | 0.63 | 0.70 |
| CDLR + RF | 0.97 | 0.98 | 0.95 | 0.97 |
| CDLR + SVM | 0.55 | 0.77 | 0.56 | 0.67 |
| CDLR + MPNN | 0.97 | 0.98 | 0.95 | 0.97 |
| CDLR + KNN | 0.72 | 0.94 | 0.73 | 0.77 |

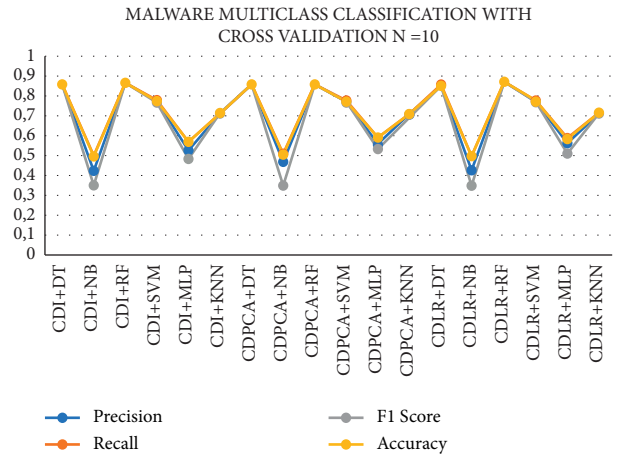


Figure 7: Experimental result of malware Multiclass classification with cross-validation $N = 10$.

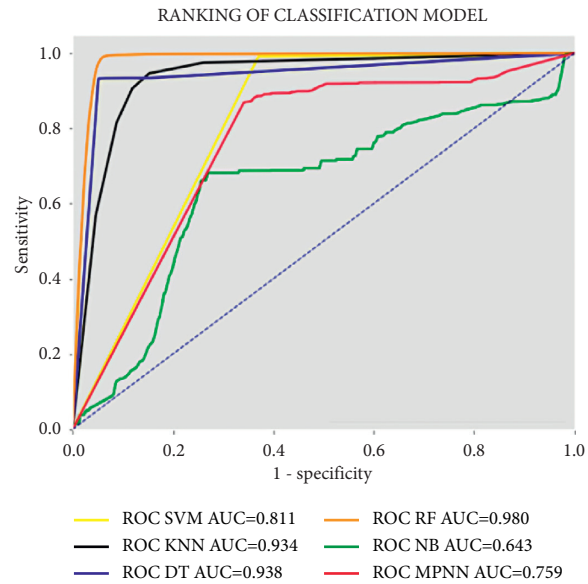


Figure 8: Performance results of each of the classifiers combined with the 13 features obtained by the logistic regression method.

methods, together with the application of the six supervised algorithms mentioned (DT, NB, RF, SVM, MPNN, and KNN).

In the binary classification experiment without Cross-validation (Table 7) it was found that, using the combination of features of logistic regression and random forest (CDLR + RF), the best performance was obtained, with an average precision rate of 0.96, recall of the 0.96, F1 value of 0.96, accuracy of 96.42% and AUC of 0.98 with respect to the rest of the experiments (See Figure 4). Likewise, for the binary classification with Cross-validation $N = 10$ (See Tables 8 and Figure 5), the combination of logistic regression and random forest features (CDLR + RF) obtained a slightly lower accuracy performance of 96.41%, a precision of 0.96, a recall of 0.96 and F1 value of 0.96. Among both experiments of binary classification with initial features, selected by PCA and logistic regression, Naïve Bayes obtained the worst performance (see Figures 4 and 5). For the multiclass classification scenario without cross-validation (see Tables 9, 10 and Figure 6) it was found that, for the combination of the features selected by the Logistic Regression method and the Random Forest algorithm (CDLR + RF), the best performance was obtained. In this case, an average precision of 0.87 was obtained, recall of 0.87, F1 value of 0.87, accuracy of 87.06% and an AUC average rate greater than 85% for the malware classification. Likewise, for the multiclass classification with Cross-validation $N = 10$ (See Tables 11, 12 and Figure 7), the combination of Logistic Regression and Random Forest features (CDLR + RF) obtained a slightly lower accuracy performance with 87.05%, equating the average precision with 0.87, recall of 0.87 and F1 value of 0.87. Among both multiclass classification experiments with

Table 13: Result of performance evaluation and comparison metrics in binary classification.

| Results works | Comparison metrics | | | | |
|---|---|---|---|---|---|
| | Precision | Recall | F1 score | Accuracy | AUC |
| Lashkari et al. [28] | 0.85 | 0.85 | 0.85 | 85% | N/A |
| Abuthawabeh et al. [11] | 0.86 | 0.86 | 0.86 | 86% | N/A |
| Murtaz et al. [48] | N/A | N/A | N/A | N/A | N/A |
| Abuthawabeh et al. [49] | N/A | N/A | N/A | 87% | N/A |
| Chen et al. [42] | 0.95 | 0.95 | 0.95 | 95% | N/A |
| Proposed method | 0.96 | 0.96 | 0.96 | 96% | 0.98 |

Table 14: Result of performance evaluation and comparison metrics in multiclass classification.

| Results works | Comparison metrics | | | | |
|---|---|---|---|---|---|
| | Precision | Recall | F1 score | Accuracy (%) | AUC |
| Lashkari et al. [28] | 0.49 | 0.49 | 0.49 | 49 | N/A |
| Abuthawabeh et al. [11] | 0.79 | 0.79 | 0.79 | 79 | N/A |
| Murtaz et al. [48] | N/A | N/A | N/A | 94 | N/A |
| Abuthawabeh et al. [49] | 0.89 | 0.89 | 0.89 | 89 | N/A |
| Chen et al. [42] | 0.84 | 0.84 | 0.84 | 84 | N/A |
| Proposed method | 0.87 | 0.87 | 0.87 | 87 | 0.98 |

initial features, selected by PCA and logistic regression, Naïve Bayes obtained the worst performance (See Figures 6 and 7). The ROC curve (see Tables 10, 12 and Figure 8) summarizes the final mixture of the performance results of each of the classifiers combined with the features obtained by the logistic regression method. The ROC curve shows a good initial discriminative ability, losing it smoothly at the rate of 90% true positive (TP). Random forest presented the highest AUC for the 13 most representative features selected from Logistic Regression (CDLR + RF).

## 5. Discussion

Some of the features selected by Logistic Regression with $p$-value were expected according to the state of the art. The duration of flows in microseconds, the number of byte flows per second, the total of packets in the outbound direction and the total of packets in the inbound direction, are well known to researchers in malware detection and identification in network traffic. Logistic Regression presented features like those recorded in previous studies in the related work section and the authors' prior knowledge. The network flow time variables Flow Byts/s and flow Mean Len Pkts/s discarded by PCA are associated with a higher probability of identifying malware and differentiating it from benign applications. The features selected by Logistic Regression obtained good results, especially when the Random Forest and Decision Tree algorithms were used in binary and multiclass classification. In [28] a good performance of Random Forest (RF), Decision Tree (DT) and K-Nearest Neighbor (KNN) was also achieved with an average precision of 85% and recall of 88% in binary classification. However, RF, DT and KNN presented an average precision and recall of less than 49% in multiclass classification. In [28], 9 features and the CfsSubsetEval, Best First and Infogain methods were used. The selection of features through Logistic Regression allowed to obtain representative features for the identification of malware, avoiding the shadowing of features generated by the PCA method. As presented in Tables 9–11 and 12, the results obtained by the proposed method are optimistic towards the selection of new network traffic features. Likewise, Random Forest achieved promising results in binary and competitive classification compared to the results presented by other works in multiclass classification Tables 13 and 14. This is especially true when there are a greater number of features related to network flow times.

## 6. Related Work

In this section, related works are discussed and its analysis is logically divided into two parts. Firstly, we review related works about classification task to detect and identify android network malware in general. Secondly, we discuss related works about classification task to detect and identify android network malware that use specifically the CICAndMal2017 dataset.

*6.1. Classification of Android Network Malware.* The selection of features is key in most cases of classification and regression tasks and impacts the performance of machine learning algorithms, in particular this is valid in the detection and identification of malware-type network traffic. First of all, the task of identifying malicious traffic (binary classification) is important, for which selecting the relevant characteristics and discarding the redundant ones have a significant impact on the construction and application of machine learning models. Secondly, it is also very important to identify the type of malware (multiclass classification) that is present in traffic identified as malicious, in order to better understand and resolve the potential attack [68].

Mainly, it is possible to distinguish between machine learning applications that detect versus those that identify malware traffic based on the type of output of the

Table 15: Total number of benign software and malware network traffic conversations for each related job.

| Works | N°TrafficConversation | Method | Feature selected |
|---|---|---|---|
| Lashkari et al. [28] | 5.494 | CfsSubsetEval<br>Best first<br>Infogain | 9 |
| Abuthawabeh et al. [11] | 244.594 | Random forest<br>Recursive feature<br>Elimination, light GBM | 14 |
| Murtaz et al. [48] | 126.391 | Data gain<br>Cfs subset<br>SVM weka | 9 |
| Abuthawabeh et al. [49] | 305.743 | Random forest<br>Recursive feature<br>Elimination, light GBM | 9 |
| Chen et al. [42] | 244.802 | Python method | 15 |

classification problem. A malware detection system generates as output a binary value or a value in the range between 0 and 1, $y = f(x)$, that is, an output value $y$ before an input vector $x$. In the case of a malware identification system, the output is associated with a probability of belonging to a class or family of malicious traffic, being $y \in \mathbb{R}^N$, where $N$ is the number of different families [23].

Among the different amount of studies facing malware detection and identification in networks, there are some works that deal with feature selection prior to testing detection and classification techniques.

In [69], the analysis of computer network traffic based on a system that detects the presence of malware is exposed. A total of 972 behavioral characteristics of network traffic over the Internet, at the transport and application level, were extracted and analyzed. The selection of the subset of features was based on the correlation feature selection algorithm, which fed three classification algorithms: random forest, Naive Bayes and decision tree (specifically the J48 algorithm).

In [70], 9 traffic features were selected to improve the efficiency of a network traffic classifier, which implements a mobile malware traffic detector. This subset of features was selected using CfsSubsetEval attribute evaluator and the Best First search method. In order to characterize malware families, the proposed model uses features including flow-based, packet-based and time-based features. The results obtained by using the proposed feature set reach an accuracy above 93% in the detection of malware. In addition, a 92% of success probability on characterization and on average a false positive rate less than 0.08 percent are obtained. These performance values are required in the implementation of a system operating on real world malware detection.

In [71] the authors proposed a dynamic analysis technique in Android Malware detection.

Initially, the data obtained is related to memory and CPU usage, packet transfer and system calls, which were considered as input to the feature extraction task. Second, the Gain Ratio Attribute Evaluator algorithm was used to select features. Third, the APKPure and Genome Project datasets were used to perform the classifier training and validation process to discriminate between malicious and benign traffic. The results obtained indicate that, using the Random Forest algorithm, 91.7%, 93.1% and 90% of global accuracy, precision and recall, respectively, are obtained.

In Hernandez and Goseva -Popstojanova [72], the authors focused on malware detection based on the use of characteristics extracted from network traffic and system logs. features were used. Experimental work was carried out based on four algorithms (Naive Bayes, J48, Random Forest and PART) for the malware detection task. In determining the least number of characteristics, information gain was used as a metric to rank the attributes. Based on the F1 score and G-score metrics, the classifiers with the best performance turned out to be those obtained with the J48 and PART algorithms. Remarkably, the J48 algorithm obtained a similar performance using only the 5 best features than in the case of using all 88 original features, which translates into a decrease in computational cost during training. In the case of the PART algorithm, similar results were obtained when using the 14 best features versus using all 88 original features.

Wang et al. [73] proposed an efficient malware detection method using the text semantics of HTTP network traffic with NPL (natural language processing), chi-square algorithm to automatically select the best features, and an SVM machine learning linear classifier. In the evaluation, 31,706 benign streams and 5,258 malicious streams were used, and the proposed classifier outperforms existing approaches and obtains an accuracy of 99.15%.

Shabtai et al. [74] contributed a system that detects malicious behavior through network traffic analysis. This is done by logging user-specific network traffic patterns per examined app and subsequently identifying deviations that can be flagged as malicious. To evaluate their model, the C4.5 algorithm is employed, achieving an accuracy of up to 94%.

*6.2. Classification of Android Network Malware Using the CICAndMal2017 Dataset.* Some selected related works [11, 28, 42, 48, 49] associated with the detection and identification of malware Android in network traffic based on classification task of machine learning, which consider

the selection of features in the CICAndMal2017 dataset, are presented in Table 15. This table shows the total number of benign software and malware network traffic conversations, the feature selection method, and the number of features selected for the classification process for each related job. For this work, the number of 15 network features obtained from Chen et al. [42] and the set of benign software and malware traffic conversations acquired from the 2,216 CSV files in the work of Lashkari et al. [28], were used.

In 2018 Lashkari et al. [28] present a systematic approach to generate real Android mobile traffic using CICAndMal2017. Also [28] proposed an experimental strategy of binary and multiclass classification, together with three classifiers, carrying out the training and evaluation of their performance based on the CICAndMal2017 dataset. The results of [28] for binary classification show an average precision of 85% and recall of 88% for the random forest (RF), K-nearest neighbor (KNN) and decision tree (DT) algorithms. However, random forest (RF), K-nearest neighbor (KNN) and decision tree (DT) presented an average precision and recall of less than 49% in multiclass classification.

Abuthawabeh et al. [11], present an improved model for the detection, categorization, and classification of malware families in network traffic using CICAndMal2017. The authors use the enhanced PeerShark tool for 14 feature extraction and an assembly with three feature selection algorithms to achieve choosing the 9 most representative features from the dataset. The feature selection algorithms are RF, Recursive Feature Elimination (RFE), and Light GBM. The model developed in [11] was trained and evaluated using three classifiers: RF, KNN and DT. The study by Abuthawabeh et al. [11], compared the results of the improved model with the model of Lashkari et al. [28], through precision and recall metrics, obtaining slightly better results in binary detection and a significant improvement in multiclass classification, over an average greater than 79% in precision and recall.

In [42] the malware identification work based on Android network traffic analysis in the CICAndMal2017 dataset is presented. The authors selected a PCAP file from each family of benign malware and software to build the customized dataset. The chosen conversations were taken at random. Features were extracted from PCAP files by two steps. The first step was developed using a Java program to separate the network flows. Then 15 features were extracted, using a Python program. Three supervised machine learning classifiers were used: RF, KNN, and DT. In [42] a binary (malware and benign) and multiclass experimental strategy was used with three categories: Adware, Ransomware and Scareware. The authors use three metrics to evaluate the performance of the classifiers: precision, recall and measure F. For the binary classification of malware, the results show that the random forest classifier achieved the highest results with 92% of the measure F and a 95% accuracy and recall. The rest of the classifiers obtained more than 85% of all the metrics used. For the multiclass classification of malware, the RF, KNN and DT classifiers achieved an average of more than 80% in each of the metrics selected by the authors. As in binary classification, Random Forest achieved the highest results in multiclass classification with an average of 84% precision, recall and measure F.

In [48], a framework for the detection and classification of Android malware is proposed in the CICandMal2017 dataset. An experimental multiclass classification strategy was proposed with network traffic from benign applications, adware, and general malware. Weka's Data Gain, CFSSubset and Support Vector Machine (SVM) feature selection algorithms were used. The CFSSubset algorithm selected the 9 most significant features for the framework presented by [48]. The results presented indicate that, for the random forest (RF), K-nearest neighbor (KNN), decision tree (DT), random tree (RT) and LOGISTIC REGRESSION (LR) classifiers, an accuracy of 94% was obtained. The authors do not show precision and recall results.

In [49], the authors propose a model to detect and categorize malware based on network traffic features considering the CICAndMal2017 dataset. The 9 most significant features were chosen using the assembly technique through three feature selection algorithms: random forest, recursive feature elimination (RFE) and light GBM. Likewise, the model was evaluated with three classifiers: random forest (RF), decision tree (DT) and extra tree (ET). The experimental results show that the selected features improved the detection and categorization of Android malware. The extra tree (ET) algorithm obtained the best accuracy with 87.75%, precision of 89.35% and a recall of 85.33% for binary classification. For multiclass classification, extra tree also obtained the best performance with 79.7% accuracy, 80.24% precision and 79.3% recall. Using the same dataset (CICAndMal2017) in [49], Manzano et al. in [75] study the classification between benign applications and ransomware using only three classificators (RF, DT, and KNN), without a focus on the feature selection. In [75], the authors conclude that the selection of features can help differentiate ransomware from the traffic of benign applications.

In summary, the results obtained by these experimental works will provide a baseline of comparison for binary and multiclass classification for the network traffic data considered. In terms of the feature selection method, this work performs an exhaustive exploration based on two feature reduction methods: principal component analysis (PCA) and logistic regression (LR). Both are combined with six traditional learning machine learning algorithms (RF, KNN, DT, NB, MLP, and SVM), building subsets of 10 and 13 features for PCA and LR, respectively. In terms of performance of the models generated by these algorithms, the proposed method exhibits better results than the related works [11, 28, 42, 48, 49] reviewed, in binary classification, and superior to the same works in terms of precision and recall for multiclass classification. The exception is that reported in [48], where only the global multiclass accuracy is presented, without reporting other relevant metrics such as precision, recall and the F measure, both for the case of binary and multiclass classification.

## 7. Conclusion and Future Work

This work presents the results of an empirical evaluation of the performance of six supervised algorithms: Naïve Bayes, support vector machine, multilayer perceptron neural

network, decision tree and K-nearest neighbors, to identify malware traffic, considering two statistical methods of reduction and selection of features of the Android network traffic dataset CICAndMal2017. First, the PCA and Logistic Regression feature selection methods were run, extracting the most representative features for the identification of malware and benign Android applications. For the PCA method, the first 9 components obtained 94% of the variance of the data, being the most representative candidate features between network input and output packets. However, the PCA experiments evidenced a shading of network flow time-type features, which were evidenced as a significant contribution to the Logistic Regression method. This is the case of the network flow time variables Flow Byts/s and Mean Len Pkts/s discarded by the PCA. PCA-based feature selection performed the worst on the accuracy metrics for binary and multiclass classification. The logistic regression algorithm was able to detect a more accurate and useful feature correlation weight for binary and multiclass classification experiments. Logistic regression provided the features that contributed to obtaining the best binary and multi-classification results, with and without cross-validation, using the random forest algorithm. Although PCA managed to reduce the initial set of features to have better performance of the algorithms used, Logistic Regression managed to return with its $p$-value score <0.05 those network flow time variables, to improve the general precision of the models, both binary and multiclass.

The experimental classification results show that the network traffic classification technique based on random forest obtained the best identification of malware traffic and benign traffic with an average accuracy of 96% and an AUC of 0.98, over the rest of the binary classification algorithms. In addition, random forest had the best average malware accuracy performance at 87% and AUC average over all other multiclass classification algorithms. The lowest malware classification result around the binary and multiclass classification scenarios was the Naïve Bayes algorithm. Future work will consider improving the rate of identification of malware and benign applications through experiments based on cross-validation with different N-fold settings. Class balancing methods will be addressed to achieve a more efficient work on malware classification.

Finally, as the evolution of Android malware attacks is rapid and permanent, the features of our dataset may not be practical for detecting new malware cases. Therefore, applying deep learning methods can be good alternatives for the detection and identification of the traffic of new cases of malware, since these approaches do not depend on predefined features, but are built internally, as part of the complex and hierarchical process of deep learning.

## Data Availability

The dataset, CICAndMal2017, and scripts used for this study are available on https://github.com/cmanzanomm/ManzanoEtal_paper2_2021.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] G. Ramesh and A. Menen, "Automated dynamic approach for detecting ransomware using finite-state machine," *Decision Support Systems*, vol. 138, Article ID 113400, 2020.

[2] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture*, March, vol. 112, , Article ID 101861, 2021.

[3] M. Odusami, O. Abayomi-Alli, S. Misra, and O. Shobayo, "Android malware detection: a survey," *Communications in Computer and Information Science*, vol. 942, pp. 255–266, 2018.

[4] McAfee Labs, "McAfee Labs COVID-19 Threats Report scale and impact cyber-related attacks have," 2020, https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-july-2020.pdf.

[5] E. C. Bayazit, O. K. Sahingoz, and B. Dogan, "Malware detection in android systems with traditional machine learning models: a survey," in *Proceedings of the 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, June 2020.

[6] K. Tam, F. Ali, N. B. Anuar, R. Salleh, and L. Cavallaro, "The evolution of android malware and android analysis techniques," *ACM Computing Surveys*, vol. 49, no. 4, 2017.

[7] M. Scalas, D. Maiorca, F. Mercaldo, C. Aaron Visaggio, F. Martinelli, and G. Giacinto, "On the effectiveness of system API-related information for Android ransomware detection," *Computers & Security*, vol. 86, pp. 168–182, 2019.

[8] H. Zhang, Xi Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, "Classification of ransomware families with machine learning based on N-gram of opcodes," *Future Generation Computer Systems*, vol. 90, pp. 211–221, 2019.

[9] S. Jan, T. Pevný, and R. Martin, "Probabilistic analysis of dynamic malware traces," *Computers & Security*, vol. 74, pp. 221–239, 2018.

[10] O. M. K Alhawi, J. Baldwin, and D. Ali, "Leveraging machine learning techniques for Windows ransomware network traffic detection," *Advances in Information Security,Cyber Threat Intelligence*, Springer International Publishing, vol. 70, , pp. 93–106, 2018.

[11] M. Abuthawabeh and K. Mahmoud, "Enhanced android malware detection and family classification, using conversation-level network traffic features," *The International Arab Journal of Information Technology*, vol. 17, no. 4, pp. 607–614, 2020.

[12] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[13] E. Biersack, F. Measurement, and D. Hutchison, "Lncs 7754 - data traffic monitoring and analysis," *Springer Berlin Heidelberg*, vol. 7754, pp. 2–27, 2013.

[14] Y. Elovici, A. Shabtai, R. Moskovitch, T. Gil, and C. Glezer, "Applying machine learning techniques for detection of malicious code in network traffic," in *Lecture Notes in Computer Science*vol. 4667, , pp. 44–50, Springer-Verlag, 2007.

[15] F. Ali, A. Nor Badrul, and R. Salleh, "Evaluation of network traffic analysis using fuzzy C-means clustering algorithm in mobile malware detection," *Advanced Science Letters*, vol. 24, no. 2, 2018.

[16] A. Zulkifli and I. Rahmi, "Hamid, Wahidah Md Shah and Zubaile Abdullah. "Android malware detection based on network traffic using decision tree algorithm"," *Advances in Intelligent Systems and Computing*, vol. 700, pp. 485–494, 2018.

[17] M. L. Abbas and A. R. Ajiboye, "The effects of dimensionality reduction in the classification of network traffic datasets via clustering," *Journal of Applied Sciences*, vol. 1, no. 1, 2020.

[18] F. Ali, A. Nor Badrul, R. Salleh, and A. W. Abdul Wahab, "A review on feature selection in mobile malware detection," *Digital Investigation*, vol. 13, pp. 22–37, 2015.

[19] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[20] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: evaluation and comparison," *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, 2010.

[21] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of android malware detection approaches based on machine learning," *IEEE Access*, vol. 8, pp. 124579–124607, 2020.

[22] M. C. Prakash, L. Liu, S. Saha, P.-N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," in *Proceedings of the 2013 IEEE INFOCOM*, pp. 2022–2030, IEEE, Turin, Italy, April 2013.

[23] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, Article ID 102526, 2020.

[24] L. Onwuzurike, E. Mariconti, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "Mamadroid: detecting android malware by building Markov chains of behavioral models (extended version)," *ACM Transactions on Privacy and Security*, vol. 22, no. 2, 2019.

[25] P. O'kane, S. Sezer, and K. McLaughlin, "Detecting obfuscated malware using reduced opcode set and optimised runtime trace," *Security Informatics*, vol. 5, no. 1, 2016.

[26] R. A. Shah, Y. Qian, D. Kumar, M. Ali, and M. B. Alvi, "Network intrusion detection through discriminative feature selection by using sparse logistic regression," *Future Internet*, vol. 9, no. 4, pp. 1–15, 2017.

[27] X. Han, F. Jin, R. Wang, S. Wang, and Ye Yuan, "Classification of malware for self-driving systems," *Neurocomputing*, vol. 428, pp. 352–360, 2021.

[28] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," *Proceedings - International Carnahan Conference on Security Technology*, vol. 50, 2018.

[29] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "MaMaDroid: detecting android malware by building Markov chains of behavioral models," in *Proceedings of the 2017 Network and Distributed System Security Symposium*, pp. 7129–7131, Internet Society, Reston, VA, February 2017.

[30] L. Wen and H. Yu, "An Android malware detection system based on machine learning," *AIP Conference Proceedings*, vol. 1864, 2017.

[31] X. Han and B. Olivier, "Interpretable and adversarially-resistant behavioral malware signatures," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, March 2020.

[32] T. F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," *Lecture Notes in Computer Science*, vol. 5137, pp. 207–227, 2008.

[33] S. Huda, J. Abawajy, M. Abdollahian, R. Islam, and J. Yearwood, "A fast malware feature selection approach using a hybrid of multi-linear and stepwise binary logistic regression," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 23, pp. 1–18, 2017.

[34] M. Alauthaman, N. Aslam, Li Zhang, R. Alasem, and M. A. Hossain, "A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks," *Neural Computing & Applications*, vol. 29, no. 11, pp. 991–1004, 2018.

[35] I. T. Jollife and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences*, vol. 374, no. 2065, 2016.

[36] I. T Jolliffe, *Principal Component Analysis*, Springer, Berlin/Heidelberg, Germany, 2002.

[37] S. Menard, "Applied logistic regression analysis," *Quantitative Applications in the Social Sciences*, vol. 106, 2002.

[38] S. Chatterjee and S. H. Ali, *Regression Analysis by Example*, John Wiley & Sons, Hoboken, New Jersey, USA, 2013.

[39] Z. Kain and J. MacLaren, "Valor de p inferior a 0'005: ¿qué significa en realidad?" *Pediatrics*, vol. 63, no. 3, pp. 118–120, 2007.

[40] J. C. Ferreira and C. M. Patino, "What does the p value really mean?" *Jornal Brasileiro de Pneumologia*, vol. 41, no. 5, p. 485, 2015.

[41] J. T. Pohlmann and D. W. Leitner, "A comparison of ordinary least squares and logistic regression," *Ohio Journal of Science*, vol. 103, no. 5, pp. 118–125, 2003.

[42] R. Chen, Y. Li, and W. Fang, "Android malware identification based on traffic analysis," *Lecture Notes in Computer Science*, vol. 11632, pp. 293–303, 2019.

[43] Q. Liu and Z. Liu, "A comparison of improving multi-class imbalance for internet traffic classification," *Information Systems Frontiers*, vol. 16, no. 3, pp. 509–521, 2014.

[44] Z. Liu, R. Wang, M. Tao, and X. Cai, "A class-oriented feature selection approach for multi-class imbalanced network traffic datasets based on local and global metrics fusion," *Neurocomputing*, vol. 168, pp. 365–381, 2015.

[45] R. Panigrahi, S. Borah, A. Kumar Bhoi et al., "A consolidated decision tree-based intrusion detection system for binary and multiclass imbalanced datasets," *Mathematics*, vol. 9, no. 7, p. 751, 2021.

[46] Y. Bai, Z. Xing, D. Ma, X. Li, and Z. Feng, "Comparative analysis of feature representations and machine learning methods in android family classification," *Computer Networks*, vol. 184, Article ID 107639, 2021.

[47] B. Chidlovskii and L. Lecerf, "Scalable feature selection for multi-class problems," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 227–240, Springer, Antwerp, Belgium, 2008 September.

[48] M. Murtaz, A. Hassan, A. Syed Baqir, and S. Rehman, "A framework for android malware detection and classification," in *Proceedings of the 2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pp. 1–5, IEEE, Bangkok, Thailand, November 2018.

[49] M. K. A. Abuthawabeh and K. W. Mahmoud, "Android malware detection and categorization based on conversation-level network traffic features," in *Proceedings of the 2019 International Arab Conference on Information Technology (ACIT)*, pp. 42–47, IEEE, Al Ain, United Arab Emirates, December 2019.

[50] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[51] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Computer Networks*, vol. 174, 2020.

[52] V. Y. Kulkarni, M. Petare, and P. K. Sinha, "Analyzing random forest classifier with different split measures," *Advances in Intelligent Systems and Computing*, Springer, in *Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012)*, pp. 691–699, December 2012.

[53] E. Fix and J. L. Hodges, . *Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties*, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[54] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[55] P. A. Jaskowiak and R. J. G. B. Campello, "Comparing correlation coefficients as dissimilarity measures for cancer classification in gene expression data," *VI Brazilian Symposium on Bioinformatics (BSB2011)*, vol. 1, 2011.

[56] D. Sharma, "Android malware detection using decision trees and network traffic," *International Journal of Computer Science and Information Technologies*, vol. 7, no. 4, pp. 1970–1974, 2016.

[57] L. Čehovin and Z. Bosnić, "Empirical evaluation of feature selection methods in classification," *Intelligent Data Analysis*, vol. 14, no. 3, pp. 265–281, 2010.

[58] M. B. Al Snousy, H. Mohamed El-Deeb, K. Badran, and I. A. Al Khlil, "Suite of decision tree-based classification algorithms on cancer gene expression data," *Egyptian Informatics Journal*, vol. 12, no. 2, pp. 73–82, 2011.

[59] S. Tufféry, *Data Mining and Statistics for Decision Making*, John Wiley & Sons, Hoboken, New Jersey, USA, 2011.

[60] Om P. Samantray and S. N. Tripathy, "A knowledge-domain analyser for malware classification," in *Proceedings of the 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pp. 1–7, IEEE, Gunupur, India, March 2020.

[61] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54024–54033, 2019.

[62] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques: Concepts and Techniques*, Elsevier, Amsterdam, Netherlands, 3rd Edition, 2012.

[63] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[64] Z. Chen, Q. Yan, H. Han et al., "Machine learning based mobile malware detection using highly imbalanced network traffic," *Information Sciences*, vol. 433-434, pp. 346–364, 2018.

[65] A. C. Tan and D. Gilbert, "An empirical comparison of supervised machine learning techniques in bioinformatics,"vol. 19, pp. 219–222, in *Proceedings of the First Asia-Pacific Bioinformatics Conference on Bioinformatics 2003*, vol. 19, pp. 219–222, Australian Computer Society, Sydney NSW, April 2003.

[66] S. Ilham, G. Abderrahim, and B. A. Abdelhakim, *Clustering Android Applications Using K-Means Algorithm Using Permissions*, 2019.

[67] F. Noorbehbahani, F. Rasouli, and M. Saberi, "Analysis of machine learning techniques for ransomware detection," in *Proceedings of the 2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, August 2019.

[68] M. Shafiq, X. Yu, A. K. Bashir, H. N. Chaudhry, and D. Wang, "A machine learning approach for feature selection traffic classification using security analysis," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 4867–4892, 2018.

[69] D. Bekerman, B. Shapira, L. Rokach, and A. Bar, "Unknown malware detection using network traffic classification," in *Proceedings of the 2015 IEEE Conference on Communications and NetworkSecurity, CNS 2015*, pp. 134–142, IEEE, Florence, Italy, september 2015.

[70] A. H. Lashkari, A. F. Akadir, H. Gonzalez, K. F. Mbah, and A. A. Ghorbani, "Towards a network-based framework for android malware detection and characterization," in *Proceedings of the 2017 15th Annual Conference on Privacy, Security and Trust, PST 2017*, pp. 233–242, Institute of Electrical and Electronics Engineers Inc., September 2018.

[71] T. Rajan and J. Wong Wan, "Chiew kang leng and johari abdullah. "DATDroid: dynamic analysis technique in android malware detection"," *International Journal of Advanced Science, Engineering and Information Technology*, vol. 10, no. 2, pp. 536–541, 2020.

[72] J. M. J. Hernandez Jimenez and K. Goseva-Popstojanova, "The effect on network flows-based features and training set size on malware detection," *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, IEEE, in *Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pp. 1–9, November 2018.

[73] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "Detecting android malware leveraging text semantics of network flows," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1096–1109, 2018.

[74] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Mobile malware detection through analysis of deviations in application network behavior," *Computers & Security*, vol. 43, no. 1–18, 2014.

[75] C. Manzano, C. Meneses, and P. Leger, "An empirical comparison of supervised algorithms for ransomware identification on network traffic," in *Proceedings of the International Conference of the Chilean Computer Science Society, SCCC*, November 2020.