

Complexity

Advances in Complex Systems and Their Applications to Cybersecurity

Lead Guest Editor: Fernando Sánchez Lasheras

Guest Editors: Danilo Comminiello and Alicja Krzemień





Advances in Complex Systems and Their Applications to Cybersecurity

Complexity

Advances in Complex Systems and Their Applications to Cybersecurity

Lead Guest Editor: Fernando Sánchez Lasheras

Guest Editors: Danilo Comminiello and Alicja Krzemień



Copyright © 2019 Hindawi. All rights reserved.



This is a special issue published in "Complexity." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Editorial Board






- José A. Acosta, Spain
Carlos F. Aguilar-Ibáñez, Mexico
Mojtaba Ahmadiéh Khanesar, UK
Tarek Ahmed-Ali, France
Alex Alexandridis, Greece
Basil M. Al-Hadithi, Spain
Juan A. Almendral, Spain
Diego R. Amancio, Brazil
David Arroyo, Spain
Mohamed Boutayeb, France
Átila Bueno, Brazil
Arturo Buscarino, Italy
Guido Caldarelli, Italy
Eric Campos-Canton, Mexico
Mohammed Chadli, France
Émile J. L. Chappin, Netherlands
Diyi Chen, China
Yu-Wang Chen, UK
Giulio Cimini, Italy
Danilo Comminiello, Italy
Sara Dadras, USA
Sergey Dashkovskiy, Germany
Manlio De Domenico, Italy
Pietro De Lellis, Italy
Albert Diaz-Guilera, Spain
Thach Ngoc Dinh, France
Jordi Duch, Spain
Marcio Eisencraft, Brazil
Joshua Epstein, USA
Mondher Farza, France
Thierry Floquet, France
Mattia Frasca, Italy
José Manuel Galán, Spain
Lucia Valentina Gambuzza, Italy
Bernhard C. Geiger, Austria
Carlos Gershenson, Mexico
- Peter Giesl, UK
Sergio Gómez, Spain
Lingzhong Guo, UK
Xianggui Guo, China
Sigurdur F. Hafstein, Iceland
Chittaranjan Hens, India
Giacomo Innocenti, Italy
Sarangapani Jagannathan, USA
Mahdi Jalili, Australia
Jeffrey H. Johnson, UK
M. Hassan Khooban, Denmark
Abbas Khosravi, Australia
Toshikazu Kuniya, Japan
Vincent Labatut, France
Lucas Lacasa, UK
Guang Li, UK
Qingdu Li, China
Chongyang Liu, China
Xiaoping Liu, Canada
Xinzhi Liu, Canada
Rosa M. Lopez Gutierrez, Mexico
Vittorio Loreto, Italy
Noureddine Manamanni, France
Didier Maquin, France
Eulalia Martínez, Spain
Marcelo Messias, Brazil
Ana Meštrović, Croatia
Ludovico Minati, Japan
Ch. P. Monterola, Philippines
Marcin Mrugalski, Poland
Roberto Natella, Italy
Sing Kiong Nguang, New Zealand
Nam-Phong Nguyen, USA
B. M. Ombuki-Berman, Canada
Irene Otero-Muras, Spain
Yongping Pan, Singapore
- Daniela Paolotti, Italy
Cornelio Posadas-Castillo, Mexico
Mahardhika Pratama, Singapore
Luis M. Rocha, USA
Miguel Romance, Spain
Avimanyu Sahoo, USA
Matilde Santos, Spain
Josep Sardanyés Cayuela, Spain
Ramaswamy Savitha, Singapore
Hiroki Sayama, USA
Michele Scarpiniti, Italy
Enzo Pasquale Scilingo, Italy
Dan Seluşteanu, Romania
Dehua Shen, China
Dimitrios Stamovlasis, Greece
Samuel Stanton, USA
Roberto Tonelli, Italy
Shahadat Uddin, Australia
Gaetano Valenza, Italy
Alejandro F. Villaverde, Spain
Dimitri Volchenkov, USA
Christos Volos, Greece
Qingling Wang, China
Wenqin Wang, China
Zidong Wang, UK
Yan-Ling Wei, Singapore
Honglei Xu, Australia
Yong Xu, China
Xinggang Yan, UK
Baris Yuçe, UK
Massimiliano Zanin, Spain
Hassan Zargazadeh, USA
Rongqing Zhang, USA
Xianming Zhang, Australia
Xiaopeng Zhao, USA
Quanmin Zhu, UK

Contents

Advances in Complex Systems and Their Applications to Cybersecurity

Fernando Sánchez Lasheras , Danilo Communiello , and Alicja Krzemień
Editorial (2 pages), Article ID 3261453, Volume 2019 (2019)


Delving into Android Malware Families with a Novel Neural Projection Method

Rafael Vega Vega , Héctor Quintián, Carlos Cambra , Nuño Basurto , Álvaro Herrero ,
and José Luis Calvo-Rolle 
Research Article (10 pages), Article ID 6101697, Volume 2019 (2019)

Multiclass Classification Procedure for Detecting Attacks on MQTT-IoT Protocol

Hector Alaiz-Moreton , Jose Avelaira-Mata , Jorge Ondicol-Garcia, Angel Luis Muñoz-Castañeda,
Isaías García, and Carmen Benavides
Research Article (11 pages), Article ID 6516253, Volume 2019 (2019)

Detection of Jihadism in Social Networks Using Big Data Techniques Supported by Graphs and Fuzzy Clustering

Cristina Sánchez-Rebollo, Cristina Puente , Rafael Palacios , Claudia Piriz, Juan P. Fuentes,
and Javier Jarauta
Research Article (13 pages), Article ID 1238780, Volume 2019 (2019)

Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm

Noemí DeCastro-García , Ángel Luis Muñoz Castañeda, David Escudero García, and Miguel V. Carriegos
Research Article (16 pages), Article ID 6278908, Volume 2019 (2019)

Practical Employment of Granular Computing to Complex Application Layer Cyberattack Detection

Rafał Kozik , Marek Pawlicki , Michał Choraś, and Witold Pedrycz
Research Article (9 pages), Article ID 5826737, Volume 2019 (2019)

Editorial

Advances in Complex Systems and Their Applications to Cybersecurity

Fernando Sánchez Lasheras ¹, **Danilo Comminiello** ², and **Alicja Krzemień**³

¹Mathematics Department, University of Oviedo, c/Federico García Lorca 18, 33007 Oviedo, Spain

²Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, Via Eudossiana 18, 00184 Rome, Italy

³Department of Risk Assessment and Industrial Safety, Central Mining Institute, Plac Gwarków 1, 40166 Katowice, Poland

Correspondence should be addressed to Fernando Sánchez Lasheras; sanchezfernando@uniovi.es

Received 25 February 2019; Accepted 25 February 2019; Published 4 June 2019

Copyright © 2019 Fernando Sánchez Lasheras et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Cybersecurity is one of the fastest growing and largest technology sectors and is increasingly being recognized as one of the major issues in many industries, so companies are increasing their security budgets in order to guarantee the security of their processes. Successful menaces to the security of information systems could lead to safety, environmental, production, and quality problems.

One of the most harmful issues of attacks and intrusions is the ever-changing nature of attack technologies and strategies, which increases the difficulty of protecting computer systems. As a result, advanced systems are required to deal with the ever-increasing complexity of attacks in order to protect systems and information.

This special issue received several contributions, 5 of which have been accepted for publication.

In the article “Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm” by N. DeCastro-García et al., the authors investigate on the use of different partitions of a dataset in the hyperparameter optimization phase over the efficiency of a machine learning algorithm. Nonparametric inference has been used to measure the rate of different behaviors of the accuracy. A level of gain is assigned to each partition allowing authors to study patterns and allocate whose samples are more profitable. The statistical analyses were carried out over five cybersecurity datasets.

In the article “Detection of Jihadism in Social Networks Using Big Data Techniques Supported by Graphs and Fuzzy

Clustering” by C. Sánchez-Rebollo et al., the authors performed an analysis of Twitter messages to detect the leaders orchestrating terrorist networks and their followers. A big data architecture is proposed to analyze messages in real time in order to classify users according to different parameters like level of activity, the ability to influence other users, and the contents of their messages. Graphs have been used to analyze how the messages propagate through the network and fuzzy clustering techniques were used to classify users in profiles. The Algorithms test was performed with the help of public database from Kaggle and other Twitter extraction techniques.

In the article “Delving into Android Malware Families with a Novel Neural Projection Method” by R. V. Vega et al., the authors proposed the application of unsupervised and supervised machine-learning techniques to characterize Android malware families. More precisely, a novel unsupervised neural-projection method for dimensionality-reduction, namely, Beta Hebbian Learning (BHL), was applied to visually analyze such malware. Additionally, well-known supervised Decision Trees (DTs) are also applied to improve characterization of such. The proposed techniques are validated when facing real-life Android malware data by means of the well-known and publicly-available Malgenome dataset.

In the article “Multiclass Classification Procedure for Detecting Attacks on MQTT-IoT Protocol” by H. A. Moretón et al., the authors proposed the creation of classification

models that can feed an Intrusion Detection System using a dataset containing frames under attacks of an Internet of Things (IoT) system that uses the MQTT protocol. Two kinds of methods are applied: ensemble methods and recurrent networks, achieving very satisfactory results.

Finally, in the article “Practical Employment of Granular Computing to Complex Application Layer Cyberattack Detection” by R. Kozik et al., the authors propose a novel approach to the detection of cyberattacks taking inventory of the practical application of information granules. Also, the feasibility of utilizing Granular Computing (GC) as a solution to the most current challenges in cybersecurity is researched. Promising results have been shown on a benchmark dataset.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

The editorial team would like to express appreciation to all authors for their valuable contributions and to all reviewers for their valuable comments. In addition, the editors would like to thank the Complexity Journal’s Editorial Board for their valuable help and support regarding this special issue.

*Fernando Sánchez Lasheras
Danilo Comminiello
Alicja Krzemień*

Research Article

Delving into Android Malware Families with a Novel Neural Projection Method

Rafael Vega Vega ¹, Héctor Quintián,¹ Carlos Cambra ², Nuño Basurto ²,
Álvaro Herrero ² and José Luis Calvo-Rolle ^{1,3}

¹University of A Coruña, Departamento de Ingeniería Industrial, Avda. 19 de febrero s/n, 15495, Ferrol, A Coruña, Spain

²Grupo de Inteligencia Computacional Aplicada (GICAP), Departamento de Ingeniería Civil, Escuela Politécnica Superior, Universidad de Burgos, Av. Cantabria s/n, 09006, Burgos, Spain

³Research Institute of Applied Sciences in Cybersecurity (RIASC), Spain

Correspondence should be addressed to Rafael Vega Vega; rafael.alejandro.vega.vega@udc.es

Received 5 December 2018; Accepted 23 January 2019; Published 2 June 2019

Guest Editor: Alicja Krzemień

Copyright © 2019 Rafael Vega Vega et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Present research proposes the application of unsupervised and supervised machine-learning techniques to characterize Android malware families. More precisely, a novel unsupervised neural-projection method for dimensionality-reduction, namely, Beta Hebbian Learning (BHL), is applied to visually analyze such malware. Additionally, well-known supervised Decision Trees (DTs) are also applied for the first time in order to improve characterization of such families and compare the original features that are identified as the most important ones. The proposed techniques are validated when facing real-life Android malware data by means of the well-known and publicly available Malgenome dataset. Obtained results support the proposed approach, confirming the validity of BHL and DTs to gain deep knowledge on Android malware.

1. Introduction and Previous Work

Undoubtedly, smartphones are one of the emerging technologies that have revolutionized the use of computing systems. From the very beginning (late 1990s), more and more smartphones are sold every year and it is expected that the number of smartphone users passes the 2.7 billion mark by 2019 [1]. Although there is a variety of operating systems for such devices, Google's Android is the most widely used one [1] and, consequently, the number of Android users has permanently increased. Concurrently, the number of Android apps strongly increased in the last years but it started to decline from 3.6 million in March, 2017 (highest value), to 2.6 million in September, 2018 [2].

From the security standpoint, one of the main problems of smartphone apps is malware that is included in software in general and in these apps in particular. Furthermore, "users of mobile devices are increasingly subject to malicious activity pushing malware apps" [3]. It is true that some effort has been devoted by Google to remove and prevent malicious

apps from Google Play Market, but malware is still there [3]. Moreover, malware Android apps are increasing; in the third trimester of 2018 there has been an increase of 1.7 million detections [4].

As it can be seen, privacy and security of smartphones still are open challenges [5] and many researchers are working on this topic. To better fight against malware and be able to develop an effective solution, understanding it and its nature is required [6]. In keeping with this idea, present paper proposes getting deeper knowledge about Android malware for its better detection. More precisely, both supervised (Decision Trees) and unsupervised (Neural Projection Method) machine-learning techniques are applied to increase our knowledge about the main families of Android malware. In order to validate the proposed techniques, they are applied to the well-known Malgenome dataset [7] that is open and real-life.

This pioneering work on collecting Android malware found some interesting statistics [6] motivating further analysis of malware: 36.7% of the collected apps leverage root-level

exploits to fully compromise the security of the smartphone and more than 90% of the apps tried to turn the smartphone into a botnet controlled through network or short messages.

To improve present knowledge of Android malware families, a novel neural-projection technique from the family of Exploratory Projection Pursuit (EPP) techniques, named Beta Hebbian Learning (BHL) [8], is applied. Obtained results are then compared to those from several different Decision Trees (DTs) [9] when trying to predict the malware family from apps features.

Each app (data sample) that was collected for the Malgenome dataset is defined as a set of certain features using a binary representation. Apps were grouped according to the family they belong to, and features were recalculated for the whole family, taking into account which features were present in the given apps. The generated high-dimensional space is then analysed by means of BHL in order to reveal the inner structure of the dataset. Obtained projections are consequently scrutinized to get further knowledge about the app features that define the organization of the data in different groups and subgroups. For comparison purposes, DTs have been additionally generated on the same features set, in order to know the features that better discriminate between the different malware families.

A variety of problems have been addressed by artificial neural networks in recent decades [10–14]. More precisely, neural projection models have been previously applied to a wide variety of security datasets, including network traffic [15, 16], SQL code [17, 18], and HTTP traffic [19]. Similarly, from a more general perspective, different machine learning solutions have been proposed to differentiate between legitimate and malicious apps [20–22].

Visualization techniques have been previously applied to this problem of analyzing malware [23–29]. However, few dimensionality-reduction techniques have been applied to Android apps in order to detect malware; Pythagoras tree fractal visualization is proposed in [25], being all apps scattered, as leaves in the tree. Graphs for deciding about malicious apps by depicting lists of malicious methods, needless permissions, and malicious strings were proposed in [26]. Biclustering on permission information was used to generate visualization in [27], while behavior-related dendrograms are generated out of malware traces in [28]. In the later, different pieces of information are analysed, including nodes related to the package name of the application, the Android components that has called the API call, and the names of functions and methods invoked by the application. Differentiating from previous work, in present paper, a novel neural projection technique is applied for the first time to the characterization of Android malware [8, 24, 30]. Apps are not analysed one by one, but family-level is considered instead. Additionally, DTs are applied for the first time in order to improve characterization of such families.

The rest of this paper is organized as follows; initially BHL and DTs are presented and the analyzed dataset is described in the following section. Then, the proposed experiments are introduced and the obtained results are analyzed in Section 3. Finally, conclusions and future work are presented in Section 4 of the paper.

2. Materials and Methods

In present research, the EPP BHL algorithm [8] has been applied to a dataset of malware families with the aim of identifying the internal structure of such dataset and finding families of malware with similar characteristics. The obtained results have been compared with a well-known prediction algorithm (DT) [9] to validate the BHL results regarding the most relevant features to briefly characterize Malware families.

2.1. Beta Hebbian Learning. The Beta Hebbian Learning technique (BHL) [8] is an unsupervised neural network from the family of EPP that employs the Beta distribution to update its learning rule and fit the Probability Density Function (PDF) of the residual with the distribution of a given dataset.

Thus, if the PDF of the residuals is known, the optimal cost function can be determined. By using $B(\alpha, \beta)$ parameters of the Beta distribution, the residual (e) can be drawn with the following PDF:

$$p(e) = e^{\alpha-1} (1-e)^{\beta-1} = (x-Wy)^{\alpha-1} (1-x+Wy)^{\beta-1} \quad (1)$$

where α and β are used to adjust the shape of the PDF of the Beta distribution, x is the input of the network, e is the residual, W is the weight matrix, and y is the output of the network.

Then, by using the following, gradient descent is performed to maximize the likelihood of the weights:

$$\begin{aligned} \frac{\partial p}{\partial W} &= \left(e_j^{\alpha-2} (1-e_j)^{\beta-2} (-(\alpha-1)(1-e_j) + e_j(\beta-1)) \right) \\ &= \left(e_j^{\alpha-2} (1-e_j)^{\beta-2} (1-\alpha+e_j(\alpha+\beta-2)) \right) \end{aligned} \quad (2)$$

In the case of BHL, the learning rule allows for fitting the PDF of the residual, by maximizing the likelihood of such residual with the current distribution.

Therefore, the neural architecture for BHL is defined as follows:

$$\text{Feedforward} : y_i = \sum_{j=1}^N W_{ij} x_j, \quad \forall i \quad (3)$$

$$\text{Feedback} : e_j = x_j - \sum_{i=1}^M W_{ij} y_i \quad (4)$$

$$\begin{aligned} \text{Weightsupdate} : \Delta W_{ij} \\ = \eta \left(e_j^{\alpha-2} (1-e_j)^{\beta-2} (1-\alpha+e_j(\alpha+\beta-2)) \right) y_i \end{aligned} \quad (5)$$

2.2. Decision Trees. Decision Trees (DTs) [9] are machine-learning algorithms widely used for prediction that have proved their benefits in several real applications. They can be categorized as supervised nonparametric inductive learning techniques. They are based on the construction of diagrams from a dataset, in a similar way to prediction systems based

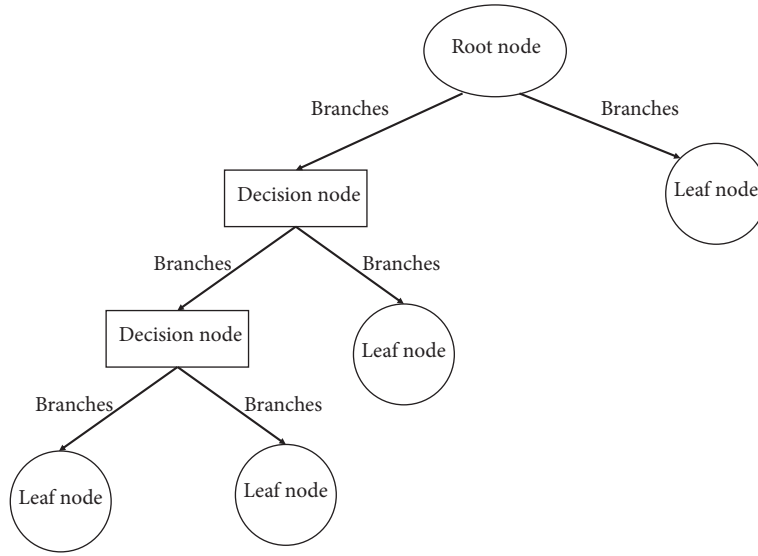


FIGURE 1: Structure of decision trees.

on rules, which serve to represent and categorize a series of conditions that occur repeatedly for the solution of a problem.

The main objective of a classification DT is to divide a dataset into groups of samples as similar as possible in relation to one of the features. They are made of three main elements: root node (contains all samples of the dataset), decision nodes (represent a decision or rule), and leaf nodes (final label). A dataset is then classified based on subdivisions of the DT nodes to reach one of the final (leaf) nodes whose label corresponds to a class (Figure 1).

Several algorithms have been proposed so far to build DTs and their efficiency has been proved. The most notable ones [31] are ID3 (Iterative Dichotomiser 3), C4.5 (successor of ID3), CART (Classification and Regression Tree), CHAID (CHi-squared Automatic Interaction Detector), MARS, and Conditional Inference Trees. Among all of them, CART has been selected in present work due to two main reasons: the binary nature of the dataset and the main objective of the study (to identify the most relevant features of the dataset) [31].

2.2.1. CART. The Classification and Regression Tree (CART) [9] is a binary tree, so each decision node has two binary branches determined by a splitting function obtained by processing variance function. In order to build the tree, this CART algorithm takes 4 main steps [9]:

- (1) Build the decision tree splitting nodes according to a given function.
- (2) Finish tree construction once the learning fits the stop criteria.
- (3) Pruning the tree to avoid overfitting.
- (4) Select the best tree after pruning process.

Originally, the splitting function used by CART is the Gini Index

$$Gini(S) = 1 - \sum_{i=1}^n p_i^2 \quad (6)$$

where S is the dataset, n is the number of classes in the dataset, and p is the probability of different classes. Therefore, a Gini index of 0 means a 100% accuracy in predicting the class.

For comparison purposes, two other splitting functions have been applied in present paper: Deviance (7) and Twoing (8)

$$Deviance(S) = - \sum_{i=1}^n p_i \log_2 p_i \quad (7)$$

Twoing is a splitting function different from Gini and Deviance. Being L_i and R_i there is fraction of members of class i in the left and right child nodes after a split, respectively. P_L and P_R are the fractions of observations that split to the left and right, respectively. Therefore, the function to be maximized is the one in

$$P_L P_R \left(\sum_{i=1}^n |L_i - R_i| \right)^2 \quad (8)$$

On the other hand, in standard CART algorithm, the split feature that is selected for a decision node is the one that maximizes the split-criterion gain. Once again, for a more comprehensive comparison, two other criteria have been applied for selecting split features: curvature [32] and interaction [33]. These criteria can be defined as follows:

- (i) Curvature: it is based on the null hypothesis of unassociated two features. With these criteria, the best split predictor feature is the one that minimizes the

significant p -values of curvature tests between each feature and the response variable. Such a selection is robust to the number of levels in individual features.

- (ii) Interaction: it is based on the null hypothesis of no interaction between the label and the predictor features. Therefore, for deep decision trees, standard CART tends to miss important interactions between pairs of features when there are also many other less important features. By means of this criterion, the detection of such important interactions is improved.

2.3. Malgenome Dataset. The dataset used in this research has been obtained from the Android Malware Genome Project [7], which consists on 1260 Android malware samples grouped in a total of 49 malware families. It was collected from August 2010 to October 2011 and still is a standard benchmark dataset for Android Malware.

This dataset contains malware apps installed in user phones and based on 3 main attack strategies: repackaging, update attack, and drive-by download. Samples of this dataset were manually classified based on different aspects such as installation and activation mechanisms and malicious payloads nature. Collected malware was split in families that were obtained “by carefully examining the related security announcements, threat reports, and blog contents from existing mobile antivirus companies and active researchers as exhaustively as possible and diligently requesting malware samples from them or actively crawling from existing official and alternative Android Markets” [6].

The different families present in the dataset are ADRD, AnserverBot, Asroot, BaseBridge, BeanBot, BgServ, CoinPirate, Crusewin, DogWars, DroidCoupon, DroidDeluxe, DroidDream, DroidDreamLight, DroidKungFu1, DroidKungFu2, DroidKungFu3, DroidKungFu4, DroidKungFuSapp, DoidKungFuUpdate, Endofday, FakeNetflix, FakePlayer, GamblerSMS, Geinimi, GGTracker, GingerMaster, GoldDream, Gone60, GPSSMSpy, HippoSMS, Jifake, jSMShider, Kmin, Lovetrap, NickyBot, Nickyspy, Pjapps, Plankton, RogueLemon, RogueSPPush, SMSReplicator, SndApps, Spitmo, TapSnake, Walkinwat, YZHC, zHash, Zitmo, and Zsone.

Therefore, the final dataset is made of a total of 49 samples, one for each family of malware, defined by a total of 26 binary features divided in 6 categories (Table 1). A detailed description of each feature can be found in the original paper [6], and some previous work where this dataset is used can be found in [34–36].

3. Experiments and Results

This section presents the experiments performed and the results obtained in the validation process of the proposed solution.

Both BHL (Section 2.1) and DT (Section 2.2) algorithms have been applied to the previously described dataset (Section 2.3), in order to identify the features that define the internal structure of the data and that support the grouping of the different families of malware attacks. In the conducted

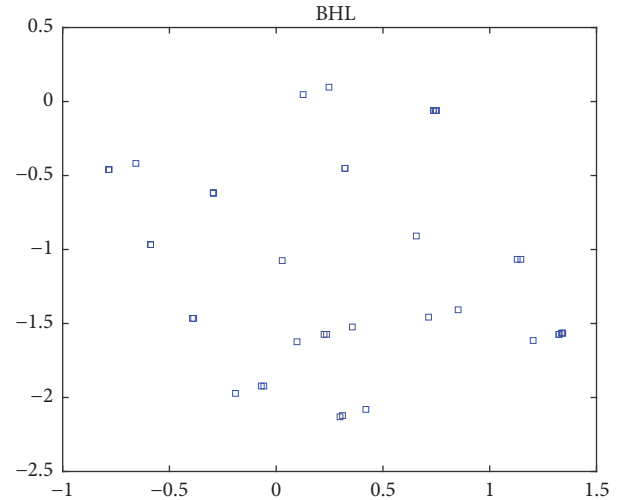


FIGURE 2: BHL: Projection of malware families.

experiments, firstly BHL is applied to identify groups of malware families with similar behaviour. This is done by visually inspecting the obtained BHL projections, and the most relevant features are consequently identified. Then, the dataset is analyzed by means of DTs to determine the level of importance of each feature, considering as the most relevant features those that are used in the decision nodes at lowest depth.

In Figure 2 it is shown the best projection obtained by BHL using the following parameter values: $\alpha = 3$, $\beta = 4$, *numberofiterations* = 1000, and *learningrate* = 0.05. These parameter values were chosen in an experimental process of trial and error. As parameter tuning is a task that is very dependent on the dataset to be used, several initial experiments were conducted with a range of combinations of these parameter values.

Based on such projection, samples are grouped in 2 main clusters: G1 and G2 (Figure 3). Additionally, several subgroups (at a 3 level depth, i.e., $G1 \rightarrow G1A \rightarrow G1A.1, G1A.2, G1A.3, \text{ and } G1A.4$) can be defined in these main groups.

Figure 4 presents the split of family groups in a schema that shows the results of thoroughly analyzing the allocation of families in groups. The most relevant features that have been identified, varying from one cluster to another, can be seen. As an example, data are split in G1 and G2 based on the features “Repackaging” and “Standalone.” The complete lists of families assigned to each one of the groups are presented in Figures 5 and 6. Malware families are allocated in the same group as they are associated to similar characteristics and behaviour, and therefore there could be similar ways to deal with them.

Based on the analysis of BHL results, the most relevant features, in decreasing order of importance, are “Repackaging” and “Standalone,” “Boot” and “Activation: SMS,” and “Financial Charges: SMS.”

BHL clearly outperforms other algorithms used in previous works [24, 29], providing a clearer visualization

TABLE 1: Features in the Malgenome Dataset.

Category 1: Installation	1.Repackaging, 2.Update, 3.Drive-by download, 4.Standalone
Category 2: Activation	5.Boot, 6.SMS, 7.Net, 8.Call, 9.USB, 10.PKG, 11.Batt, 12.SYS, 13.Main
Category 3: Privilege escalation	14.exploit, 15.RATC/zimperlich, 16.ginger break, 17.asroot, 18.encrypted
Category 4: Remote control	19.NET, 20.SMS
Category 5: Financial charges	21.phone call, 22.SMS, 23.block SMS
Category 6: Personal information stealing	24.SMS, 25.phone number, 26.user account

TABLE 2: Summary table of DT results: minimum depth of decision nodes for each one of the original features.

ID	Feature	Deviance			Gini			Twoing			Average
		Standard	Curvature	Interaction curvature	Standard	Curvature	Interaction curvature	Standard	Curvature	Interaction curvature	
1	Repackaging	1	2	4	1	2	6	1	2	4	2.56
5	BOOT	2	3	3	4	3	2	2	3	3	2.78
18	Encrypted		4	2		4			4	2	3.20
9	USB	6	3		5	3	4	6	3		4.29
3	Drive-by Download	5	5	4	2	5	6	5	5	4	4.56
24	SMS	4	3	5	10	3	6	3	3	5	4.67
26	User Account	6	1	8	3	1	8	6	1	8	4.67
2	Update	5	7	4	2	6	3	5	7	4	4.78
19	NET	6	2	8	9	2	3	4	2	8	4.89
6	SMS	3	6	5	8	7	3	3	6	4	5.00
10	PKG	6	5		4	5	4	6	5		5.00
22	SMS	3	6	4	10	6	4	3	6	4	5.11
4	Standalone	5	10	3	3	9	3	5	10	3	5.67
8	CALL	4		7	4		8	4		7	5.67
11	BATT	4	9		5	4	5	4	9		5.71
16	Ginger Break				6						6.00
15	RATC/Zimperlich	6	8	1	9	9	8	7	8	1	6.33
7	NET	5	8	6	10	9	2	5	8	6	6.56
14	Exploit	5	8	6	9	6	6	5	8	6	6.56
17	Asroot	7	4	7	11	4	9	8	4	7	6.78
23	Block SMS	3	8	5	9	9	9	4	8	6	6.78
25	Phone Number	2	11	2	12	12	11	2	11	2	7.22
12	SYS	5	10	6	10	11	7	5	10	6	7.78
21	Phone Call	4	9		12	13	7	4	9	5	7.88
13	MAIN	6	11	7	10	12	1	6	11	7	7.89
20	SMS				10		8				9.00

of the internal structure of the dataset. Groups obtained by BHL are more compact and well defined than the groups generated by other EPP techniques in the previous work.

In addition to the BHL experiments, experiments with DTs were additionally conducted in order to compare and validate the obtained results. As it has been previously mentioned, 3 different splitting functions have been applied in present paper: Gini, Deviance, and Twoing. In addition, 3 different criteria for selecting split features have been applied: Standard, Curvature, and Interaction.

As an example, one of the obtained DTs is shown in Figure 7. This is the tree generated from the Malgenome dataset when applying the standard CART split criteria and the Deviance function. It has been selected as it is the one with lowest depth. In the leaf nodes, labels refer to family numbers (alphabetically ordered as presented in Section 2.3).

To show the most interesting results from the different alternatives to build DT, information has been summarized in Table 2. For each combination of splitting function and selecting criteria, the minimum depth of decision nodes linked to each one of the original features is shown. That is,

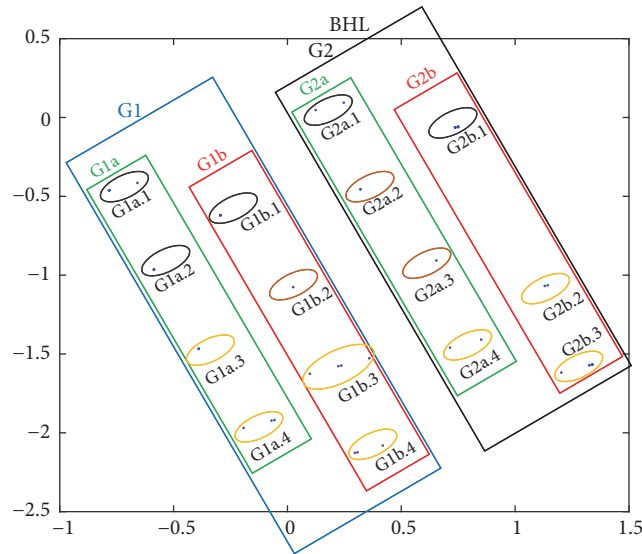


FIGURE 3: BHL: Labelling of clusters.

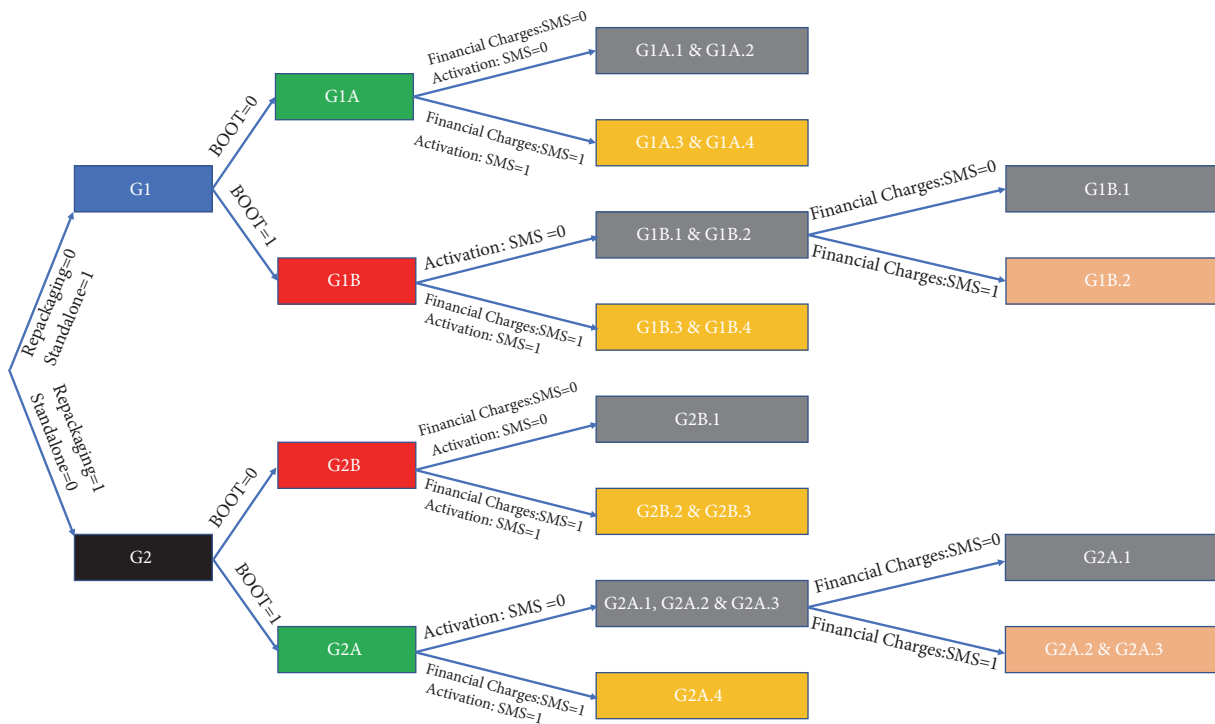


FIGURE 4: Schematic clustering and relevant features from BHL projection.

when the same feature appears in more than one node, the minimum depth of all these nodes is the one selected for the given feature. In the case a certain feature was not included in the DT, there is no value.

In this table it can be seen that results (slightly or significantly) vary when comparing the obtained results (by different splitting function and selecting criteria) for a certain feature. As general conclusions cannot be derived and to sum

up all figures, the average depth value is calculated for each feature, that is, further analyzed.

When analyzing Figure 4 and Table 2, it can be seen that results from BHL and DT are coherent. In the case of BHL, it can be seen that Repackaging is identified as the most discriminative feature, because the two main groups in the dataset (G1 and G2) take complementary values for such feature. Coherently, Repackaging is the feature with the

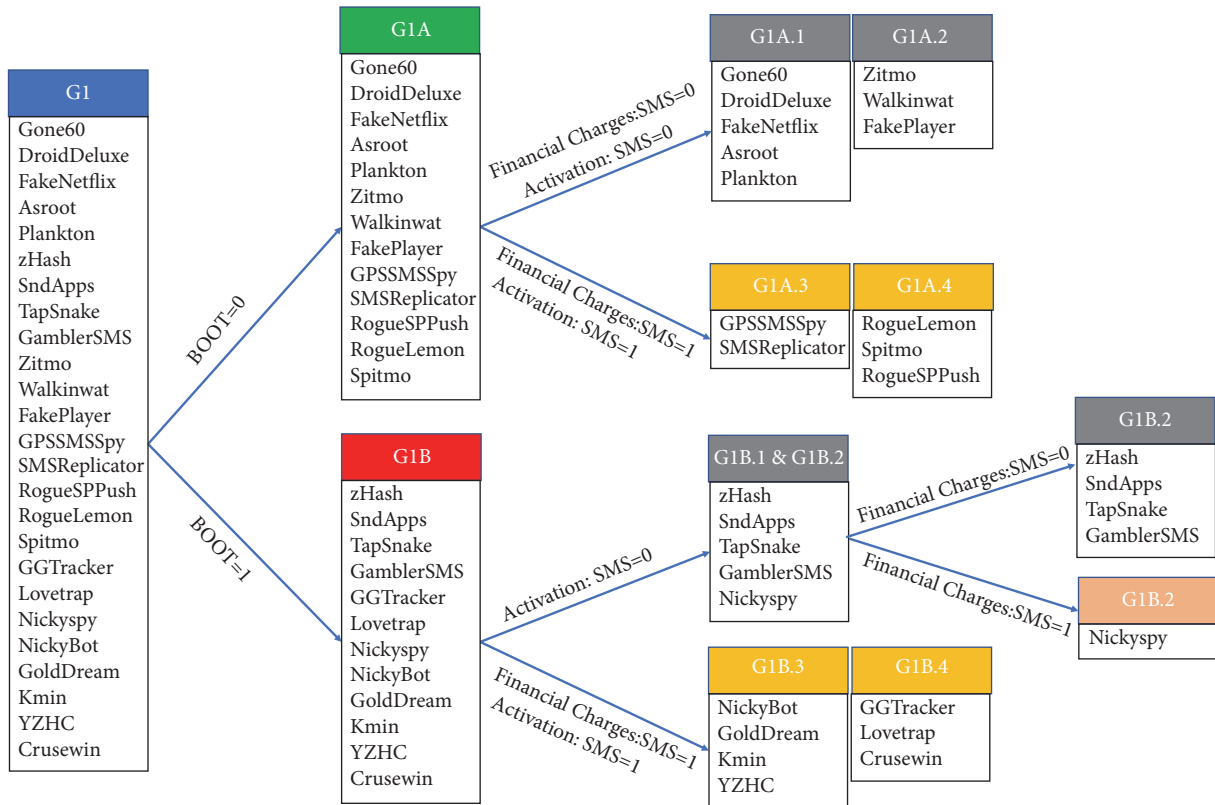


FIGURE 5: Families allocation in Group 1 and relevant features identified in BHL projection.

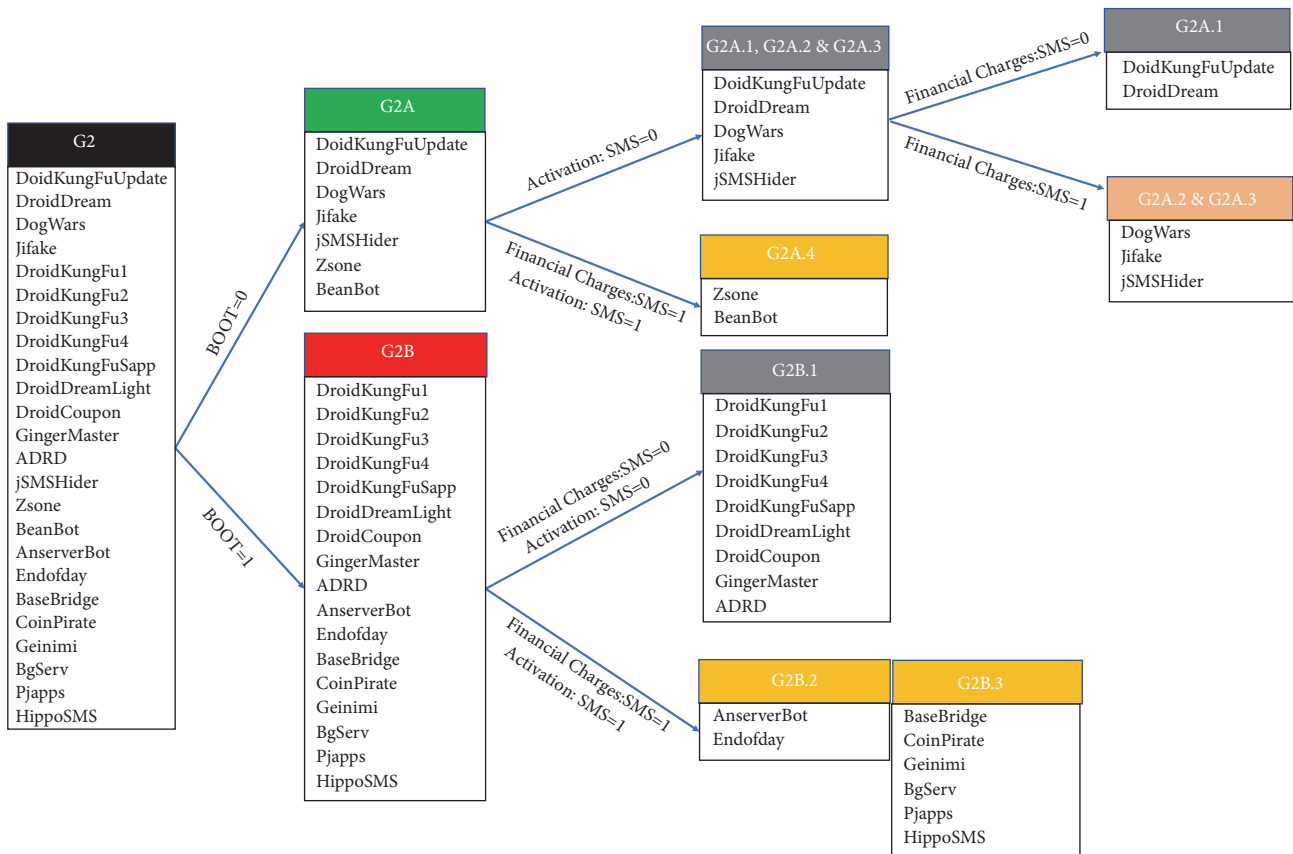


FIGURE 6: Families allocation in Group 2 and relevant features identified in BHL projection.

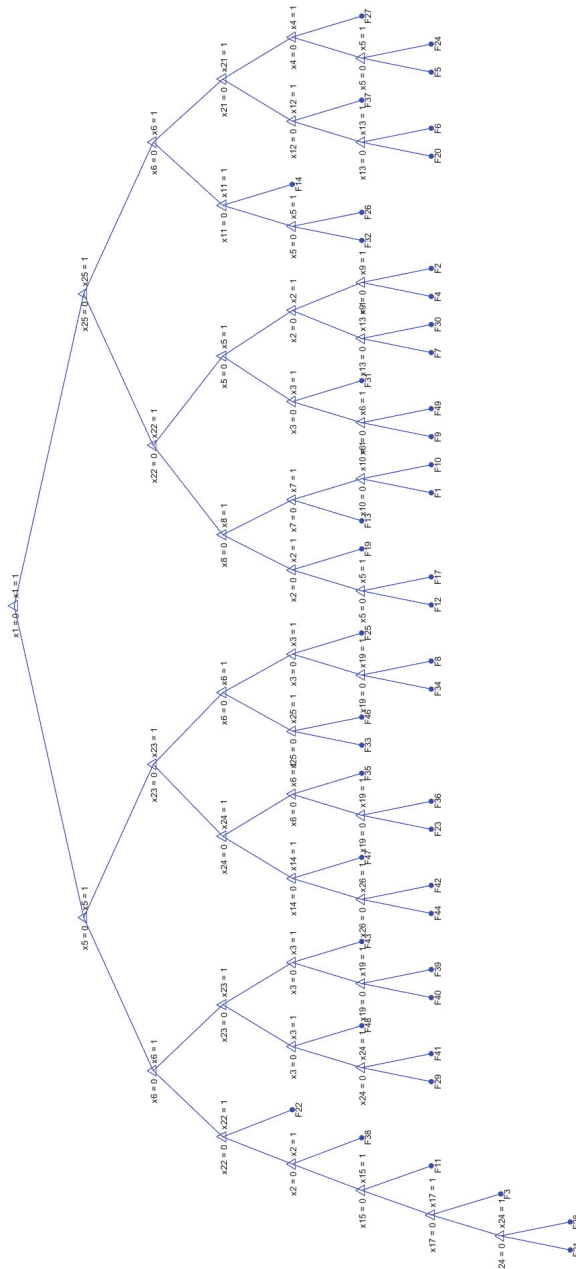


FIGURE 7: DT obtained with standard CART split criteria and Deviance function.

lowest mean depth, being included in all the generated trees. Furthermore, it was selected for the root node of 3 DTs. When analyzing subgroups in BHL projection (Figure 3), it can be seen that BOOT is the feature that drive the split in 1st-level subgroups (subgroups G1A and G1B in the case of group G1, and subgroups G2A and G2B in the case of group G2). In keeping with this idea, according to DTs results, Boot is the second feature with the lowest mean depth. For the next level of importance, the BHL projection identifies Financial Charges SMS and Activation SMS as the features that best explain the split in different subgroups. The two of them are

also selected by DTs as ranked in the first half of features with a lowest mean depth, although some other features take lower values.

Additionally, from the DTs results (Table 2), Privilege escalation-Ginger Break and Remote control-SMS can be identified as the least relevant features. The former was not included in 7 (out of 9) DTs while the latter was not included in 6. Furthermore, Remote control-SMS is the feature with a highest value of the average depth, taking a value of 9. It means that these features are almost useless when characterizing malware families.

Results from present paper are consistent with those obtained in previous work [30] when applying Feature Selection (FS) to the same dataset. Installation-Repackaging, Activation-SMS, Activation-Boot, Remote Control-NET, and Financial Charges-SMS were identified as the 5 most relevant features in order to characterize malware families, according to a given method of filter-based FS: Minimum-Redundancy Maximum Relevance. This method is intended at obtaining the maximum relevance to the output while keeping redundancy of selected features to lowest levels. Complementarily, two evolutionary approaches to FS (GA-ICC-W and GA-I-W) identified Installation-Repackaging, Installation-Standalone, Activation-SMS, Remote Control-NET, and Financial Charges-SMS as the 5 most relevant ones. These methods perform the selection of features according to the Information Correlation Coefficient and the Mutual Information, respectively.

4. Conclusions and Future Work

In this paper, some machine learning techniques have been applied to Android malware data in order to analyse the features of such apps and subsequently identify the ones that better define the organization of malware families. As a result, detection and categorization of malware could be improved and sped up at the same time. Furthermore, by knowing about these features, malware apps could be identified more quickly and precisely and then removed from the official Android market.

From the obtained results some conclusions can be derived; first of all, the proposed machine-learning techniques probed to successfully address the given challenge. BHL has outperformed previous neural projection techniques that have been applied to the same data in clearly revealing the structure of the Malgenome dataset. Additionally, features identified as the most important ones by such EPP technique are also highlighted by DTs as being relevant to better differentiate between malware families.

Obtained results are consistent with those obtained by FS and hence validate present proposal. Future work will focus on the development of a Hybrid Intelligent System to integrate results from the previously validated machine-learning techniques. In addition, it will be applied to up-to-date malware datasets in order to check its performance when facing 0-day malware.

Data Availability

Dataset used in this research is available in [7]: Bibliography: 7. (2010) Malgenome Project [Online], available at <http://www.malgenomeproject.org>.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is partially supported by Instituto Nacional de Ciberseguridad (INCIBE) and developed by Research Institute of Applied Sciences in Cybersecurity (RIASC).

References

- [1] Gartner, Global smartphone sales to end users from 1st quarter 2009, 2018 <https://www.statista.com/statistics/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system/>.
- [2] AppBrain, Android and google play statistics, <https://www.appbrain.com/stats/stats-index>.
- [3] SOPHOSLABS, "Ltd., s., sophoslabs 2019 threat report," Tech. Rep., 2019.
- [4] M. Labs, "Cybercrime tactics and techniques : Q3 2018," Tech. Rep., 2018.
- [5] S. Arshad, M. A. Shah, A. Khan, and M. Ahmed, "Android malware detection & protection: A survey," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, 2016.
- [6] Y. Zhou and X. Jiang, "Dissecting android malware: characterization and evolution," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pp. 95–109, San Francisco, Calif, USA, May 2012.
- [7] Y. Zhou, Malgenome project. 2010 <http://www.malgenomeproject.org>.
- [8] H. Quintán and E. Corchado, "Beta hebbian learning as a new method for exploratory projection pursuit," *International Journal of Neural Systems*, vol. 27, no. 6, Article ID 1750024, 2017.
- [9] L. Breiman, *Classification and regression trees*, Routledge, London, UK, 2017.
- [10] P. J. García Nieto, J. Martínez Torres, F. J. De Cos Juez, and F. Sánchez Lasheras, "Using multivariate adaptive regression splines and multilayer perceptron networks to evaluate paper manufactured using *Eucalyptus globulus*," *Applied Mathematics and Computation*, vol. 219, no. 2, pp. 755–763, 2012.
- [11] M. Paliwal and U. A. Kumar, "Neural networks and statistical techniques: a review of applications," *Expert Systems with Applications*, vol. 36, no. 1, pp. 2–17, 2009.
- [12] R. F. Garcia, J. L. C. Rolle, M. R. Gomez, and A. D. Catoira, "Expert condition monitoring on hydrostatic self-levitating bearings," *Expert Systems with Applications*, vol. 40, no. 8, pp. 2975–2984, 2013.
- [13] C. C. Turrado, M. D. C. M. López, F. S. Lasheras, B. A. R. Gómez, J. L. C. Rollé, and F. J. D. C. Juez, "Missing data imputation of solar radiation data under different atmospheric conditions," *Sensors*, vol. 14, no. 11, pp. 20382–20399, 2014.
- [14] J. L. Calvo Rolle, I. Machón González, and H. López García, "Neuro-robust controller for non-linear systems," *Dyna*, vol. 86, no. 3, pp. 308–317, 2011.
- [15] Á. Herrero, E. Corchado, M. A. Pellicer, and A. Abraham, "Hybrid multi agent-neural network intrusion detection with mobile visualization," in *Innovations in Hybrid Intelligent Systems*, pp. 320–328, 2008.
- [16] R. Sánchez, Á. Herrero, and E. Corchado, "Visualization and clustering for SNMP intrusion detection," *Cybernetics and Systems*, vol. 44, no. 6-7, pp. 505–532, 2013.
- [17] C. Pinzón, Á. Herrero, J. F. De Paz, E. Corchado, and J. Bajo, "CBRid4SQL: A CBR intrusion detector for SQL injection

- attacks,” in *Proceedings of the 5th International Conference on Hybrid Artificial Intelligence Systems HAIS 2010 - Part II*, vol. 6077 of *Lecture Notes in Computer Science*, pp. 510–519, Springer Berlin Heidelberg, 2010.
- [18] C. Pinzón, J. F. De Paz, J. Bajo, Á. Herrero, and E. Corchado, “AIIDA-SQL: An adaptive intelligent intrusion detector agent for detecting SQL injection attacks,” in *Proceedings of the 2010 10th International Conference on Hybrid Intelligent Systems, HIS 2010*, pp. 73–78, USA, August 2010.
- [19] D. Atienza, Á. Herrero, and E. Corchado, “Neural Analysis of HTTP Traffic for Web Attack Detection,” in *Proceedings of the 8th International Conference on Computational Intelligence in Security for Information Systems CISIS 2015*, vol. 369 of *Advances in Intelligent Systems and Computing*, pp. 201–212, Springer International Publishing, 2015.
- [20] L. Cen, C. S. Gates, L. Si, and N. Li, “A probabilistic discriminative model for android malware detection with decompiled source code,” *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 400–412, 2015.
- [21] H.-J. Zhu, Z.-H. You, Z.-X. Zhu, W.-L. Shi, X. Chen, and L. Cheng, “DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model,” *Neurocomputing*, vol. 272, pp. 638–646, 2018.
- [22] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, “Evaluation of machine learning classifiers for mobile malware detection,” *Soft Computing*, vol. 20, no. 1, pp. 343–357, 2016.
- [23] M. Wagner, F. Fischer, R. Luh et al., “A Survey of Visualization Systems for Malware Analysis,” in *Proceedings of the Eurographics Conference on Visualization (EuroVis) - STARs*, 2015.
- [24] A. González, Á. Herrero, and E. Corchado, “Neural visualization of android malware families,” in *International Joint Conference SOCO’16-CISIS’16-ICEUTE’16*, vol. 527 of *Advances in Intelligent Systems and Computing*, pp. 574–583, Springer International Publishing, Cham, 2017.
- [25] A. Paturi, M. Cherukuri, J. Donahue, and S. Mukkamala, “Mobile malware visual analytics and similarities of Attack Toolkits (Malware gene analysis),” in *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013*, pp. 149–154, USA, May 2013.
- [26] W. Park, K. Lee, K. Cho, and W. Ryu, “Analyzing and detecting method of Android malware via disassembling and visualization,” in *Proceedings of the 2014 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 817–818, Busan, South Korea, October 2014.
- [27] V. Moonsamy, J. Rong, and S. Liu, “Mining permission patterns for contrasting clean and malicious android applications,” *Future Generation Computer Systems*, vol. 36, pp. 122–132, 2014.
- [28] O. Somarriba, U. Zurutuza, R. Uribeetxeberria, L. Delosières, and S. Najm-Tehrani, “Detection and visualization of android malware behavior,” *Journal of Electrical and Computer Engineering*, vol. 2016, Article ID 8034967, p. 17, 2016.
- [29] R. Vega Vega, H. Quintián, J. L. Calvo-Rolle, Á. Herrero, and E. Corchado, “Gaining deep knowledge of Android malware families through dimensionality reduction techniques,” *Logic Journal of the IGPL*, 2018.
- [30] J. Sedano, S. González, C. Chira, Á. Herrero, E. Corchado, and J. R. Villar, “Key features for the characterization of Android malware families,” *Logic Journal of the IGPL*, vol. 25, no. 1, pp. 54–66, 2017.
- [31] S. Singh and P. Gupta, “Comparative study id3, cart and c4. 5 decision tree algorithm: a survey,” *International Journal of Advanced Information Science and Technology*, vol. 27, no. 7, pp. 97–103, 2014.
- [32] W.-Y. Loh and Y.-S. Shih, “Split selection methods for classification trees,” *Statistica Sinica*, vol. 7, no. 4, pp. 815–840, 1997.
- [33] W.-Y. Loh, “Regression trees with unbiased variable selection and interaction detection,” *Statistica Sinica*, vol. 12, no. 2, pp. 361–386, 2002.
- [34] L. Li, A. Bartel, T. F. Bissyande et al., “IccTA: Detecting Inter-Component Privacy Leaks in Android Apps,” in *Proceedings of the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*, vol. 1, pp. 280–291, Florence, Italy, May 2015.
- [35] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, “Drebin: effective and explainable detection of android malware in your pocket,” in *Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium*, vol. 14, pp. 23–26, 2014.
- [36] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, “Droidsieve: Fast and accurate classification of obfuscated android malware,” in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (CODASPY)*, pp. 309–320, Scottsdale, Arizona, USA, 2017.

Research Article

Multiclass Classification Procedure for Detecting Attacks on MQTT-IoT Protocol

Hector Alaiz-Moreton ¹, Jose Avelaira-Mata ², Jorge Ondicol-Garcia,²
Angel Luis Muñoz-Castañeda,² Isaías García,¹ and Carmen Benavides¹

¹Escuela de Ingenierías, Universidad de León, 24071 León, Spain

²Research Institute of Applied Sciences in Cybersecurity (RIASC) MIC, Universidad de León, 24071 León, Spain

Correspondence should be addressed to Hector Alaiz-Moreton; hector.moreton@unileon.es

Received 27 November 2018; Accepted 10 February 2019; Published 7 April 2019

Guest Editor: Alicja Krzemień

Copyright © 2019 Hector Alaiz-Moreton et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The large number of sensors and actuators that make up the Internet of Things obliges these systems to use diverse technologies and protocols. This means that IoT networks are more heterogeneous than traditional networks. This gives rise to new challenges in cybersecurity to protect these systems and devices which are characterized by being connected continuously to the Internet. Intrusion detection systems (IDS) are used to protect IoT systems from the various anomalies and attacks at the network level. Intrusion Detection Systems (IDS) can be improved through machine learning techniques. Our work focuses on creating classification models that can feed an IDS using a dataset containing frames under attacks of an IoT system that uses the MQTT protocol. We have addressed two types of method for classifying the attacks, ensemble methods and deep learning models, more specifically recurrent networks with very satisfactory results.

1. Introduction

The “Internet of Things” (IoT) describes many different systems and devices that are constantly connected to Internet, giving information from their sensors or interacting with their actuators. By 2020 it is estimated that there will be 4.5 billion IoTs joining the Internet [1]. These devices have special features, such as a low computing capacity and the use specific lighter protocols. This makes IoT devices more efficient, smaller, and less energy consuming; however these low settings reduce their encryption capacity. These heterogeneous systems and networks offer new challenges in cybersecurity, such as new vulnerabilities and anomalies [2, 3]. One of the most important attacks in recent years, the Mirai botnet, exploited these vulnerabilities by carrying out distributed denial of service attacks infecting IoT devices and attacking with as many as 400,000 simultaneously connected devices [4].

One way of improving network security is the use of Intrusion Detection Systems (IDS). IDS are one of the most productive techniques for detecting attacks within a network.

This tool can detect network intrusions and network misuses by matching patterns of known attacks against ongoing network activity [5]. With this purpose, our focus is to develop an IDS with machine learning models for the IoT. IDS use two different detection methods: signature-based detection and anomaly-based detection. Signature-based detection methods are effective in detecting well-known attacks by inspecting network traffic for specific patterns. Anomaly-based detection systems identify attacks by monitoring the behaviour of the entire system, objects, or traffic and comparing them with a predefined normal status [6].

Machine learning techniques are used to improve detection methods, by creating new rules automatically for signature-based IDS or adapting the detection patterns of anomaly-based IDS. These anomaly-based IDS have had good results in qualifying frames that may be under attack [7], and they are effective even in detecting zero-day attacks [8].

To build a machine learning classifier it is necessary to use a dataset. Within the network intrusion detection there are some well-known datasets that are used to feed IDS

with machine learning techniques [9]. As there are no public datasets based on network traffic using IoT protocols, we have used a dataset that has been created in our previous research (the dataset is available in <https://joseaveleira.es/dataset.©®reg#LE-229-18>). The main focus of this paper is on the three different machine learning techniques that classify three different attacks and normal frames at the same time using our IoT environment dataset.

2. Related Work

There are several approaches for the detection of anomalies in traditional networks using machine learning. The most widely used datasets are the KDD99 [10] and NSL-KDD Dataset [11] (an improved version of KDD'99). These datasets contain traffic captured on the TCP protocol and collect different types of attacks. Based on these datasets, some models have been developed for anomaly detection using a Support Vector Machine and Random Forest [12, 13]. Another technique used on this dataset is K-Centroid clustering, whose objective is to improve the performance of other models [14]. There are also ways of upgrading these datasets, such as balancing classes to increase the models' prediction accuracy, which improves their performance [15].

Other detection techniques with good results are the use of Fuzziness based semisupervised learning getting an accuracy of 84 [16] and also obtaining good results analyzing the network traffic using sequential extreme learning machine with accuracies around 95 [17]. These good results indicate that machine learning is a good approach to improve the detection of intrusions in the network layer.

The machine learning methods are based on deep learning [18]. There are many approaches for solving anomaly detection using deep learning. One proposed way is to use the Deep Belief Network (DBN) as a feature selector on the KDD dataset, combined with a SVM that classifies the attacks [19, 20]. Another proposed method is to use deep learning models as feature selectors using the Fisher Score, a classical statistic method, combined with an autoencoder to reduce the dimensions of the data and extract the highest-valued features [21]. Deep learning models are used as classifiers too. Understanding that the temporary data sequence of network attacks is important, the Long Short Term Memory (LSTM) network, a variant of recurrent networks, has been used to classify the KDD's attacks [22].

As regards the IoT IDS, there is an approach that uses fog computing combined with mobile edge computing. Using this combination, a numerical simulation is made for the NSL-KDD dataset, where it has been demonstrated that this type of IDS has a good performance both in accuracy and time dependence [23]. Using this dataset, there is also an IDS based on rules which rules are modified using machine learning KNN and SVM techniques [24]. Because the research into IDS schemes for IoT is still incipient, the proposed solutions do not cover a wide range of attacks or IoT technologies [25].

There are other more recent datasets such as the AWID [26] which collects TCP frames of data from a WLAN network over which several attacks were made on 802.11 security

mechanism through which a study on Wi-Fi intrusions was made using a neural network classifier [27]; another current dataset is the CICIDS2017 [28] used to validate the detection algorithms on which training has been carried out with recurrent neural networks [29].

This research is based on a dataset specialized in a protocol implemented in IoT environments to detect specific vulnerabilities. It is specialized in an IoT protocol where does not exist dissection of traffic ready to use with research purposes.

3. Methods and Materials

This section describes the methods, from a theoretical point of view as well as the materials used for implementing the experiments.

3.1. MQTT Dataset. In order to classify anomalies in an IoT environment, we built a dataset using MQTT, which is a publish-subscribe-based messaging protocol. It is a light protocol widely used in IoT [30].

The MQTT's architecture follows a star topology, with a central node that functions as a server or broker. The broker is in charge of managing the network and transmitting. The communication is based on topics created by the client that publishes the message and the nodes that wish to receive it must subscribe to it. The communication can be one to one, or one to many.

This dataset has been obtained in a test environment with several sensors, actuators, and a server. This server hosts the management application, also working as the broker that manages the messages of the MQTT protocol. The scheme of the environment is detailed in the Figure 1.

We carried out several attacks against the MQTT protocol in the test environment. We captured these attacks at the network level along with all generated traffic. The attacks carried out were as follows:

- (i) DoS: denial of service is one of the most common attacks on the Internet [31]. In the case of the MQTT protocol, the broker is attacked by saturating it with a large number of messages per second and new connections. Using the MQTT-malaria program [32], this program is used for testing the scalability and load testing utilities for MQTT environments.
- (ii) Man in the middle (MitM) consists of intercepting the messages between two communication points in an attempt to modify the content; in this case it is done between a sensor and the broker by modifying the sensor data. To carry out the attack we used the distribution Kali Linux and the tool Ettercap.
- (iii) Intrusion: taking into account the characteristics of the MQTT protocol, this attack consists of using the well-known port (1883) for this protocol and a command that uses the special character “#” can be used by an external attacker for knowing the active topics available for being subscribed. To find out which topics a client outside the system [33].

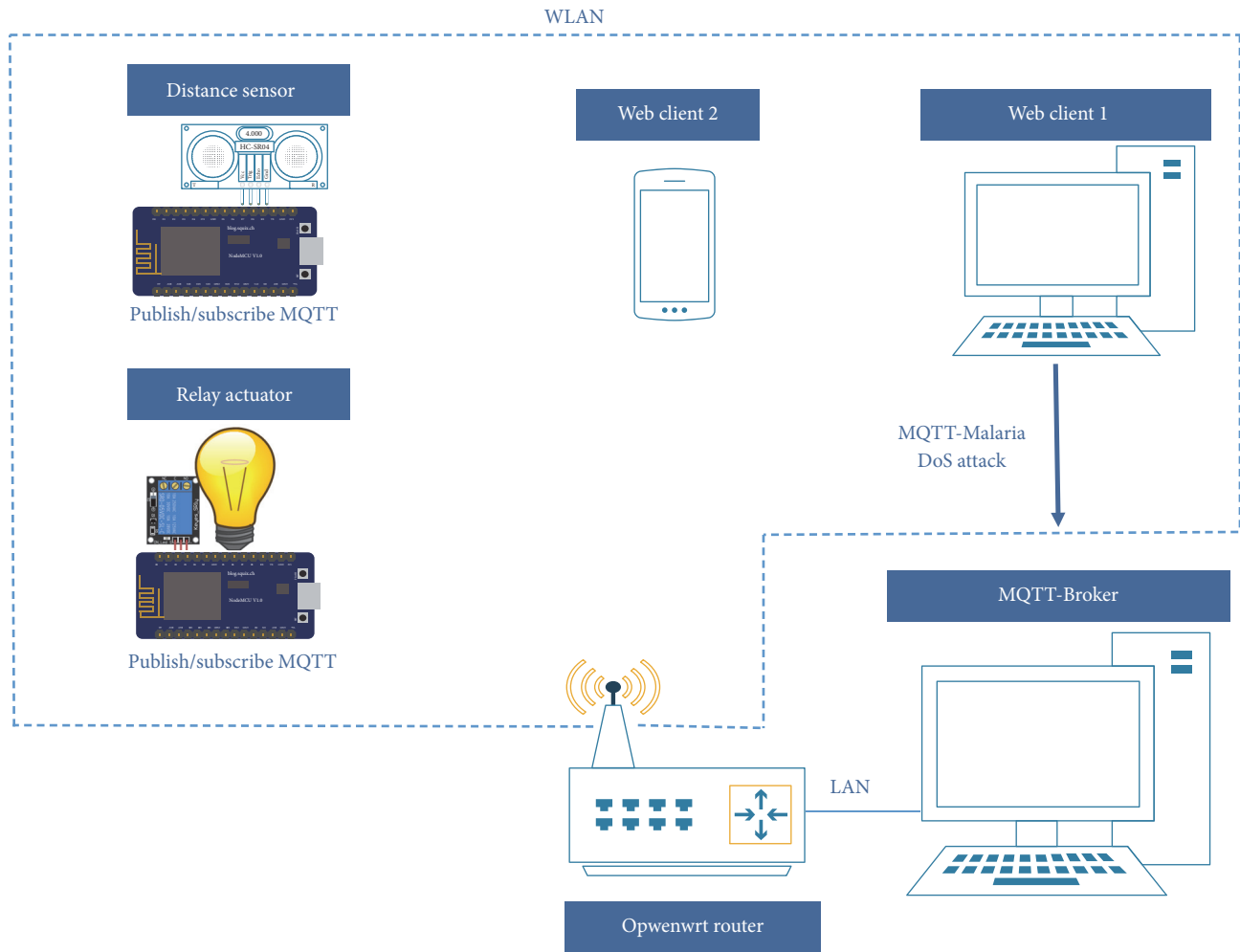


FIGURE 1: Test environment schema.

The relevant fields and the protocol are selected after capturing the network traffic in the system. All communication frames are tagged to show whether one of them is under attack or is normal. There are three CSV files generated, one for each attack, all of them being part of the dataset used. Selecting features and labeling the frames indicating whether or not they are under attack enable supervised learning techniques to be used on this dataset.

The features of the data set are as follows:

- (i) DoS.csv that contains the capture of 94.625 frames and of which 45.513 are under attack traffic and 49.112 are normal traffic.
- (ii) MitM.csv that contains 110668 frames with 3855 under man in the middle attack and 106.813 normal traffic frames.
- (iii) Intrusion.csv with 80,893 total frames with 1898 under attack and 78,995 normal traffic frames.

3.2. Classification Methods. We have chosen XGBoost because other research delivered good results like [34–36]. We have also chosen recurrent networks for our experiments

because of the importance of time in network attacks [22], as frames are produced sequentially, and the sequence and time between frames provide relevant information for detecting an attack. We shall go into our classification models in more detail in the following sections.

3.2.1. XGBoost (Gradient Boosting). Gradient boosting systems build additive models in a forward way through steps, allowing the optimization of arbitrary differentiable loss functions. In each forward step, regression trees are fitted onto the negative gradient of the binomial or multinomial loss function [37]. XGBoost stands for Extreme Gradient Boosting [38]. It is a scalable machine learning system for tree boosting which optimizes many systems and algorithms, such as a tree learning algorithm that handles sparse data, handling instance weights in approximate tree learning or exploiting out-of-core computation. For the implementation of this method, we are using the XGBoost library for Python [39].

3.2.2. Recurrent Neural Networks. Recurrent neural networks are a variant of neural networks designed for highly sequenced problems. RNN contain cycles that feed the

network activation from a previous time step as inputs into the network, influencing predictions at the current time step. The addition of cycles gives the RNN a new dimension, where instead of mapping only inputs to outputs, the network will learn a mapping function for the inputs to an output over time. One of the main disadvantages of this kind of network is the training problems, such as a vanishing gradient and exploding gradients [40]. These problems can be addressed through variations in the neurons, such as GRU or LSTM cells.

LSTM Recurrent Network. LSTM networks have a unique formulation that allows them to prevent the problems of scaling and training of the vanilla RNN, avoiding the back propagation error that either blows up or decays exponentially. An LSTM layer consists of a set of recurrently connected blocks, known as memory blocks that are the computational units of the LSTM network [41]. These cells are made up of weights and gates. Each memory block contains one or more recurrently connected memory cells and three multiplicative units: input, output, and forget gates. The gates allow the information flows to interact with the cells. The forget gate and the input gate update the internal state, while the output gate is the final limiter of the cells' output. These gates and the consistent data flow called CEC, or constant error carousel, keep each cell stable [42]. For the implementation of this network, we are using the Keras framework for Python with the Tensorflow backend, using GPU processing and an improved-performance version that uses Cudnn (CudnnLSTM) [43].

GRU Recurrent Network. Just like LSTM networks, GRU networks have a structure and formulation that improve the vanilla RNN. GRU was first proposed in Cho et al. [44] as an alternative to the LSTM to capture dependencies of different time scales adaptively. The only difference between these networks is the procedure for updating the CEC. It is similar to the LSTM but with one difference, the GRU units have no mechanism for controlling the exposure to which this data flow is submitted [45]. This lack of control mechanisms makes the GRU units faster than the LSTM and more adaptable to the changes in the time flow [46]. For the implementation of this network, we are using the Keras framework for Python with the Tensorflow backend, using GPU processing and an improved-performance version that uses Cudnn (CudnnGRU) [43].

3.3. Optimization Methods. SGD methods are iterative methods used for optimizing an objective function. Adam is based on adaptive estimates of lower-order moments. This method is simple to implement and computationally efficient and has few memory requirements. It does not change as a result of the diagonal rescaling of the gradients and works well for problems that are large in data and/or parameters. Adam is also appropriate for changing objectives and problems with very noisy and/or sparse gradients. The hyperparameters of this method are easy and intuitive to understand and usually require little tuning [47]. Based on Adam we have Nadam, which is an Adam version applying Nesterov momentum

[48]. We have tested RMSprop, Adam, and Nadam, all of which are stochastic gradient descent methods, but we got our best results using Nadam to optimize our loss. Nadam brings more speed in learning in each minibatch step. Nadam gave us better results because we have a complex net and fewer epochs. We needed a faster loss function optimizer to learn more in fewer epochs, without fearing a fast Decay into overfitting.

3.4. Batch Normalization. This method works by making normalization a part of the model architecture and carrying out a normalization step for each training minibatch. It addresses the problem of the internal covariate shift, brought about by the values of the input layers' changing during training. This problem requires low learning rates and a careful parameter initialization and makes it harder to train models with saturating nonlinearities. Batch Normalization allows us to use higher learning rates and pay less attention to initialization. It can also act as a regularizer, in some cases eliminating the need for other regularization techniques [49].

3.5. Evaluation Metrics. Metrics evaluate the performance of a machine learning model. Every metric measures the efficiency in a different way, so we use several metrics for our models in order to obtain a more accurate view.

3.5.1. Multiclass Logarithmic Loss and Categorical Cross Entropy. The logarithmic loss metric measures the performance of a classification model in which the prediction input is a probability value of between 0 and 1. Its formula is as follows:

$$-(y * \log(y_{pred}) + (1 - y) * \log(1 - y_{pred})) \quad (1)$$

where $y \in [0, 1]$ is the known label and $y_{pred} \in [0, 1]$ is the prediction of the model. Logarithmic loss and cross entropy in machine learning when calculating error rates of between 0 and 1 lead to the same thing. The cross-entropy formula is as follows:

$$H(p, q) = -\sum_x (p(x) * \log(q(x))) \quad (2)$$

If $p \in [y, 1 - y]$ and $q \in [y_{pred}, 1 - y_{pred}]$,

$$-(y * \log(y_{pred}) + (1 - y) * \log(1 - y_{pred})) \quad (3)$$

The same formula is applied in both situations. We can extend the logarithmic loss to multiclass problems, given the true labels of a set of samples encoded as a 1-of-K binary indicator matrix Y , where $y_{i,k} = 1$ if sample i has label k taken from a set of K labels. Let Y_{pred} be a matrix of probability estimates, with $ypred_{i,k} = Pr(t_{i,k} = 1)$:

$$L_{log}(Y, Y_{pred}) = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{K-1} y_{i,k} * \log(ypred_{i,k}) \quad (4)$$

3.5.2. *Multiclass Classification Error Rate.* The multiclass error rate is the percentage of misclassifications made by the model:

$$\frac{P_{\text{wrong}}}{P} \quad (5)$$

3.5.3. *F-Beta Score.* The F-beta score is the weighted harmonic average of precision and recall, obtaining its best value at 1 and its worst value at 0. The β parameter determines the weight of precision in the combined score. $\beta < 1$ lends more weight to precision, while $\beta > 1$ favors recall.

$$F_{\beta} = (1 + \beta^2) * \frac{\text{precision} * \text{recall}}{(\beta^2 * \text{precision}) + \text{recall}} \quad (6)$$

3.5.4. *Categorical Accuracy.* The calculation of the average accuracy rate across all predictions made for a multiclass problem is made using the following formula:

$$\frac{1}{N} \sum_{i=0}^{N-1} \text{Equals}(\text{argmax}(y), \text{argmax}(p)) \quad (7)$$

3.6. *Dropout.* Dropout is used to prevent overfitting. It works by randomly dropping units and their connections from the neural network during training. This prevents network units from adapting too much to a problem [50].

4. Experiments

Our experiments are based on three datasets, one for each attack. Before joining them, we balanced each one of them to reduce the huge differences between all of the classes. We balance the classes of each dataset using the resample method provided by Scikit-learn [51]. Once all of the datasets were balanced, we put them together to build a multiclass dataset. With the complete dataset ready, we chose the most representative features using a Feature Importance (FIM) report system. Our FIM algorithm is a hybrid method based on the mutual information function and it is composed by two routines; one corresponding to a filter process (based on the minimum-redundancy-maximum-relevance) and another corresponding to a wrapper process, where we used several models like SVM, Decision Trees, or Random Forests. This method confronts each feature of the dataset against the target feature. Choosing the highest values gives us the most important features for each set of data. After that, we confront each of the variables chosen in pairs between them and then we delete the highest-valued features to decrease redundancy. We also have to prepare our custom metric F-beta score. Taking into account the F-beta formula presented on the paper, we select $\beta = 1$ to increase the value of the recall variable. The recall is the amount of data well classified in both parts, referring to the amount of present false positives and negatives. The classifications problems, in networks specifically, have many problems with false positives and negatives, so giving more value to the metric can make it more sensitive to these failures and could give us

```
...
le_Msg = LabelEncoder()
dataset_combined['mqtt.msg'] = le_Msg.fit_transform(
dataset_combined['mqtt.msg'].astype(str))
...
```

Box 1

```
input_timesteps = 3
features = 11
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
#Three timesteps plus the actual one
X_train = X_train.reshape(
(X_train.shape[0], input_timesteps+1, features)
X_test = X_test.reshape(
(X_test.shape[0], input_timesteps+1, features)
```

Box 2

a more accurate vision. Immediately after that, we prepare the categorical values in order to make it possible to train both recurrent neural networks Box 1.

Finally, we set four timesteps for both recurrent LSTM and GRU networks, transforming the inputs into tensors made up of samples, timesteps, and features Box 2.

A hyperparameter search on recurrent networks is computationally expensive, so we have chosen their hyperparameters depending on logs of training, increasing the width and length of the network or increasing the periods if it is underfitted or by applying a batch normalization, dropout or reducing the length, width, or epochs to reduce overfitting. Now, we will describe our three different classification methods in greater detail.

4.1. *XGBoost.* We define the XGBoost model for our problem, highlighting the four types that we wish to classify and specifying both the tree method and the booster. We use a version of the tree method called the XGBoost fast-histogram algorithm. This method is much faster and uses considerably less memory than other methods [52], but it needs a specific version of CUDA to work. We also highlighted the evaluation metrics that we wanted to use (multiclass logarithmic loss and multiclass classification error rate), the number of threads and the number of estimators in the model. We have applied a grid search with a threefold cross validation to take the best parameters of the model. These are the parameters we wish to tune in Box 3.

This grid search gives us a set of parameters that perform best on the problem in Box 4.

4.2. *Recurrent LSTM.* For our LSTM network, we first compile some of the parameters of the net, setting the loss, the optimizer, and the metrics. Our loss is a variant

TABLE 1: Results of evaluation metrics for XGBoost.

Model	M. logarithmic loss	M. classification error rate
XGBoost-Train	0.075348	0.024753
XGBoost-Test	0.079451	0.025651

```
param = ['max-depth': [1, 5, 10, 20, 25],
'learning-rate': [0.4, 0.6, 0.8],
'min-child-weight': [1, 5, 10],
'gamma': [0.5, 1, 1.5, 2, 5],
'sub-sample': [0.6, 0.8, 1.0],
'col-sample-by-tree': [0.6, 0.8, 1.0]]
```

Box 3

```
'learning-rate': 0.4,
'gamma': 0.5,
'min-child-weight': 10,
'col-sample-by-tree': 1.0,
'max-depth': 5,
'sub-sample': 0.8
```

Box 4

```
model.compile(loss='categorical_crossentropy',
optimizer='Nadam',
metrics=[metrics.categorical_accuracy, fbeta])
```

Box 5

```
history = model.fit(X_train, y_train, batch_size=128,
validation_split=0.1, epochs=15,
verbose=2, callbacks=[tb.LOG])
```

Box 6

of cross entropy for multiclassification called categorical cross entropy. We are using the unmodified version of the Nadam optimizer. We tested Adam, RMSprop, and Nadam, establishing that Nadam is more efficient than Adam and RMSprop for our model. We set the metrics categorical accuracy and f -score to measure the model's accuracy and reliability, setting β parameter for Fbeta-score metric at 2 in Box 5.

We fit the model with our data, using a batch size of 128 and 15 epochs. We use a 10% validation split to validate the results of the training in Box 6.

We use an Encoder-Decoder approach for our LSTM network. We also use a CUDA version of the LSTM cell from

the Keras library [43]. In order to avoid overfitting, we used dropout, setting its value between 0.3 and 0.4 depending on the size of the previous layers, and batch normalization to control exploding gradients and speed up the training process in Box 7.

4.3. Recurrent GRU. For our GRU network, we compile the parameters of the net, setting the loss, the optimizer, and the metrics. We have set some of the GRU net parameters similar to our LSTM net. Our loss is a variant of cross entropy for multiclassification called categorical cross entropy. We are using the unmodified version of the Nadam optimizer. As we did on LSTM, we tested both Adam and Nadam, finding that Nadam is more efficient than Adam for our model. We set the categorical accuracy and f -score of the metrics to measure the model's accuracy and reliability, setting β parameter for F-beta-score metric at 2 in Box 8.

We fit the model with our data, using a batch size of 256 and 17 epochs. We use a 10% validation split to validate the results of the training in Box 9.

We use a linear structure approach for our GRU network. We also used a CUDA version of GRU cell from the Keras library [43]. In order to avoid overfitting, we used dropout, setting its value between 0.2 and 0.3 depending on the size of the previous layers, and batch normalization to control exploding gradients and speed up the training process. Because of the GRU cell design, it needs more time to learn than LSTM, although it is faster. This also affects dealing with the overfitting, needing fewer dropout values for the GRU net in Box 10.

5. Results and Discussion

Once we had the results of the search for the XGBoost, we trained the model and tested it on our data, with the following results as detailed in Figure 2 and Table 1.

Our LSTM and GRU networks gave us the following results for training and validation (Figures 3, 4, 5, 6 and Table 2).

In our previous work, the ensemble methods gave us better accuracies than the linear methods on the three datasets separately; i.e., for DoS, the best accuracy achieved was 0.99377 using a random forest model and Boosting Gradient achieved 0.99373, while SVM achieved 0.99023, taking into account the best two models and the worst. The difference was smaller for DoS, but on intrusion and MitM the difference between these two types of method was higher. Specifically, on intrusion Random Forest and Boosting Gradient they got 0.95294 and 0.95385, respectively, while SVM got 0.93031. The results were similar for intrusion. In our experiments on this paper, XGBoost achieved the highest accuracy. This result confirms that ensemble methods achieve higher accuracies


```

...
model.add(CuDNNLSTM(128, return_sequences=True))
model.add(BatchNormalization())
model.add(Dropout(0.3))
model.add(CuDNNLSTM(128, return_sequences=True))
model.add(BatchNormalization())
model.add(CuDNNLSTM(256, return_sequences=True))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(CuDNNLSTM(256, return_sequences=True))
model.add(BatchNormalization())
...

```

Box 7

TABLE 2: Results of evaluation metrics for LSTM and GRU.

Model	Categorical Cross Entropy	Categorical Accuracy	F-beta Score
LSTM-Train	0.2093	0.9276	0.9148
GRU-Train	0.1334	0.9554618	0.952418
LSTM-Validation	0.1821	0.9337	0.9328
GRU-Validation	0.1280	0.960836	0.95777

```

model.compile(loss='categorical_crossentropy',
optimizer='Nadam',
metrics=[metrics.categorical_accuracy, fbeta])

```

Box 8

```

history = model.fit(X_train, y_train, batch_size=256,
validation_split=0.1, epochs=18,
verbose=2, callbacks=[tb.LOG])

```

Box 9

and less loss than other linear models or neural networks such as SVM, GRU, and LSTM for problems involving attacks on IoT networks.

Multiclass classification problems tend to be more complex than binary problems, making getting better results harder for these problems. We had similar results in both experiments on ensemble models when classifying, where we maintain the highest metrics and results. Focusing on our GRU and LSTM models, we had better results overall using deep learning than using linear models, but we had worse results than ensembles. LSTM got worse result than the SVM's DoS model and slightly better results than the SVM model for intrusion and MitM. GRU performed better, getting worse results than the SVM's DoS, but better than SVM for intrusion and MitM.

Even though we dealt with imbalance, there are still huge differences between classes. This may have affected the accuracy in some of our models negatively, specifically the GRU and LSTM. We have been able to maintain a good result by taking the sequencing of the problem into account.

6. Conclusion

IoT systems have been growing in recent years and are expected to increase considerably. The special features of these devices make the network technologies more heterogeneous than traditional networks, presenting new challenges to cybersecurity. Taking into account the fact that IDS are an important security barrier that can detect intrusions and security risks in the network quickly, we propose models for the detection of attacks in IoT environments that can provide an IDS oriented for IoT. We use specific datasets with particular attacks for these systems, specifically for the MQTT protocol. In this case, machine learning techniques can be used to classify the frames that an IDS can assign as attack or normal. We chose the LSTM, GRU, and XGBoost models for our classification problem. We selected these recurrent models because of the importance of time and sequencing in network attacks. We picked XGBoost because the structure of the problem benefits the hierarchical ensemble method's performance, enabling them to achieve the highest accuracies. All these three classification methods are very efficient, with GPU implementations. Ensemble methods obtained the highest results, and deep learning models achieved better results in general than linear models, but not as good as ensemble methods. These models can be used for future work in which an IDS is fed with a model. This IDS will be implemented in a standard computer

```
...  
model.add(CuDNNGRU(128, return_sequences=True))  
model.add(BatchNormalization())  
model.add(Dropout(0.2))  
model.add(CuDNNGRU(256, return_sequences=True))  
model.add(BatchNormalization())  
model.add(CuDNNGRU(256, return_sequences=True))  
model.add(BatchNormalization())  
model.add(Dropout(0.3))  
model.add(CuDNNGRU(256, return_sequences=True))  
model.add(BatchNormalization())  
...
```

Box 10

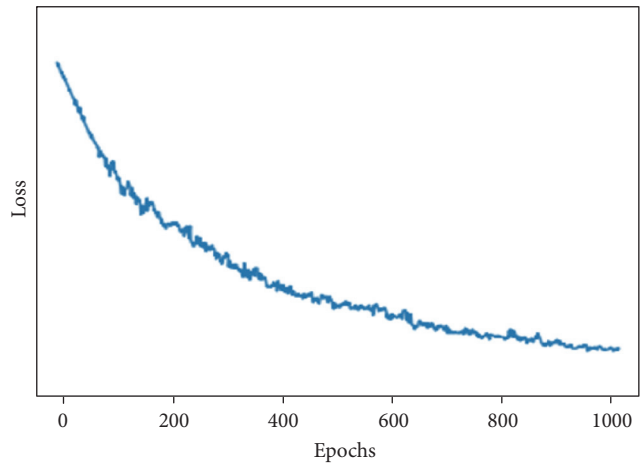
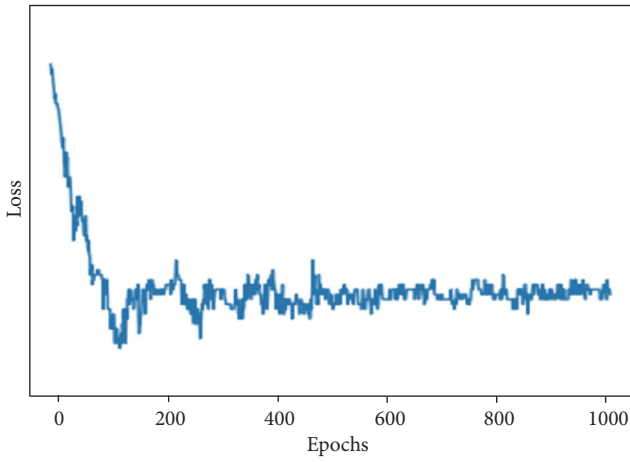


FIGURE 2: Training and testing graphics for XGBoost.

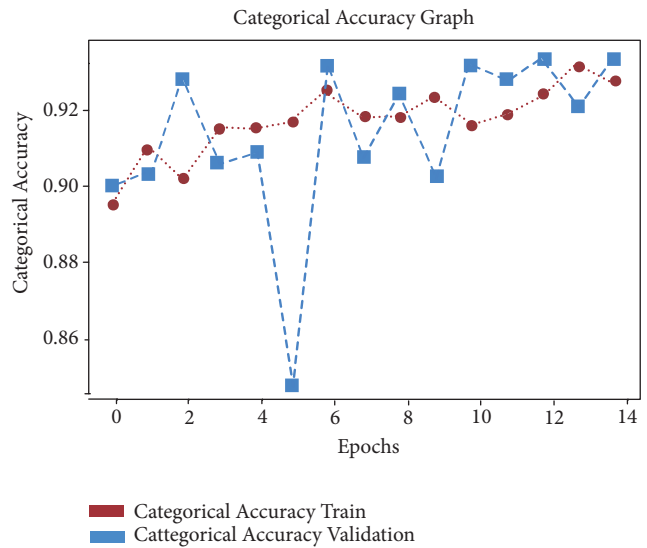
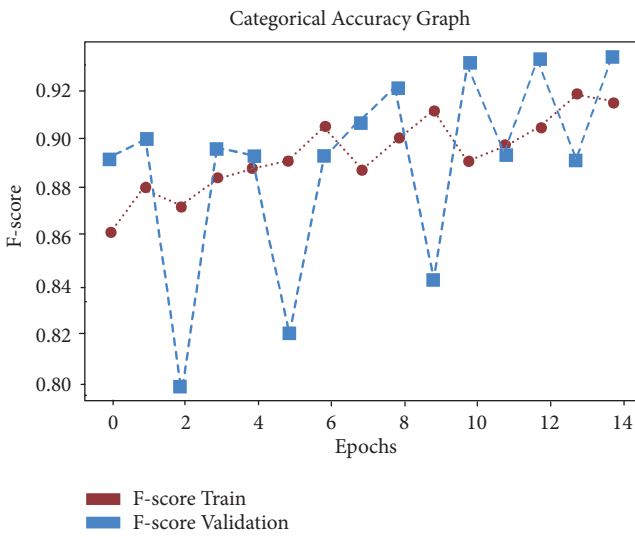


FIGURE 3: F-beta score and categorical accuracy LSTM.

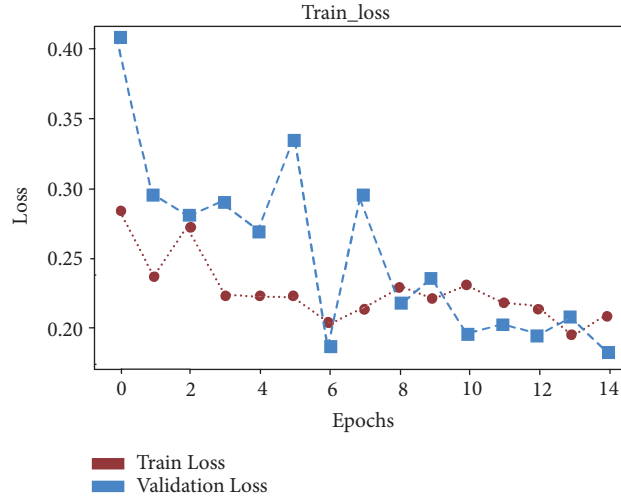


FIGURE 4: Categorical cross-entropy LSTM.

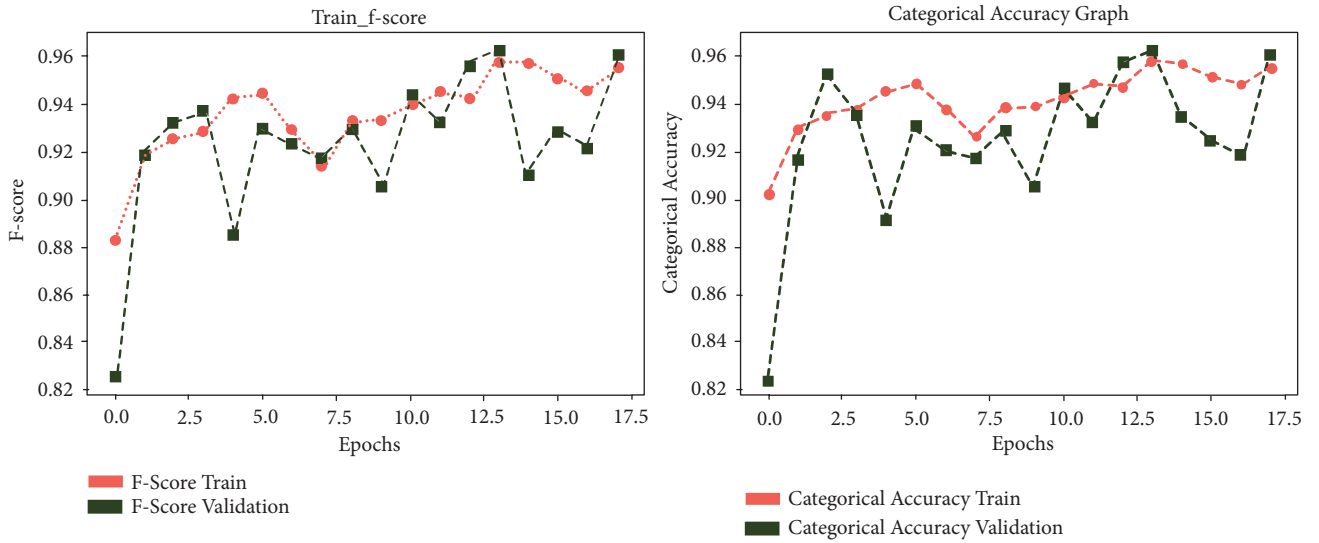


FIGURE 5: F-beta score and categorical accuracy GRU.

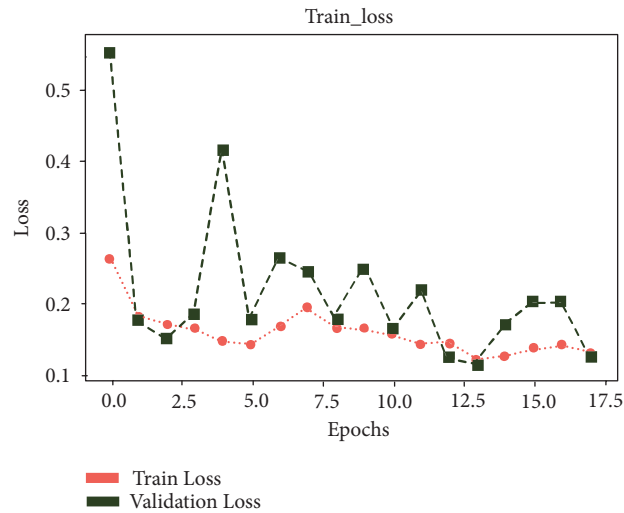


FIGURE 6: Categorical cross-entropy GRU.

similar to the one use for the creation of machine and deep learning models. Thus, the Python model will be deployed in a standard unit with a Port Mirroring from the router.

Data Availability

The dataset used to support the findings of this study is available in <https://joseaveleira.es/dataset>. © reg#LE-229-18.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by (i) Instituto Nacional de Ciberseguridad (INCIBE) and developed Research Institute of Applied Sciences in Cybersecurity (RIASC); (ii) Junta de Castilla y León, Consejería de Educación, Project LE078G18. UXXI2018/000149. U-220.

References

- [1] J. Green, "The Internet of Things Reference Model," *Internet of Things World Forum*, pp. 1–12, 2014.
- [2] M. A. Razzaq, S. H. Gill, M. A. Qureshi, and S. Ullah, "Security Issues in the Internet of Things (IoT): A Comprehensive Study," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, p. 383, 2017.
- [3] C. Ordóñez Galán, F. Sánchez Lasheras, F. J. de Cos Juez, and A. Bernardo Sánchez, "Missing data imputation of questionnaires by means of genetic algorithms with different fitness functions," *Journal of Computational and Applied Mathematics*, vol. 311, pp. 704–717, 2017.
- [4] C. Koliás, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: mirai and other botnets," *IEEE Computer Society*, vol. 50, no. 7, pp. 80–84, 2017.
- [5] N. Ben-Asher and C. Gonzalez, "Effects of cyber security knowledge on attack detection," in *Computers in Human Behavior* 48, , 5161. doi:10.1016/j.chb.01.039, URL <http://dx.doi.org/10.1016/j.chb.2015.01.039>, 2015, <http://dx.doi.org/10.1016/j.chb.2015.01.039>.
- [6] K. Prabha and S. Sudha, "A Survey on IPS Methods and Techniques," in *Proceedings of the International Journal of Computer Science Issues* 13 (2, vol. 13, pp. 38–43, 2016, <http://ijcsi.org/contents.php>.
- [7] T. Hamed, J. B. Ernst, and S. C. Kremer, "A Survey and Taxonomy of Classifiers of Intrusion Detection Systems," in *Proceedings of the URL*, vol. 39, p. 21, Cham, 2018.
- [8] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "McPAD: a multiple classifier system for accurate payload-based anomaly detection," *Computer Networks*, vol. 53, no. 6, pp. 864–881, 2009.
- [9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection," *International Journal of Network Security*, vol. 17, no. 6, pp. 683–701, 2015.
- [10] S. J. Stolfo, *KDD cup*, URL <http://kdd.ics.uci.edu/>, 1999, <http://kdd.ics.uci.edu/>.
- [11] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2nd IEEE Symposium on Computational Intelligence for Security and Defence Applications*, pp. 1–6, IEEE, July 2009.
- [12] M. Hasan, M. Nasser, B. Pal, and S. Ahmad, "Support Vector Machine and Random Forest Modeling for Intrusion Detection System (IDS)," *Journal of Intelligent Learning Systems and Applications*, Article ID 9601233, pp. 45–52, 2014, http://file.scirp.org/Html/5-9601233_42869.htm.
- [13] P. G. Nieto, J. A. Fernández, F. S. Lasheras, F. de Cos, and C. D. Juez, "A new improved study of cyanotoxins presence from experimental cyanobacteria concentrations in the trasona reservoir (northern Spain) using the mars technique," *Science of the total environment*, vol. 430, pp. 88–92, 2012.
- [14] B. Chakrabarty, O. Chanda, and M. Saiful, "Anomaly based intrusion detection system using genetic algorithm and K-centroid clustering," *International Journal of Computer Applications*, vol. 163, no. 11, pp. 13–17, 2017.
- [15] J.-h. Seo and Y.-h. Kim, Machine-Learning Approach to Optimize SMOTE Ratio in Class Imbalance Dataset for Intrusion Detection, 2018.
- [16] R. A. R. Ashfaq, X. Z. Wang, J. Z. Huang, H. Abbas, and Y. L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484–497, 2017.
- [17] R. Singh, H. Kumar, and R. K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8609–8624, 2015.
- [18] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*.
- [19] Y. Li, R. Ma, and R. Jiao, "A Hybrid Malicious Code Detection Method based on Deep Learning," *International Journal of Security and Its Applications*, vol. 9, no. 5, pp. 205–216, 2015.
- [20] P. García Nieto, J. Martínez Torres, F. de Cos Juez, and F. Sánchez Lasheras, "Using multivariate adaptive regression splines and multilayer perceptron networks to evaluate paper manufactured using *Eucalyptus globulus*," *Applied Mathematics and Computation*, vol. 219, no. 2, pp. 755–763, 2012.
- [21] X. Tao, D. Kong, Y. Wei, and Y. Wang, "A Big Network Traffic Data Fusion Approach Based on Fisher and Deep Auto-Encoder," *Information*, vol. 7, no. 2, p. 20, 2016.
- [22] J. Kim, H. L. T. Thu, and H. Kim, "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection," in *Proceedings of the International Conference on Platform Technology and Service (PlatCon, 2016)*, pp. 1–5, 2016, <http://ieeexplore.ieee.org/document/7456805/>.
- [23] Xingshuo An, Xianwei Zhou, Xing Lü, Fuhong Lin, and Lei Yang, "Sample Selected Extreme Learning Machine Based Intrusion Detection in Fog Computing and MEC," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 7472095, 10 pages, 2018.
- [24] F. Lu and L. Wang, "Intrusion Detection System Based on Integration of Neural Network for Wireless Sensor Network," *Journal of Software Engineering*, vol. 8, no. 4, pp. 225–238, 2014.
- [25] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.
- [26] C. Koliás, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, 2016.

- [27] M. E. Aminanto, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Wi-Fi intrusion detection using weighted-feature selection for neural networks classifier," in *Proceedings of the 2017 International Workshop on Big Data and Information Security (IWBIS)*, pp. 99–104, Jakarta, September 2017.
- [28] ids-2017@www.unb.ca, <https://www.unb.ca/cic/datasets/ids-2017.html>, 2017.
- [29] Z.-Q. Qin, X.-K. Ma, and Y.-J. Wang, "Attentional Payload Anomaly Detector for Web Applications," in *Proceedings of the Neural Information Processing*, L. Cheng, A. C. S. Leung, and S. Ozawa, Eds., Springer, pp. 588–599, Cham, Switzerland, 2018.
- [30] P. Sethi and S. R. Sarangi, *Internet of Things: Architectures, Protocols, and Applications*, 2017.
- [31] T. Halabi and M. Bellaiche, "How to evaluate the defense against dos and ddos attacks in cloud computing: a survey and taxonomy," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 14, no. 12, pp. 1–10, 2016.
- [32] K. Palsson, [mqtt-malaria@github.com](https://github.com/remakeelectric/mqtt-malaria) (2018). <https://github.com/remakeelectric/mqtt-malaria>.
- [33] S. Andy, B. Rahardjo, and B. Hanindhito, "Attack Scenarios and Security Analysis of MQTT Communication Protocol in IoT System," in *Proceedings of the 4th International Conference on Electrical Engineering, Computer Science and Informatics, EECSI 2017*, pp. 19–21, IEEE, Yogyakarta, Indonesia, 2017.
- [34] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, CODASPY '16*, pp. 183–194, ACM, New York, NY, USA, 2016, <http://doi.acm.org/10.1145/2857705.2857713>.
- [35] D. Barradas, N. Santos, and L. Rodrigues, "Effective detection of multimedia protocol tunneling using machine learning," in *Proceedings of the 27th USENIX Security Symposium (USENIX Security 18)*, pp. 169–185, USENIX Association, Baltimore, MD, USA, 2018, <https://www.usenix.org/conference/usenixsecurity18/presentation/barradas>.
- [36] Y. Meidan, M. Bohadana, A. Shabtai et al., "Profiliot: A machine learning approach for iot device identification based on network traffic analysis," in *Symposium on Applied Computing, SAC '17*, pp. 506–509, ACM, New York, NY, USA, 2017, <http://doi.acm.org/10.1145/3019612>.doi:10.1145/3019612.
- [37] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [38] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016*, pp. 785–794, 2016, <https://arxiv.org/abs/1603.02754>.
- [39] XGBoost library, <https://xgboost.readthedocs.io/en/latest/tutorials/index.html>.
- [40] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training Recurrent Neural Networks," <https://arxiv.org/abs/1211.5063>, 2012.
- [41] T. U. Munich, "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures," *Neural Networks*, vol. 18, no. 5, Article ID 7647316, pp. 602–610, 2004.
- [42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [43] F. Chollet et al., "Keras: The Python Deep Learning library," <https://keras.io>.
- [44] K. Cho, B. van Merriënboer, C. Gulcehre et al., "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," <https://arxiv.org/abs/1406.1078>, 2014.
- [45] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," <https://arxiv.org/abs/1412.3555>, p. 1–9, 2014.
- [46] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," <https://arxiv.org/abs/1409.1259>, 2014.
- [47] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, <https://arxiv.org/abs/1412.6980v8>, p. 1-5, 2015.
- [48] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 8609–8613, May 2013.
- [49] S. Ioffe and C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, <http://arxiv.org/abs/1502.03167>.
- [50] A. Agarwal, S. Negahban, and M. J. Wainwright, "Noisy matrix decomposition via convex relaxation: Optimal rates in high dimensions," *The Annals of Statistics*, vol. 40, no. 2, pp. 1171–1197, 2012.
- [51] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research (JMLR)*, vol. 12, pp. 2825–2830, 2011.
- [52] R. Mitchell and E. Frank, "Accelerating the XGBoost algorithm using GPU computing," *PeerJ Computer Science*, vol. 3, 2017, <https://peerj.com/articles/cs-127>.

Research Article

Detection of Jihadism in Social Networks Using Big Data Techniques Supported by Graphs and Fuzzy Clustering

Cristina Sánchez-Rebollo,¹ Cristina Puente ,¹ Rafael Palacios ,¹ Claudia Piriz,² Juan P. Fuentes,² and Javier Jarauta²

¹Universidad Pontificia Comillas, 28015 Madrid, Spain

²Grupo SIA, Alcorcón, 28922 Madrid, Spain

Correspondence should be addressed to Cristina Puente; cristina.puente@comillas.edu

Received 7 December 2018; Accepted 11 February 2019; Published 10 March 2019

Guest Editor: Alicja Krzemień

Copyright © 2019 Cristina Sánchez-Rebollo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Social networks are being used by terrorist organizations to distribute messages with the intention of influencing people and recruiting new members. The research presented in this paper focuses on the analysis of Twitter messages to detect the leaders orchestrating terrorist networks and their followers. A big data architecture is proposed to analyze messages in real time in order to classify users according to different parameters like level of activity, the ability to influence other users, and the contents of their messages. Graphs have been used to analyze how the messages propagate through the network, and this involves a study of the followers based on retweets and general impact on other users. Then, fuzzy clustering techniques were used to classify users in profiles, with the advantage over other classifications techniques of providing a probability for each profile instead of a binary categorization. Algorithms were tested using public database from Kaggle and other Twitter extraction techniques. The resulting profiles detected automatically by the system were manually analyzed, and the parameters that describe each profile correspond to the type of information that any expert may expect. Future applications are not limited to detecting terrorist activism. Human resources departments can apply the power of profile identification to automatically classify candidates, security teams can detect undesirable clients in the financial or insurance sectors, and immigration officers can extract additional insights with these techniques.

1. Introduction

Social networks are playing a very important role in the way people think. When accurately targeted, repeated messages can reinforce political ideas or even flip the way of thinking of the most indecisive. In this regard, Jihadism has been identified as one of the movements that relies the most on social networks to spread propaganda and try to influence the public opinion. The Madrid bombings of 2004 [1] are used as a case study, where they analyze grassroot jihadist networks and how terrorist organizations use collective action from local level to cause enormous impact.

Social networks are also used by terrorist organizations as a tool for recruiting new members. Sentiment analysis to detect radicalization has been applied to social networks in the past as an evolution of previous analysis that were traditionally focused on websites and forums [2]. The problem of

nodes that play an important role as influencers or that spread propaganda and the way in which it is propagated is a growing area of research [3–5].

A very challenging part of the analysis presented in this paper is how to measure the impact of each user in the network, as it depends on the volume of tweets (activity) combined with the number of followers but is also amplified by the number of retweets. For this purpose, a deep analysis is carried out using graphs. General theory of networks and graphs, in particular, have been used for social network analysis (SNA) as one of the most relevant tools [6].

Labelling users as influencers, followers, or neutrals is also very difficult and false positives or false negatives of a standard classifier may yield dramatic consequences. Missing data can be corrected by applying genetic algorithms able to predict the absent text, as in the case of missing answers in questionnaires [7]. However, in the current research the

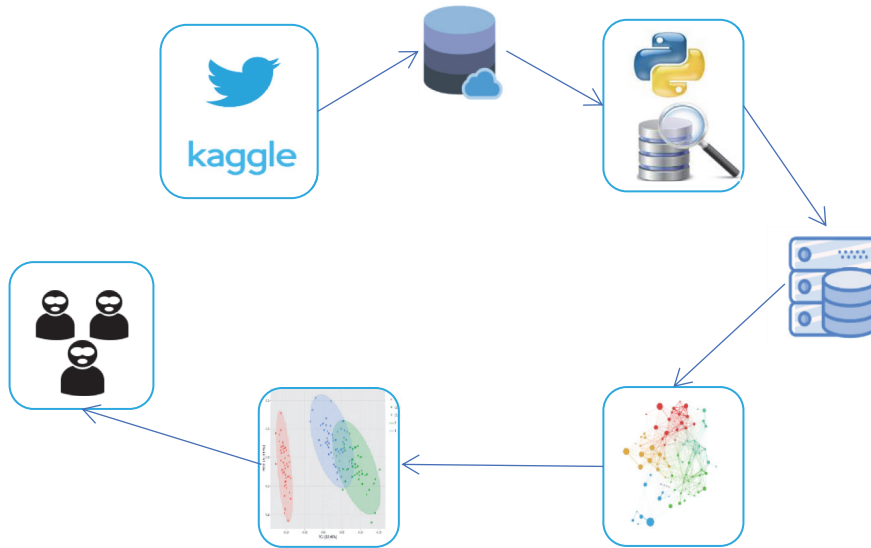


FIGURE 1: Stages of the procedure to select hidden Twitter profiles.

problem is more related to ambiguity as a result of unreliable data or contradictory terms in the messages. In this case advanced text mining or natural language processing techniques would be appropriate [8–10]. Nevertheless, due to the fact that the messages to be analyzed are mostly translations from Arabic language and dialects to English, it was decided not to put a big amount of effort into natural language, because of the risk of ending up modeling the translation process more than the original meaning of the messages.

For the purpose of assigning profiles to the users, the proposed methodology utilizes fuzzy clustering techniques that provide probability of classification for each possible profile. Fuzzy clustering has been successfully applied in semisupervised environments [11], in combination with the classic k-means clustering method [12], and more specifically to detect malicious components [13]. In this paper the fuzzy clustering method takes as an input the results obtained from the graph analysis, along with some characteristics directly extracted from the social network.

2. Description of the Methodology: Architecture Based on Graphs and Fuzzy Clustering

2.1. Big Data Architecture. A big data architecture is proposed with the goal of monitoring Twitter in real time being able to predict threats either by detecting changes in the profiles, or by detecting changes in the level of activity. The system can retrain itself to update profiles and classifications patterns, while maintaining its detection capabilities. A big data approach is very suitable for this kind of real-time analysis, especially on social networks such as Twitter in which messages are generated continuously and the system must collect, analyze, and archive-or-discard them [14, 15].

The proposed implementation was simulated using Kaggle’s databases plus Twitter extraction API for demonstration

purposes and to refine the algorithms based on graphs and fuzzy clustering (as shown on Figure 1).

2.2. Fuzzy Architecture to Isolate Suspicious Profiles. Using the previous works as inspiration and [13], we have designed a system capable of locating those profiles hidden at first sight but prone to modify their behaviour based on the influence received. In order to do so, we have considered a set of tweets as the source of information to measure the impact of an influence user in others.

Our process has followed five steps. First, the information was acquired using Kaggle’s database and Twitter user’s accounts by extracting their tweets. Then, that information was filtered to select the most and the less active users (understanding active users as those that both generated information or actively retweeted information from others). Next, we established those parameters with the potential to differentiate users, like sentiment analysis of the messages, users followed, and some others. Next the network graph was created, using the relationships among users to apply centrality measures in order to obtain new parameters that will serve as additional inputs to the fuzzy clustering process. Finally, the system will show those hidden users with an undefined profile, susceptible to be traced in the future. Figure 1 represents the interaction of these five stages.

2.3. Tweet Extraction Techniques. We have used several sources of information to get as many profiles as possible to perform the study. As primary source, we have used Twitter and Kaggle and, as secondary sources, several forums with terrorist ideologies and the official website of ISIS, known as Wafa Media Foundation, as seen in Figure 2. Primary sources were to collect the base information to be studied, to define the profile of the users actively spreading terrorist content while secondary sources were used to get familiar with the vocabulary used by these users in social propaganda. Many

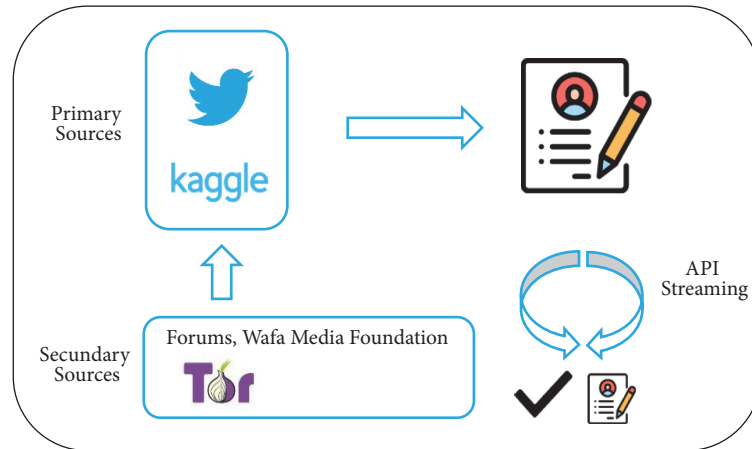


FIGURE 2: Tweet extraction sources.

keywords were obtained in this analysis to perform Twitter searches in a second step.

To download tweets from users speaking of terrorism (in favour and against), a connection between API Rest of Twitter and API Streaming was established. API Rest was used to check if the users whose information is being extracted had active accounts. API Streaming was used to download new users in real time and so expand the knowledge base about terrorists or potential targets. Although previous studies on network evolution show that social networks properties tend to reach an equilibrium [16], studies focused directly on terrorism are able to detect new trends and platform changes [17].

This way, we have expanded the suspicious profiles that we had from the initial databases “Isis_fanboy” and “About_isis” (Kaggle), with profiles that follow and spread the information and users included in these databases. The probability of obtaining users and repeated content is very high, as many times downloads belonged to followers of the downloaded users retweeting the same content (friends of friends sharing the same news, opinions and so). This was understood a clue to consider that we were searching information in the right direction.

This set of data was already registered and classified by the level of risk and continued downloading information associated, indicating the existence of active accounts, which allowed us to continue researching about it.

The extraction and analysis were focused on a social circle, in which we defined as “popular users” those with highest number of followers + publications + impact, serving as basis to categorize others that we already had. For this purpose, we retrieved several fields as

- (i) Location is one attribute that we retrieved but have not been used, as we have not considered it as relevant information. In Twitter, as in many other social networks, location can be removed, or even faked.
- (ii) Tweet_ID is the identification of a user on Twitter, though in this analysis we have not taken it into account as we have the username of the profile.

- (iii) Time: date and time of the tweet. We used it to measure the periods of higher activity in an interval of five months. Once those maximums were located, we contrasted the dates to check if those days had any correlation with some political, social, or economic ISIS event.

With this information, we built the database shown in Figure 3.

2.4. Information Preprocessing. With the previous fields, we have discarded the fields name (many times is fake) and location (many times undetermined), using the rest to generate new knowledge. In particular the following variables were introduced:

- (i) Frequency: using the field date, to calculate the interval within the user sends a tweet.
- (ii) Sentiment: analysis of the sentiment of tweets of each user to polarize them in positives and negatives. For this analysis the Python library nltk and Vader Lexicon were used, specialized in natural language processing.
- (iii) Extraction from each tweet the mentioned or retweeted users, by using regular expressions and Python.

From this new dataset, a new filter was applied in order to locate those active users generating tweets and named or retweeted by other users (most impact users). From this set of users, the set of connections by means of a graph was computed.

2.5. Graphing the Network. We have created several graphs based on different metrics that would lead to different interpretations. The objective was to analyze and visualize social relationships among users and communities.

Once the graph was created, as the one displayed in Figure 4, we have applied indicators of centrality to identify the most important vertices based on several criterions, to detect as well as the most influential users those who receive

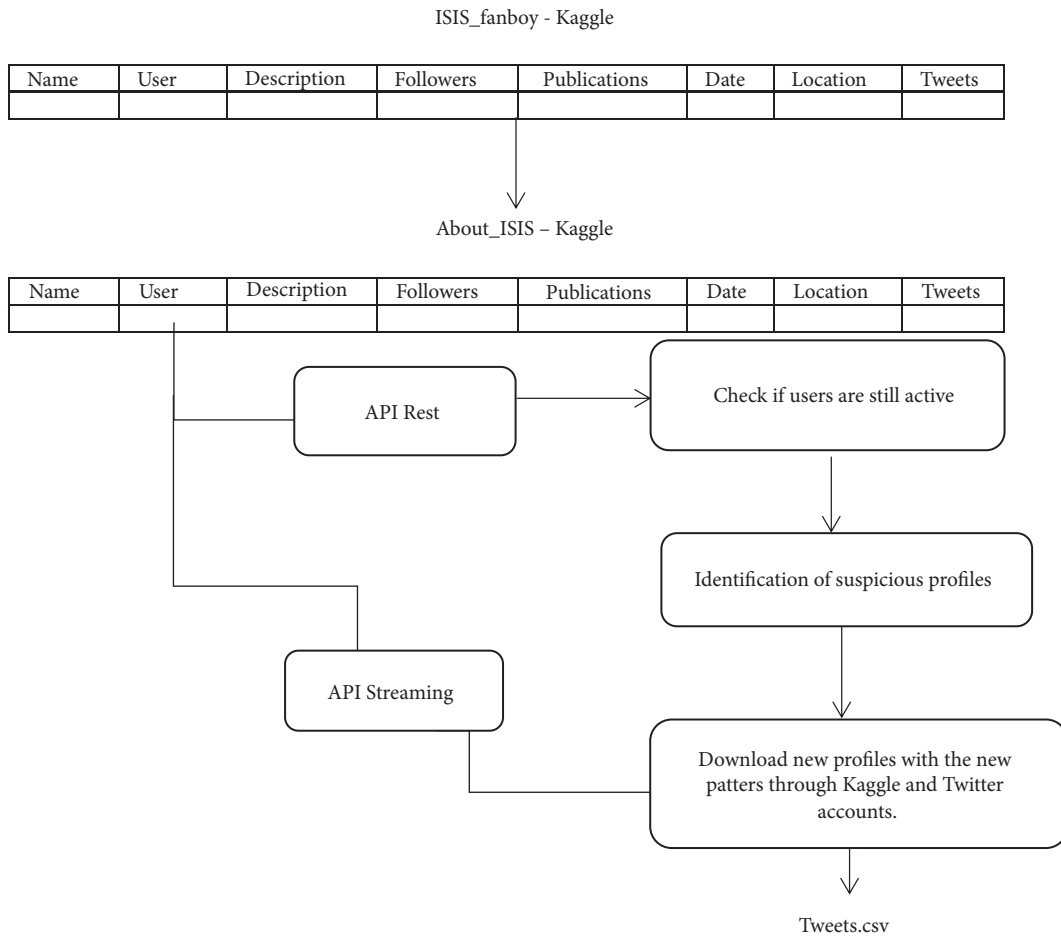


FIGURE 3: Tweet extraction procedure.

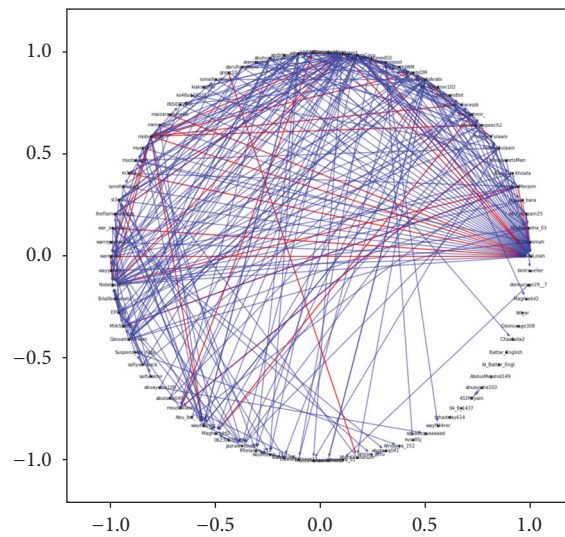


FIGURE 4: Graph indicating the interaction of users in Twitter related to the topic Jihadism.

information to broadcast it, or those who are following many influential users. To do so, we have weighted every link based on the number of retweets or mentions to a user (inside of the message @user, @user2... and so). We have not taken into account the sentiment of the message nor the frequency of shipment; these parameters will be used later in the fuzzy clustering procedure.

Most influential nodes are important in graph analysis, but many times in social communities those users are detected and located. Other criteria can be more important like nodes likely to be the most direct route between two influencer's nodes, or key nodes to reach the rest of nodes. That is why we have used centrality measures, to get different properties of a network and its behaviour. The more a node is centred, the more important it is. In particular, we have used two geometric measures, one being path based and the other one being a spectral measure to evaluate the influence and connections of a node within its community.

- (i) Degree Centrality Measure is defined as $C_{deg}(V) = deg(V)/|N| - 1$.

This geometric measure is used to find users very connected, those with the highest number of links with other nodes on the net. It takes into account the weighting on edges. We are not focusing on this type of users, as usually this measure could serve as a measure of popularity among nodes, though it is important to evaluate their relationship with the rest of nodes.

- (ii) Closeness is defined as $C(x) = (|N| - 1) / \sum_y d(y, x)$

This geometric measure represents the importance of each node according to how close is to the rest of nodes. Nodes with a high value tend to be very well connected to the most relevant nodes on their network and are perfect to broadcast information. They do not have to be very influential but are very active followers who broadcast information on the net and are very close to the most influential nodes.

- (iii) Betweenness centrality is a path based measure defined as

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (1)$$

where σ_{st} is the total number of shortest paths from node s to node t .

$\sigma_{st}(v)$ is the number of those paths that pass through v .

It indicates the nodes included in the shortest path between most of the nodes. For this work, this was a very interesting measure because it could highlight those nodes that serve as bridge for influential nodes. In this case, these nodes might not have many followers but can connect many relevant nodes.

- (iv) The last measure to be applied is the eigenvector, which takes into account the number of links of a given user, as the number of links of its connections,

and so could be considered as a hierarchical measure that computes not only your connections but the connections of whom you follow. Its value for node v is given by the v_{th} element of the eigenvector related to the first eigenvalue of the adjacency matrix of the graph [18],

As seen in Figure 5, the results obtained applying eigenvector and betweenness are quite similar in distribution. In this case measures that are not correlated are used as input for the fuzzy cluster to avoid redundant information and overtraining the cluster giving more relevance to some variables than others. We have chosen eigenvector as input for the clustering part as we are dealing with a structured problem, where there are people that train other users to broadcast information and so in a hierarchical model. The measure that better reflects this way of behavior is the eigenvector.

2.6. Fuzzy Clustering. Soft Computing techniques have been used in many different fields to deal with imprecision and uncertainty [19, 20].

In our problem, segmentation techniques are unsupervised methods used to classify information in groups created from similarities among individuals. The potential of the segmentation algorithms to show underlying structures in data can be applied in different fields such as classification, image processing, pattern recognition, modeling, and identification.

Segmentation techniques can be applied to quantitative or qualitative data. In this paper only quantitative ones will be used, to build the data matrix, which will have records as columns and measured variables as rows. By segmenting this data, the users are grouped in base to their similitude, understood from a mathematical point of view and defined as the "distance" among data or according to some prototypes of the group. This group depends, therefore, on the individuals being grouped together and on the definition of distance.

Within these segmentations we found two approaches:

- (i) Hard Clustering: Objects belong to just one segmentation. Different groupings are excluding.
- (ii) Fuzzy Clustering: This grouping technique applying fuzzy logic [21] allows different objects to belong to different segments simultaneously, but with different membership degree [22]. In many cases, this segmentation is more logical than the previous. In our case, a user can be interested in terrorism, though it has not been catalogued as dangerous, but with an elevated belonging degree in this group.

Therefore, fuzzy segmentation or fuzzy clustering is applied in this work, taking into account the nature of our problem with an objective function to obtain the optimal number of partitions. This optimization will lead to applying some nonlinear optimization algorithms to find a local minimum.

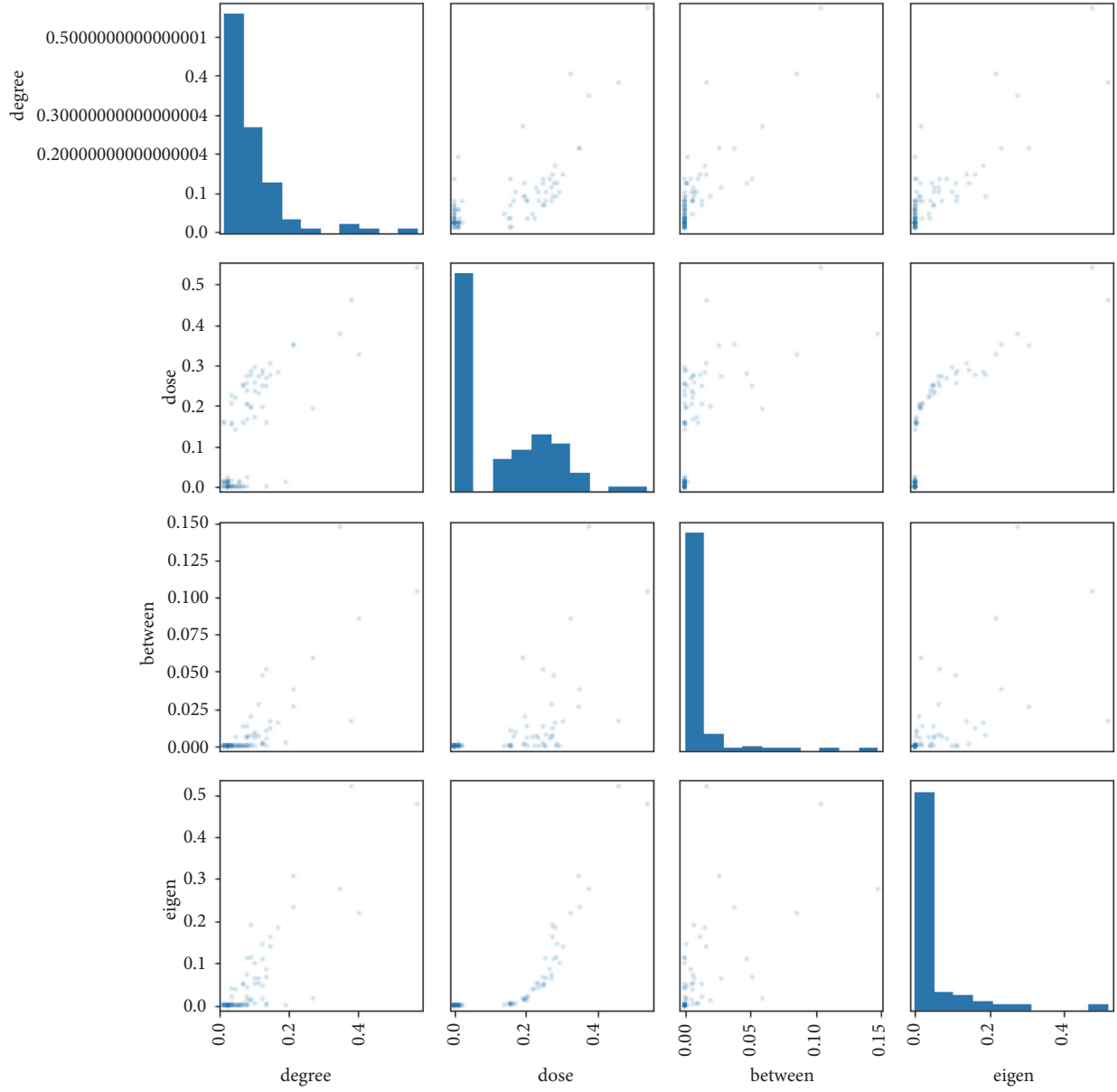


FIGURE 5: Distribution of users according to the proposed metrics.

2.6.1. Fuzzy Clustering. Groupments in this algorithm satisfy the following conditions:

$$\begin{aligned}
 \mu_{ij} &\in [0, 1], \quad 1 \leq i \leq c, \quad 1 \leq j \leq N \\
 \sum_{i=1}^c \mu_{ij} &= 1, \quad 1 \leq j \leq N \\
 0 < \sum_{j=1}^N \mu_{ij} &< N, \quad 1 \leq i \leq c
 \end{aligned} \tag{2}$$

where c is the number of groups and N is the number of records.

Fuzzy space for our data is the set defined by

$$F_c = \left\{ U \in \mathbb{R}^{c \times N} \mid \mu_{ij} \in [0, 1], \forall i, k; \sum_{i=1}^c \mu_{ij} = 1, \forall k; 0 < \sum_{i=1}^c \mu_{ij} < N, \forall i \right\} \tag{3}$$

2.6.2. Fuzzy Clustering c -Means. This algorithm is based on the optimization of Fuzzy partitions [23, 24].

$$J(Z, U, V) = \sum_{i=1}^c \sum_{j=1}^N (\mu_{ij})^m \|z_j - v_i\|_A^2 \tag{4}$$

where U is the membership matrix $[\mu_{ij}] \in F_c$ of our data and $V = [v_1, v_2, \dots, v_c]$ are the vectors characterizing the centers of these groupings for which we want to minimize our functional.

The standard A between our centers and the data is given according to the following:

$$D_{ijA}^2 = \|z_j - v_i\|_A^2 = (z_j - v_i)^T A (z_j - v_i) \quad (5)$$

The parameter $m \in [1, \infty)$ determines the fuzziness of the segments. The value of the cost function $J(Z, U, V)$ can be interpreted as a measure of the deviation between points v_i and centers z_j .

The minimization of this functional leads to a nonlinear optimization problem that can be solved through different methods as genetic algorithms or iterative minimization. However, the most popular for this application in particular is the iterative method of Picard.

The restriction of membership values μ_{ij} is imposed by Lagrange multipliers.

$$J(Z, U, V) = \sum_{i=1}^c \sum_{j=1}^N (\mu_{ij})^m \|z_j - v_i\|_A^2 + \sum_{j=1}^N \lambda_j \sum_{i=1}^c (\mu_{ij} - 1) \quad (6)$$

We can demonstrate that to minimize the functional it is necessary that

$$\mu_{ij} = \frac{1}{\sum_{k=1}^c (D_{ijA}/D_{kjA})^{2/(m-1)}}, \quad 1 \leq i \leq c, \quad 1 \leq j \leq N$$

$$v_i = \frac{\sum_{j=1}^N (\mu_{ij})^m z_j}{\sum_{j=1}^N (\mu_{ij})^m}, \quad 1 \leq i \leq c \quad (7)$$

Therefore, the parameters to be determined in the algorithm are as follows:

- (i) *Number of clusters*: this parameter is the most important and the one with greatest impact in the segmentation. If the number of groups to be divided our data is known, this parameter would be determined. We will determine the number of clusters though the fuzzy partition coefficient (FPC). This validation measure indicates how well our data are explained by this grouping; that is, the membership to each one of the segments of our data is, in general, strong and not fuzzy.
- (ii) *Fuzziness parameter*: parameter m affects significantly fuzziness in the segmentation. As it approaches 1, grouping ceases being fuzzy to be *hard*, and if it tends to ∞ it will be completely fuzzy. We have chosen the value ($m = 2$), as being the most used in bibliography.
- (iii) *Termination criterion*: as it is an iterative algorithm, it is necessary to establish a termination criterion to stop iterations. In our case, we have set 1000 iteration or reach an error lower than 0.005.

- (iv) *Distance matrix*: the calculation of distance implies establishing the scalar product matrix. The natural election is the identity matrix ($A = I$) but a distance matrix that is very extended is the inverse of the covariance matrix of the data, leading to the Mahalanobis standard.

$$A = R^{-1},$$

$$R = \frac{1}{N} \sum_{i=1}^N (z_i - \bar{z})(z_i - \bar{z})^T \quad (8)$$

The norm used influences the segmentation criterion changing the measure of dissimilarity. The Euclidean norm leads to hyperspherical groupings in the coordinate axes, while Mahalanobis leads to hyperellipsoidal groupings in the axes given by covariances between variables.

In addition to these parameters, in bibliography we can find several modifications of the algorithm:

- (i) Modifications that use an adaptive distance measure, as the algorithm of Gustafson-Kessel [25] and the fuzzy maximum likelihood estimation [26].
- (ii) Algorithms relaxing the condition on the probability of belonging to each segment ($\sum_{i=1}^c \mu_{ij} = 1, 1 \leq j \leq N$) indicating a level between each one of the groups.

In this work, we have checked the Euclidean norm, Mahalanobis, and Gustafson-Kessel.

The Gustafson-Kessel algorithm expanded the adaptive distance to detect different groupings with different geometrical forms. Each segment has its own distance given by

$$D_{ijA_i}^2 = (z_j - v_i)^T A_i (z_j - v_i) \quad (9)$$

The matrices A_i become variables that are optimized within the functional J so each group will have the distance that minimize its value. The only restriction imposed is that the determinant has to be positive, ($|A_i| = \rho_i, \rho_i > 0, \forall i$). Optimizing using the Lagrange multipliers method, we obtain that the distance matrices must fulfill this

$$A_i = [\rho_i \det(F_i)]^{1/m} F_i^{-1} \quad (10)$$

where F_i is the fuzzy covariance matrix of each one of the segments.

$$F_i = \frac{\sum_{j=1}^N (\mu_{ij})^m (z_j - v_i)(z_j - v_i)^T}{\sum_{j=1}^N (\mu_{ij})^m} \quad (11)$$

The parameters of this algorithm are, in addition to the general parameters of segmentation, the volumes of the groups ρ_i . If we do not have knowledge about this value, we set 1.

We have tested several measures to check which ones fits better to segment our dataset [27].

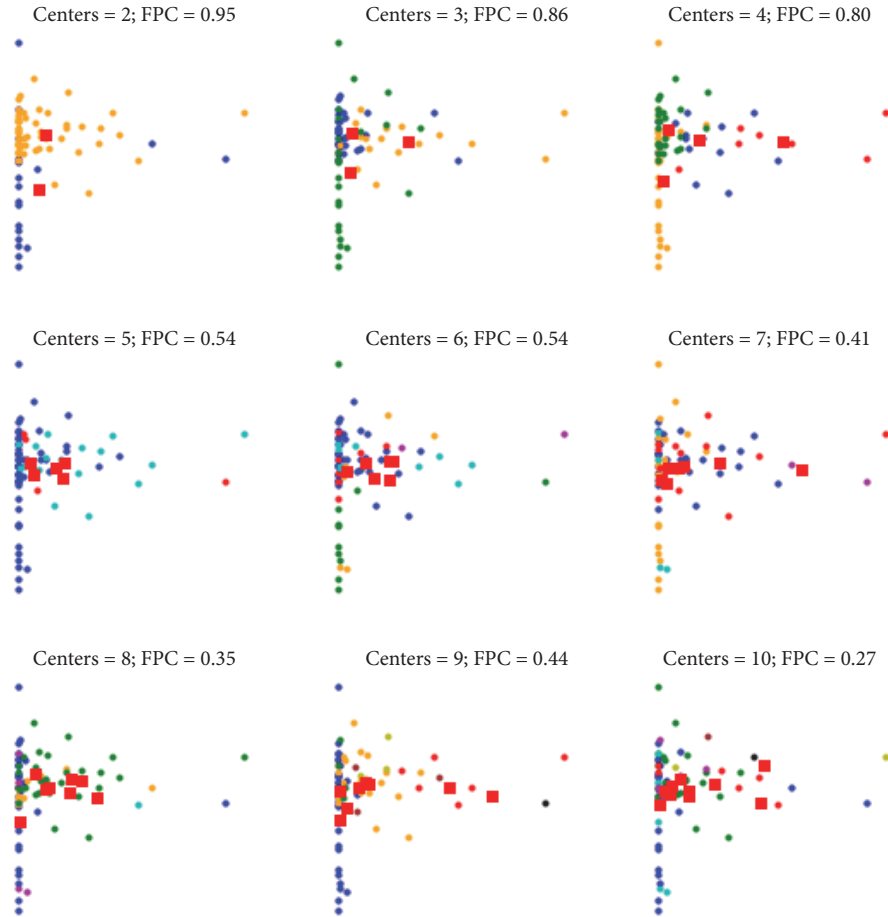


FIGURE 6: Results of the pro-Isis users with different levels of segmentation.

3. Results and Discussion

Once the previous methodology was defined and programmed, the Kaggle dataset with the pro-Isis Twitter registered users was firstly used. As criterion to choose which distance matrix and number of distances to be used, the configuration that allows a high membership degree for most of data has been established. For this dataset, the maximum value was obtained using the Gustafson-Kessel algorithm with two segments, as seen in Figure 6.

With this criterion, the division has been performed according to the variables mentioned (frequency, sentiment), eigenvector, and coherently with the rest of variables; it is possible to identify a more dangerous user group (red) and a less active group (blue).

However, one of the main advantages of this methodology is that we can identify users that have been identified within one group more than another, in spite of having a low membership degree. Figure 6 shows the membership degree to both groups, and the marked zone would be a user zone to be analyzed in detail. For example, with this same dataset, establishing a fuzzy membership threshold of 35-65%, we would obtain 2 doubtful users among 74 profiles. These

2 users are the focus of our work and should be studied individually and in time, to check whether they remain in the same place or have turned their behaviour to a more radical one, as seen in Figures 7 and 8.

To verify the consistency of our methodology, we performed a new experiment with an expanded dataset, added to the profiles identified as “fan boys”. We have included other profiles considered of interest because of their connections with the users of the first set.

When downloading Twitter information, we have discarded the number of followers (as we have the information of users whom mention/retweet and are mentioned/retweeted) and number of publications (as we have the number of tweets published in the sampling).

After filtering users by those who generate content and those who receive it, we programmed a weighted graph to obtain centrality measures in our network.

The best segmentation value was obtained using Mahalanobis distance in two clusters, obtaining a FPC of 0.85 as we can see in Figure 9, which are represented the FPC obtained in base to the number of fuzzy clusters.

To clarify the results, in Figure 10 we are representing segmentation in base to the variables that we have used to

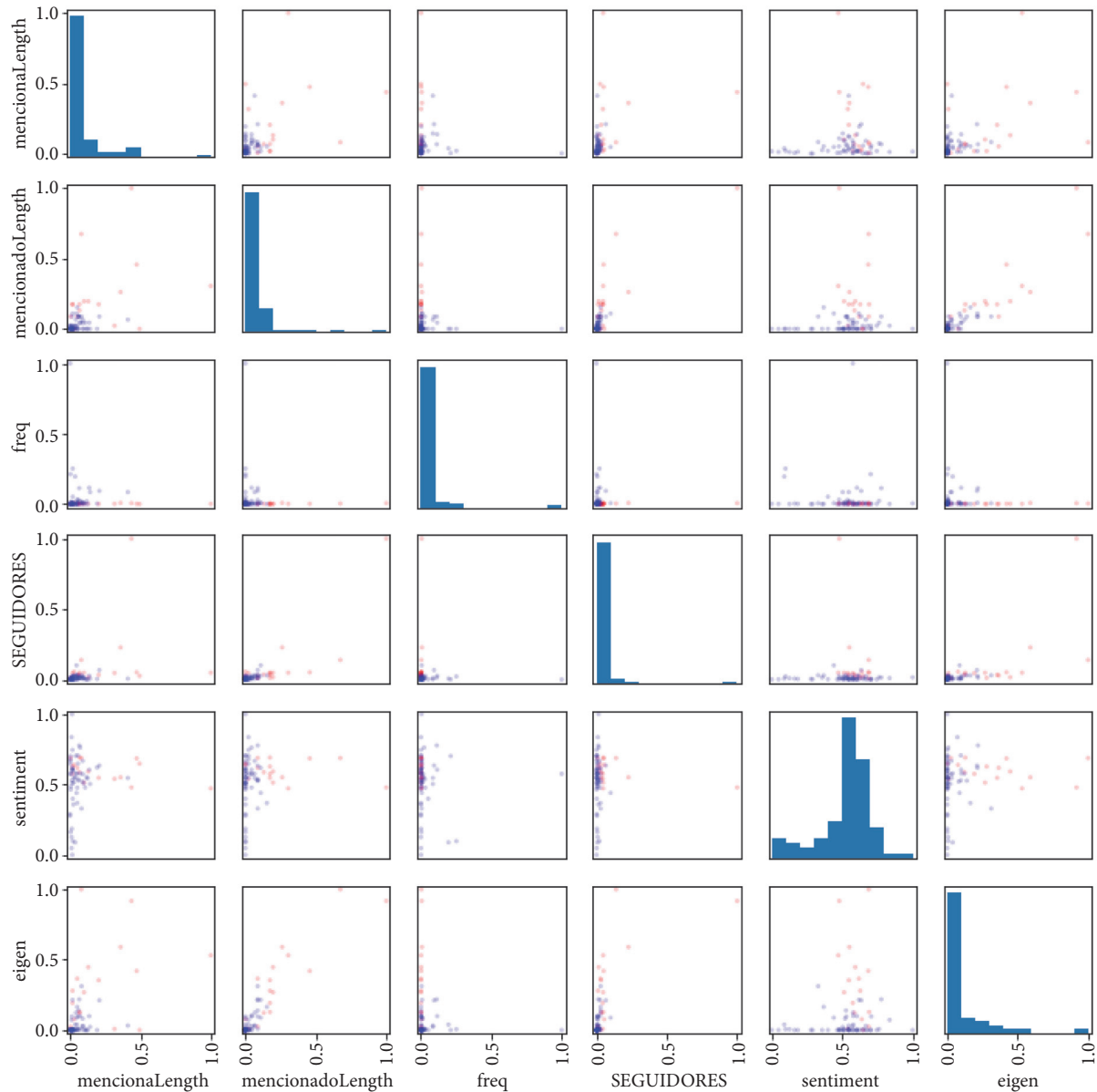


FIGURE 7: Distribution of the variables used in the fuzzy clustering.

calculate it, as frequency, eigenvector, and number of times that it has been mentioned.

For this experiment, if we set a fuzzy membership threshold of 0.45, we find 123 users to be studied from the 3395 in total. As in the previous case, these users, represented as the blue box of Figure 11, are susceptible to be studied and monitored to trace their behavior along time.

On the other hand, after identifying the most active users, we checked how many of the users categorized as active users in the first dataset were categorized as active as well as in this second set (blue group). From 74 users of the first categorization, 59 were correctly grouped, and the fuzzy membership of 15 left users is represented in the boxplot of Figure 12. The distribution of the membership shows that more than 25% of these false positive and negative users had a

very weak membership (between 0.50 and 0.6), which means that these users should be traced to check their behavior using our methodology.

4. Conclusions

The use of social networks as a manner to broadcast information has become a popular way to attract new followers to terrorism in general and Jihadism in particular. In this work, we have developed a methodology to identify potentially dangerous users that remain partially hidden, separated from those that are best known for being very active. In fact, the terrorist attack in Barcelona in 2017 was organized by terrorists whose profiles had not been classified as dangerous. Detecting and monitoring those new profiles can be crucial

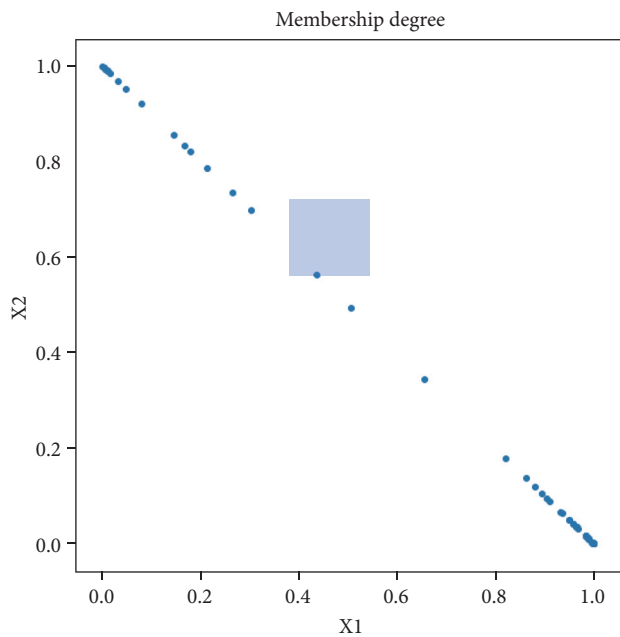


FIGURE 8: Membership degree function remarking the zone of users that should be analyzed.

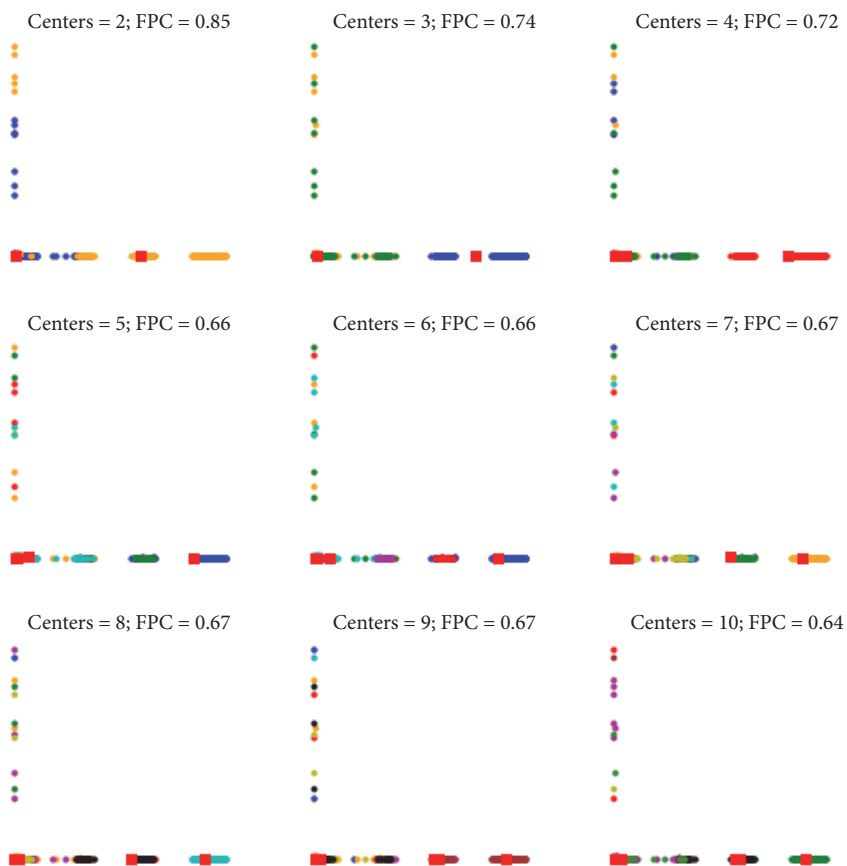


FIGURE 9: Representation of the FPC in base to the number of clusters.

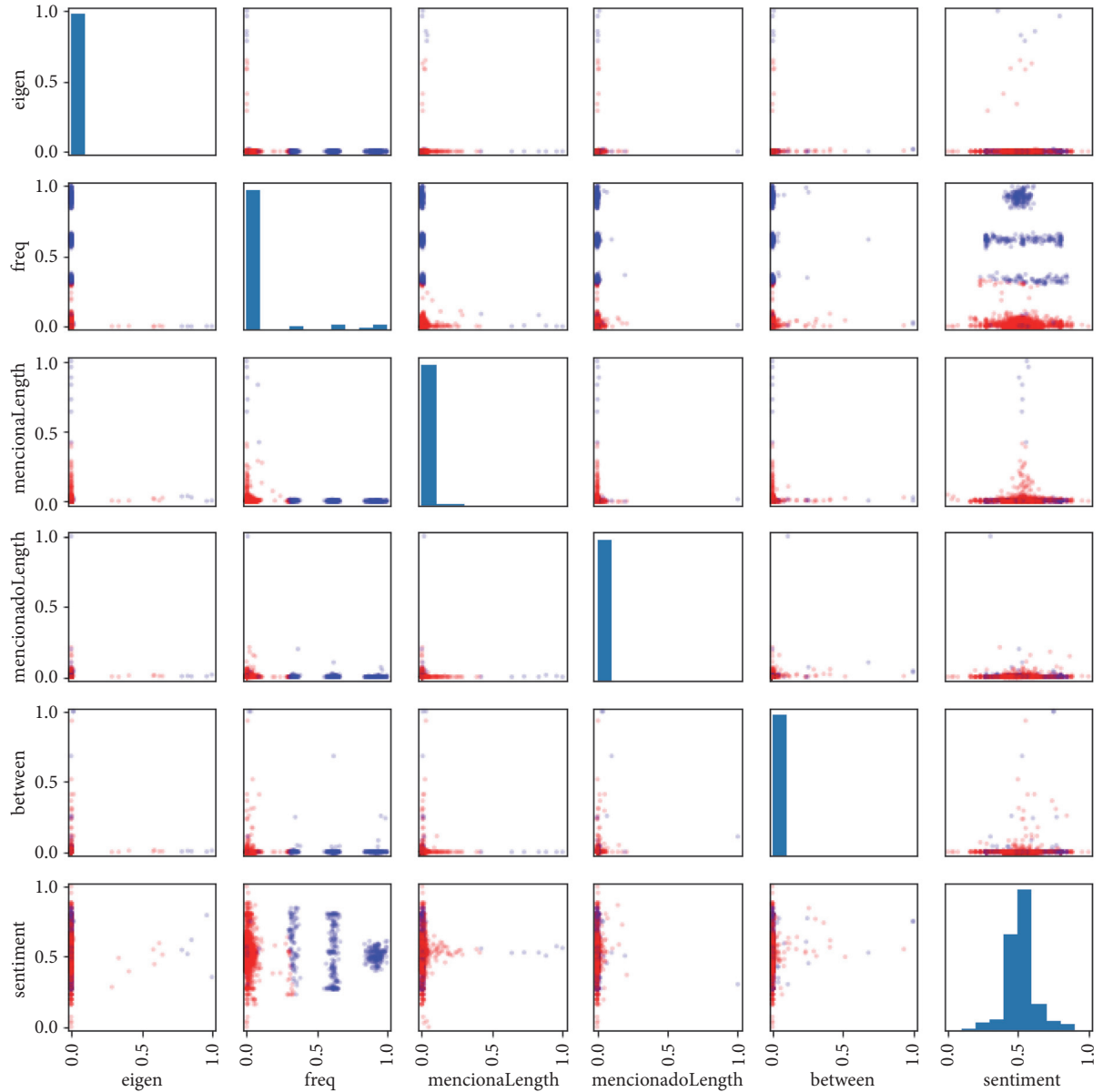


FIGURE 10: Segmentation variables for experiment 2.

to prevent or predict future terrorist actions and terrorist recruiting.

The presented methodology consists of defining a dataset of users plus several metrics to locate influential users. In addition, other metrics are obtained like frequency or the sentiment of their tweets. These metrics are used as vectors to perform segmentation of data. For this type of procedures, it is recommendable to use Soft Computing techniques that deal with the imprecision of the information. In the present problem we have used fuzzy clustering techniques to point out those users that were susceptible to be more active in the future and in consequence to be followed in time to check their behavior. Moreover, the analysis techniques proposed involve unsupervised algorithm, so they can be applied continuously, thereby this same

methodology could be used to monitor users marked as fuzzy.

As for future works, we would like to expand this methodology other environments where user profiles could have similar patterns, like pedophilia or fake news. Fake news is known to have an important economical impact if they damage the image of a company and an important political impact if they can manipulate a significant number of voters.

Data Availability

The Kaggle dataset used to support the findings of this study have been deposited in the Kaggle repository under the name How ISIS Uses Twitter: <https://www.kaggle.com/fifthtribe/how-isis-uses-twitter>.

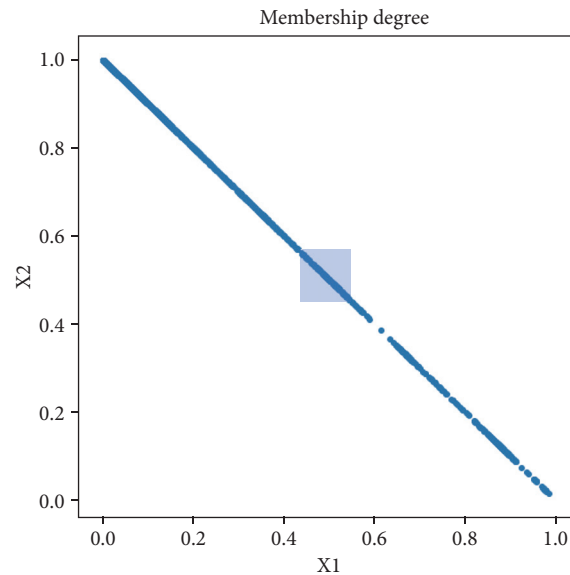


FIGURE 11: Membership degree function of experiment 2.

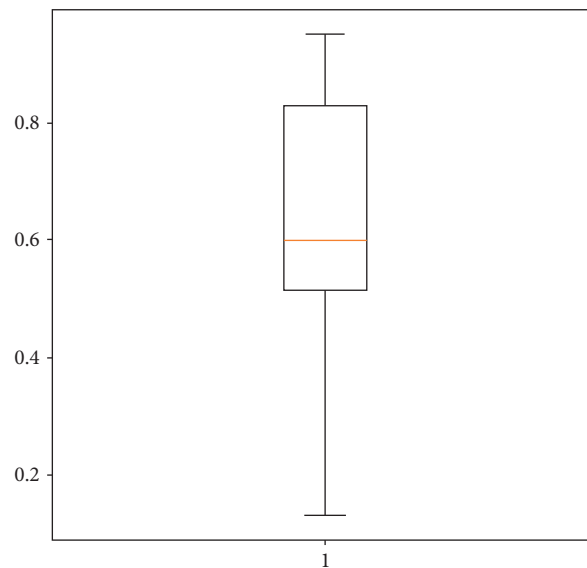


FIGURE 12: Boxplot representing the categorization of the most active users.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

- [1] J. Jordan, F. M. Mañas, and N. Horsburgh, "Strengths and weaknesses of grassroots jihadist networks: the madrid bombings," *Studies in Conflict & Terrorism*, vol. 31, no. 1, pp. 17–39, 2008.
- [2] A. Bermingham, M. Conway, L. McInerney, N. O'Hare, and A. F. Smeaton, "Combining social network analysis and sentiment analysis to explore the potential for online radicalisation," in *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining (ASONAM)*, pp. 231–236, Athens, Greece, July 2009.
- [3] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pp. 137–146, New York, NY, USA, August 2003.
- [4] J. H. Fowler and N. A. Christakis, "Dynamic spread of happiness in a large social network: longitudinal analysis over 20 years in the Framingham heart study," *British Medical Journal*, vol. 337, Article ID a2338, 9 pages, 2008.
- [5] C. Mao and W. Xiao, "A comprehensive algorithm for evaluating node influences in social networks based on preference analysis and random walk," *Complexity*, vol. 2018, pp. 1–16, 2018.

- [6] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, "Network analysis in the social sciences," *Science*, vol. 323, no. 5916, pp. 892–895, 2009, American Association for the Advancement of Science.
- [7] C. Ordóñez Galán, F. Sánchez Lasheras, F. J. de Cos Juez, and A. Bernardo Sánchez, "Missing data imputation of questionnaires by means of genetic algorithms with different fitness functions," *Journal of Computational and Applied Mathematics*, vol. 311, pp. 704–717, 2017.
- [8] M. Delgado, M. J. Martín-Bautista, D. Sánchez, and M. A. Vila, "Mining text data: special features and patterns," in *Pattern Detection and Discovery*, vol. 2447 of *Lecture Notes in Computer Science*, pp. 140–153, Springer, Berlin, Germany, 2002.
- [9] W. Aziguli, Y. Zhang, Y. Xie et al., "A robust text classifier based on denoising deep neural network in the analysis of big data," *Scientific Programming*, vol. 2017, Article ID 3610378, 10 pages, 2017.
- [10] C. J. de la Torre, D. Sánchez, I. Blanco, and M. J. Martín-Bautista, "Text mining: techniques, applications, and challenges," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 26, no. 04, pp. 553–582, 2018, World Scientific Publishing Company.
- [11] A. Hamdi, N. Monmarché, M. Slimane, and A. M. Alimi, "Fuzzy rules for ant based clustering algorithm," *Advances in Fuzzy Systems—Applications and Theory*, vol. 2016, Article ID 8198915, 16 pages, 2016.
- [12] I. A. Atiyah, A. Mohammadpour, and S. M. Taheri, "K C - means: a fast fuzzy clustering," *Advances in Fuzzy Systems—Applications and Theory*, vol. 2018, Article ID 2634861, 8 pages, 2018.
- [13] R. Singh, J. Singh, and R. Singh, "Fuzzy based advanced hybrid intrusion detection system to detect malicious nodes in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 3548607, 14 pages, 2017.
- [14] G. Bello-Orgaz, J. J. Jung, and D. Camacho, "Social big data: Recent achievements and new challenges," *Information Fusion*, vol. 28, pp. 45–59, 2016.
- [15] Z. Su, Q. Xu, and Q. Qi, "Big data in mobile social networks: a QoE-oriented framework," *IEEE Network*, vol. 30, no. 1, pp. 52–57, 2016.
- [16] G. Kossinets and D. J. Watts, "Empirical analysis of an evolving social network," *Science*, vol. 311, no. 5757, pp. 88–90, 2006.
- [17] J. Klausen, E. T. Barbieri, A. Reichlin-melnick, and A. Y. Zelin, "The YouTube jihadists: a social network analysis of al-muhajiroun's propaganda campaign," *Perspective on Terrorism*, vol. 6, no. 1, pp. 1–12, 2012, Terrorism Research Institute, <http://www.jstor.org/stable/26298554>.
- [18] M. E. J. Newman, *Networks: An Introduction*, Oxford University Press, Oxford, UK, 2010.
- [19] L. Álvarez Menéndez, F. J. de Cos Juez, F. Sánchez Lasheras, and J. A. Álvarez Riesgo, "Artificial neural networks applied to cancer detection in a breast screening programme," *Mathematical and Computer Modelling*, vol. 52, no. 7–8, pp. 983–991, 2010.
- [20] P. J. G. Nieto, J. R. A. Fernández, F. S. Lasheras, F. J. de Cos Juez, and C. D. Muñiz, "A new improved study of cyanotoxins presence from experimental cyanobacteria concentrations in the Trasona reservoir (Northern Spain) using the MARS technique," *Science of the Total Environment*, vol. 430, pp. 88–92, 2012.
- [21] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.
- [22] R. L. Cannon, J. V. Dave, and J. C. Bezdek, "Efficient implementation of the fuzzy c-means clustering algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 248–255, 1986.
- [23] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of Cybernetics*, vol. 4, no. 1, pp. 95–104, 1974, Taylor & Francis Group.
- [24] J. C. Bezdek, "Objective function clustering," in *Advanced Applications in Pattern Recognition*, pp. 43–93, Springer, Boston, MA, USA, 1981.
- [25] D. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *Proceedings of the 1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, pp. 761–766, 1979.
- [26] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 773–780, 1989.
- [27] H.-C. Liu, J.-M. Yih, D.-B. Wu, and S.-W. Liu, "Fuzzy C-mean algorithm based on "complete" Mahalanobis distances," in *Proceedings of the 2008 International Conference on Machine Learning and Cybernetics (ICMLC)*, pp. 3569–3574, Kunming, China, July 2008.

Research Article

Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm

Noemí DeCastro-García ¹, Ángel Luis Muñoz Castañeda,²
David Escudero García,² and Miguel V. Carriegos¹

¹*Departamento de Matemáticas, Universidad de León, Campus de Vegazana s/n, 24071 León, Spain*

²*Research Institute on Applied Sciences in Cybersecurity, Universidad de León, Campus de Vegazana s/n, 24071 León, Spain*

Correspondence should be addressed to Noemí DeCastro-García; ncasg@unileon.es

Received 7 December 2018; Accepted 17 January 2019; Published 4 February 2019

Guest Editor: Fernando Sánchez Lasheras

Copyright © 2019 Noemí DeCastro-García et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Selecting the best configuration of hyperparameter values for a Machine Learning model yields directly in the performance of the model on the dataset. It is a laborious task that usually requires deep knowledge of the hyperparameter optimizations methods and the Machine Learning algorithms. Although there exist several automatic optimization techniques, these usually take significant resources, increasing the dynamic complexity in order to obtain a great accuracy. Since one of the most critical aspects in this computational consume is the available dataset, among others, in this paper we perform a study of the effect of using different partitions of a dataset in the hyperparameter optimization phase over the efficiency of a Machine Learning algorithm. Nonparametric inference has been used to measure the rate of different behaviors of the accuracy, time, and spatial complexity that are obtained among the partitions and the whole dataset. Also, a level of gain is assigned to each partition allowing us to study patterns and allocate whose samples are more profitable. Since Cybersecurity is a discipline in which the efficiency of Artificial Intelligence techniques is a key aspect in order to extract actionable knowledge, the statistical analyses have been carried out over five Cybersecurity datasets.

1. Introduction

A Machine Learning (ML) solution for a classification problem is effective if it works efficiently in terms of accuracy and the required computational cost. The improvement of the first factor is faced on by several points of view that could affect to the second one in different forms.

The simplest way to get a ML model with a good accuracy is by testing and comparing different ML algorithms for the same problem and choosing, finally, the one that performs better. However, it is clear that, for instance, a decision tree model does not require, in general, as much computational time and memory to be trained as a Multilayer Perceptron. So, we will need to adjust the achieved accuracy with the available resources.

Another usual effective approach to reach a high accuracy is working with large training datasets. Nevertheless, this

solution is limited because of the associated computational cost (obtaining and storing the data, cleaning and transformation processes, and learning from the data). A possible alternative to the mentioned problem is to reduce the training without losing too much information [1, 2]. However, these kinds of solutions used to need an expensive data preprocessing phase.

The research related to this aspect, in addition to usual filtering the data, is focused on how to optimize the training set, and not only reduce it. The progressive sampling method shows that the performance with random samples with determined sizes is equal or more effective than working with the entire dataset [3]. Also, this solution used to be perfected with specifications about the classes' distribution, the selection of the samples, or the treatment of unbalanced datasets [4–6].

On the other hand, tuning hyperparameters of a ML algorithm is a critical aspect of the model training process that is considered the best practice for obtaining a successful Machine Learning application [7]. The Hyperparameters Optimization (HPO) problem requires a deep understanding of the ML model at hand due to the hyperparameters values settings and their effectivity, depending strongly on the ML algorithm, and the type of hyperparameter, discrete or continuous values. Also, it is a very costly process due to the large number of possible combinations to test and the needed resources to carry out the computations. Excluding the human expert of the process and minimizing the hyperparameter configurations to test are the underlying ideas in the automatic HPO.

Given a supervised ML algorithm, the continuous HPO is usually solved by gradient descent-based methods [8–10]. In the discrete case, not all optimization methods are suitable. The most common approaches to the HPO problem in the discrete case can be divided into two: Bayesian [11] and decision-theoretic methods. However, there are other optimization algorithms that are applied to the problem of hyperparameter values selection. This is the case, for instance, of the derivative-free optimization, the genetic algorithms such as Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES), or the simplex Nelder-Mead (NM) method [12–14]. In addition, continuous optimization methods, such as the Particle Swarm (PS) used for the ML algorithms of least-squares support vector machines [15–17], could be applied over discrete HPO problems [18].

Bayesian HPO algorithms balance the exploration process for finding promising hyperparameter configurations and the exploitation of the setting configuration in order to obtain always better results or gain more information with each test [19]. Consequently, Bayesian HPO are serialized methods that are difficult to parallelize, but they usually can find better combinations of hyperparameters in a few iterations. One of the most important Bayesian HPO methods is the Model-Based Optimization (SMBO). In the SMBO techniques, the way to construct the surrogate function to model the error distribution provides different methods. There are those that use Gaussian Process (GP) [20] or tree-based algorithms such as the Sequential Model Automatic Configuration (SMAC) or the Tree Parzen Estimators (TPE) method [21, 22]. On the other hand, the decision-theoretic approaches are based on the idea to search combinations of hyperparameter in the hyperparameter space, computing their accuracy, and finally pick the one that performed the best. If we test over a fixed domain of hyperparameters values, we have the *grid search*. More effective than grid search, even than some Bayesian optimization techniques, is the random sampling of different choices Random Search (RS) [23]. It is easy to implement, independent of previous knowledge, and easily parallelizable. Other optimization approaches also have reached very good results when applied to the selection of hyperparameter values. This is the case of research of [24] where evolutionary computation is used over deep neural networks.

Although applying HPO algorithms on a ML model reflects a great improvement in the results' quality of the

models' accuracy, we cannot overlook the computational complexity to implement these techniques. It is a critical issue because obtaining a good performance by applying HPO could require generations' samples, several function evaluations, and expensive computational resources. For example, the GP methods usually require a high number of iterations. Likewise, some derivative-free optimizations behave poorly in hyperparameter optimization problems because the optimization target is smooth [25]. In addition, some ML algorithms such as the neural networks are especially delicate because the number of possible values for the hyperparameters grows exponentially with the number of hidden layers. Other HPO algorithms are designed taking into account these limitations.

Recently, a Radial Basis Function (RBF) has been proposed as a deterministic surrogate model to approximate the error function of the hyperparameters through dynamic coordinate search that requires fewer evaluations in Multilayer Perceptron (MLP) and convolutional neural networks [26]. In [27], the Nelder-Mead and coordinate-search (derivative-free) methods are tested over deep neural networks, presenting more efficient numerical results than other well-known algorithms. Another option is to try to accelerate the algorithm. In [28], one way to implement the scheme Successive Halving (SH) is developed. The main ingredient behind SH is based on the observation that most of the HPO algorithms are iterative which suggests that stopping the algorithm when testing with a set of hyperparameters is not giving good results can be a good option.

Examples of this application are the accelerated RS version (2x) [29, 30] or Hyperband [31], which is based on the bandit-based approach. In this problem, a fixed finite set of resources must be distributed between different choices in a way that maximizes their expected gain. One recent example is developed in [19], where an algorithm for optimization of discrete hyperparameters based on compressed sensing is introduced, *Harmonica*.

We can also find studies about the effect of these HPO methods in the efficiency of the ML algorithms comparing different methods [32], or the possible options that can be tuned when a HPO algorithm runs over a dataset such as the number of iterations, number of hyperparameters, type of optimization, etc. [24, 33]. As far as we know, in the above context, analyzing the efficiency of the automatic HPO methods for specific supervised ML problems in terms of the size of the used dataset is needed.

The goal in this article is to carry out an empirical statistical analysis about the effect of the factor size of datasets used in the stage of the HPO in the performance of several ML algorithms. The studied response variables are the main issues in the efficiency of the algorithm: quality and dynamic complexity [34]. Regarding the quality, we take into account the most used metric for the goodness of a ML model, that is, the Accuracy. The others variables are the time complexity and the spatial complexity (amount of memory it requires for execution with a given input) due to these issues usually are influent limited resources. The underlying idea is to allocate the proportion of the whole dataset that maintains or improves the quality of the accuracy of the obtained

ML models and optimizing the dynamic complexity of the algorithms.

The research questions that will be studied are the following:

RQ1: Given a dataset, are there statistically meaningful differences in the performance (accuracy, time, and special complexity) of a ML algorithm when the HPO method it is applied to samples of different size of the dataset?

In the case that we obtain a positive answer, we follow to the next questions.

RQ2: Which are the effect and the behavior of each HPO algorithm depending on the dataset's size? In which cases we can gain efficiency if the HPO algorithm operates over small samples?

RQ3: Are the above results reliable? That is, are the results consistent for different HPO algorithms and different datasets?

In order to answer the research questions formulated above, an experiment has been carried out with different publicly available datasets about learning some tasks regarding cybersecurity events. Cybersecurity is a challenging research area due to the sophistication and the amount of kind of Cybersecurity attacks, which, in fact, increase very fast as time goes by. In this framework, the traditional tools and infrastructures are not useful because we deal with big data created with a high velocity, and the solutions and predictions must be faster than the threats. Artificial Intelligence and ML analytics have turned out in one of the most powerful tools against the cyberattackers (see [35–41]), but obtaining actionable knowledge from a database of Cybersecurity events by applying ML algorithms usually is a computationally expensive task for several reasons. A database of Cybersecurity contains, in general, a huge amount of dynamical, and unstructured but highly correlated and connected data, so we need to deal with some costly aspects of the quality of the data such as noise, trustworthiness, security, privacy, heterogeneity, scaling, or timeliness [42–44]. Also, the information is highly volatile, so the key issue is to get the profit as faster as possible, with the best possible performance and the smallest cost of resources and time. Then, the study of the complexity of applying Data Science over databases of Cybersecurity is an emergent and necessary field in order to increase confidence and social profit from automatizing processes and develop prescriptive policies to prevent and react to incidents faster and more secure.

Experimental analyses have been carried out in order to investigate the possible statistical differences over the efficiency of the Machine Learning algorithms Random Forest (RF), Gradient Boosting (GB), and MLP among using different sizes of samples in several HPO selection algorithms. We have used nonparametric statistical inference because, in an experimental design in the field of computational intelligence, these types of techniques are very useful to analyze the behavior of a method with respect to a set of algorithms [45]. In addition and based on the results of the above tests, we have assigned a profit level for each size of the whole dataset in terms of the response variables: accuracy, time, and spatial complexity. Finally, we have discussed the observed patterns and obtained results.

The paper is structured as follows. In Section 2, we describe the problem definition. In Section 3 we develop the Materials and Methods. This section includes the analyzed selected methods, both HPO and ML algorithms with the libraries or tools that we have used. Also, the tested datasets and the sampling of partitions have been explained, as well as the statistical analyses that have been carried out. In Section 4, the results and the discussion are included. Finally, our conclusions and references are given.

2. Problem Definition

Let \mathcal{P} be a distribution function. A Machine Learning algorithm, A , is a functional that maps each data set containing i.i.d. samples from \mathcal{P} , D^{train} , to a function $f_{A,D^{train}} := A(D^{train})$ that belongs to certain space of functions and that minimizes a fixed expected loss $\mathcal{L}(D^{train}, f_{A,D^{train}})$. Usually one has two more ingredients in this general framework. On one hand, the target space of functions of the algorithm depends on certain parameters, $\lambda = (\lambda_1, \dots, \lambda_n)$, that might take discrete or continuous values and that has to be fixed before applying the algorithm. In order to make explicit the dependency on λ , we will use the notation A_λ to refer to the algorithm and f_λ to refer to the target functions of A_λ . On the other hand, another data set obtained from \mathcal{P} , D^{test} , is given and serves to evaluate the loss, $\mathcal{L}(D^{test}, f_{A_\lambda, D^{train}})$, of the function provided by the algorithm over data independent from D^{train} .

Let Λ be the hyperparameter space, that is, the space in which λ takes values, and fix both the train and the test data sets. Denote by $D := D^{train} \cup D^{test}$ the union of both data sets. In this situation the only data that remains free in $\mathcal{L}(D^{test}, f_{A_\lambda, D^{train}})$ are the hyperparameters $\lambda = (\lambda_1, \dots, \lambda_n)$. Assume further that we are dealing with a classification Machine Learning problem, so that we can take \mathcal{L} to be the error rate, that is, one minus the cross-validation value. In this situation one can define the following function:

$$\begin{aligned} \Phi_{A,D} : \Lambda &\longrightarrow [0, 1] \\ \lambda &\longmapsto \text{mean}_{D^{test}} \mathcal{L}(D^{test}, f_{A_\lambda, D^{train}}) \end{aligned} \quad (1)$$

The HPO problem consists on minimizing $\Phi_{A,D}$ and, therefore, a HPO algorithm is a procedure that tries to reach $\lambda^* := \min_\lambda (\text{mean}_{D^{test}} \mathcal{L}(D^{test}, f_{A_\lambda, D^{train}}))$.

Usually, in practice, one has a dataset D that is split into two parts, D_1, D_2 , one for the optimization of the hyperparameters of the Machine Learning model and another one for training (and testing) the Machine Learning model with the given hyperparameters. The goal of the article is focused on how the size of the dataset in the HPO phase influences the performance of classifier given at the output of the training phase.

Suppose we have a ML model A , a HPO algorithm H , and a dataset D . The dataset D is split into different partitions randomly. We denote by $P_j(D)$ the partition j of the dataset D , and we assume that the size of $P_j(D)$ is smaller than the size of $P_l(D)$ for $j < l$. Then, applying H to the problem defined by $\Phi_{A, P_j(D)}$, we find optimal hyperparameters λ^j that can be used

TABLE 1: HPO algorithms. The star symbol * means that the chosen algorithms have needed some minor modifications.

Name	Reference	Ready?	Python minimal version	Smart	Library
PS	[15, 16]	√	2.7 y 3	X	[18]
TPE	[21]	√*	2.7 y 3	X	[51]
CMA-ES	[13]	√	2.7 y 3	X	[52]
NM	[14]	√	2.7 y 3	√	[52]
RS	[23]	√	2.7 y 3	X	[52]
SMAC	[22]	√	3	X	[53]

over D^{train} to create a classifier that will be tested on D^{test} . We denote the cross-validation value obtained in this last process by Acc^j . Also, the train time and the total time of the process and the spatial complexity spent will be collected (denoted by TC^j and SC^j respectively) for the analysis.

The study that will be done is statistical, so we will consider several datasets as well as many Machine Learning algorithms. Different state-of-the-art HPO algorithms will be considered and applied to every possible combination of Machine Learning models and datasets in order to compare them.

3. Materials and Methods

Experiments were conducted testing eight HPO methods over five cybersecurity datasets for three ML algorithms.

3.1. HPO Methods and ML Algorithms. As we mentioned above, we have evaluated the efficiency of the three well-known classifiers and predictor Machine Learning techniques algorithms: RF, GB, and MLP, commonly used in classification and prediction problems. The library used is [46].

Ensembles methods are commonly used because are based on the underlying idea that many different joint predictors will perform in a better way than any single predictor alone. Ensembling techniques could be divided into bagging and boosting methods.

First ones build many independent learners that are combined with average methods in order to give a final prediction. These handle overfitting and reduce the variance. The most known example of bagging ensemble methods is the RF. An RF is a classifier consisting of a chain of decision tree algorithms. Each tree is constructed by applying an algorithm to the training set and an additional random vector that is sampled via bootstrap resampling, so the trees will run and give independent results (see [47]). The scikit-learn implementation, RandomForestClassifier, combines classifiers by calculating an average of their probabilistic prediction, in contrast to the original publication. In the case of RF, we have two main hyperparameters: the number of decision trees that we should use and the maximum depth for each of them.

Second ones, Boosting, are ensemble techniques in which the predictors are made sequentially, learning from the mistakes of the previous predictor in order to optimize the subsequent learner. It usually takes less time/iterations to reach close to actual predictions, but we have to choose the

stopping criteria carefully. These reduce bias and variance and can with the overfitting. One example of the most common boosting methods is GB. The library that is used is the GradientBoostingClassifier, and we tune the discrete hyperparameters that are the number of predictors and the maximum depth of them.

On the other hand, an artificial neural network is a model that is organized in layers (input layer, output layer, and hidden layers). An MLP is a modification of the standard linear perceptron where multiple layers of connected nodes are allowed. The standard algorithm for training a MLP is the backpropagation algorithm (see [48, 49]). Class MLPClassifier in Scikit-learn implements MLP training algorithms for this ML model. By default, MLPClassifier has only one hidden layer with 100 neurons, the activation function for the hidden layer is $f(x) = \max(0, x)$, and the solver for weight optimization is "Adam" [50]. In this study, we will allow two hidden layers and we will run the number of neurons in each of these hidden layers.

In Table 1, the selected HPO algorithms to study are developed. Also, the references related to each one are given, as well as the minimal version of Python for which these algorithms work. In addition, we highlight if they are smart: that it is, if they stop by themselves in contrast to a number of iterations (trials) must be proposed.

3.2. Datasets. The choice of datasets selected for the experiments was motivated by different reasons: available in public servers, diversity in the number of instances, classes, and features, and relating to cybersecurity. The set of datasets $\mathcal{D} = \{D_1, D_2, D_3, D_4, D_5\}$ is described in Table 2.

Regarding the transformation of features data of treatable datasets, this has been performed manually by Python.

Dataset D_1 is a collection of spam (advertisements for products/web sites, make money fast schemes, chain letters, pornography, etc.) and nonspam e-mails whose purpose is to construct spam filters. Dataset D_2 is a database obtained by a real-time localization system for an autonomous robot with examples that are constructed with simulated attacks (Denial of Service and Spoofing) and nonattacks. Dataset D_3 is about de detection of Phishing websites. D_4 is a data set suggested to solve some of the inherent problems of the well-known KDD'99 data set. This has been treated by the approach given in [59, 60]. Finally, D_5 contains transactions made by credit cards. This unbalanced collection of data includes examples labeled as fraudulent or genuine. Also, the features are the result of a PCA transformation.

TABLE 2: Description of the set of datasets \mathcal{D} .

Dataset	Name	Instances	Features	Classes	Reference
D_1	Spambase	4601	57	2	[54]
D_2	Robots in RTLS	6422	12	3	[55]
D_3	Phishing websites	11055	30	2	[56–58]
D_4	Intrusion Detection (NSL-KDD)	148517	39	5	[59, 60]
D_5	Credit Card Fraud Detection	284807	30	2	[61]

TABLE 3: Description of the partitions of the set of datasets \mathcal{D} .

Dataset	P_1	P_2	P_3	P_4
D_1	383	766	2300	4601
D_2	535	1070	3211	6422
D_3	921	1842	5527	11055
D_4	12376	24752	74258	148517
D_5	23733	47467	142403	284807

Also, the datasets have different number of instances as well as features and number of classes of the target variable.

3.3. Sampling and Collection of Data. For each dataset D_i , four partitions, $\{P_j(D_i)\}_{j=1,\dots,4}$, have been created in a random way. These correspond to the proportions that are obtained when multiplied by 1/2, 1/6, 1/12 the whole set ($\mathcal{P} = \{P_1 = 8, 3\%, P_2 = 16, 3\%, P_3 = 50\%, P_4 = D_i\}$). We denote by $P_j(D_i)$ the partition on the dataset D_i where $i = 1, \dots, 5$ and $j = 1, 2, 3, 4$. These proportions have been chosen due to the fact that, in this way, we can deal with an amount of instances of different order of magnitude, ($10^2, 10^3, 10^4$, and 10^5), having similar orders in the same partition of each dataset; see Table 3.

On the other hand, we fix once and for all a partition $D_i^{train} \cup D_i^{valid} = D_i$ for each D_i into train data (80%) and validation data (20%), as well as a partition with the same proportion for each partition $P_j(D_i)$.

Finally, in order to build response variables to measure the goal of the study, we apply each $H_k \in \mathcal{H}$ over the all partitions, $P_j(D_i)$, obtaining the hyperparameter configuration $\lambda_{i,j}^k$. Then, the learning algorithm with the obtained hyperparameter configuration is run over D_i^{train} to construct a classifier that is validated over D_i^{valid} . At this step, we collect the accuracy obtained. This scenario is repeated 50 iterations. Also, the total time and spatial complexity of all process are stored. Then we create three response variables. We denote by $Acc_{i,j}^k$ the 50×1 array where the m -th component is the accuracy of the predictive model tested on D_i^{valid} that was trained over D_i^{train} with the hyperparameters $(\lambda_{i,j}^k)_m$ at the m -th iteration. We can measure the time and the spatial complexity used along this process and collect these data in two 50×1 arrays, $TC_{i,j}^k$, and $SC_{i,j}^k$, respectively.

The time complexity, measured in seconds, is the sum of the time needed by the HPO algorithm for finding the optimum hyperparameters and the time needed by the ML algorithm for the training phase. The spatial complexity,

measured in Kb, is defined as the maximum of use of memory along the HPO algorithm run, including the internal structures of the algorithm as well as train and test datasets load.

3.4. Technical Specifications. The analyses have been carried by the authors at high-performance computing facilitated by SCAYLE (www.scayle.es) over HP ProLiant SL270s Gen8 SE, with 2 processors Intel Xeon CPU E5-2670 v2 @ 2.50GHz with 10 cores each one, and 128 GB of RAM memory. They are equipped with 1 hard disk of 1TB and cards Infiniband FDR 56Gbps.

The analyses script has been implemented in Python language. Python uses an automated manager system of memory called garbage collector that releases the unused memory space. This phenomenon might be nondeterministic and certain fluctuations shown in the results may be due to not releasing the memory in that case.

It is worth noting that different technical tools (either software or hardware) could affect to the data about time and spatial complexity that have been collected. This is a fact that should be taken into account if we want to measure the effect of data sizes over the response variables in absolute terms, that is, over a single HPO algorithm. However, the influence of the technical specifications in the response variables is not a relevant factor in this study. This is a comparative study in which all the measures are collected under the same conditions, and the possible effect of the technical elements on the data is the same in each experiment. It is expected that the same behavior of the comparative encountered patterns will appear with other technical characteristics.

3.5. Analysis. The aim of the study is to decide if the size of data is a factor that influences in the efficiency of a ML algorithm using a HPO method among partitions of the data.

We perform the following statistical analysis:

- (1) First, for each level of the size, that is, the partitions P_j where $j = 1, 2, 3, 4$, we study the normality of $Acc_{i,j}^k$

TABLE 4: Level of gains where ‘=, <, >’ denotes statistically meaningful equality and differences, Me represents the median, and $j < l$.

Level	Condition
9	If $Me(Acc_{i,j}^k) > Me(Acc_{i,l}^k) \& Me(SC_{i,j}^k) < Me(SC_{i,l}^k)$
8	If $Me(Acc_{i,j}^k) > Me(Acc_{i,l}^k) \& Me(SC_{i,j}^k) = Me(SC_{i,l}^k)$
7	If $Me(Acc_{i,j}^k) = Me(Acc_{i,l}^k) \& Me(SC_{i,j}^k) < Me(SC_{i,l}^k)$
6	If $Me(Acc_{i,j}^k) = Me(Acc_{i,l}^k) \& Me(SC_{i,j}^k) = Me(SC_{i,l}^k)$
5	If $Me(Acc_{i,j}^k) > Me(Acc_{i,l}^k) \& Me(SC_{i,j}^k) > Me(SC_{i,l}^k) \& \Delta_{j,t}(SC_i^k) < \Delta_{j,t}(Acc_i^k) \& \Delta_{j,t}(SC_i^k) < \Delta_{j,t}(TC_i^k)$
4	If $Me(Acc_{i,j}^k) = Me(Acc_{i,l}^k) \& Me(SC_{i,j}^k) > Me(SC_{i,l}^k) \& \Delta_{j,t}(SC_i^k) < \Delta_{j,t}(TC_i^k)$
3	If $Me(Acc_{i,j}^k) < Me(Acc_{i,l}^k) \& Me(SC_{i,j}^k) < Me(SC_{i,l}^k) \& \Delta_{j,t}(SC_i^k) > \Delta_{j,t}(Acc_i^k) \& \Delta_{j,t}(TC_i^k) > \Delta_{j,t}(Acc_i^k)$
2	If $Me(Acc_{i,j}^k) < Me(Acc_{i,l}^k) \& Me(SC_{i,j}^k) = Me(SC_{i,l}^k) \& \Delta_{j,t}(TC_i^k) > \Delta_{j,t}(Acc_i^k)$
1	If $Me(Acc_{i,j}^k) < Me(Acc_{i,l}^k) \& Me(SC_{i,j}^k) > Me(SC_{i,l}^k) \& \Delta_{j,t}(TC_i^k) > \Delta_{j,t}(Acc_i^k) \& \Delta_{j,t}(TC_i^k) > \Delta_{j,t}(SC_i^k)$
0	In another case

(resp., $TC_{i,j}^k$ and $SC_{i,j}^k$). Once it has been determined that these data do not follow a normal statistical distribution, we have performed nonparametric tests. Wilcoxon’ test for two paired samples was conducted in order to decide whether there are statistically meaningful differences or not among $Acc_{i,1}^k$, $Acc_{i,2}^k$, $Acc_{i,3}^k$, and $Acc_{i,4}^k$ (resp., for $TC_{i,j}^k$ and $SC_{i,j}^k$) for all $D_i \in \mathcal{D}$, $H_k \in \mathcal{H}$ and for the three selected ML algorithms (RF, GB, and MLP). The choice of this test is due to the fact that the response variables that we compare are obtained by the application of the ML algorithms over the dataset D_i with the splits D_i^{train} and D_i^{valid} , but with the different setting $(\lambda_{i,j}^k)^*$ and $(\lambda_{i,4}^k)^*$. The study has been carried out with a significance level $\alpha = 0.05$.

- (2) At this point, we have applied the inference described over the accuracy, the time, and the spatial complexity. Then, we have obtained the p -values of each partition’s comparison for each response variable, for each HPO method, and for each D_i . That is six comparisons for P_1VsP_4 along five datasets providing a total of 30 decisions about statistical equality or not for each ML algorithm (the same process for P_2VsP_4 , P_3VsP_4 , respectively). Then, we have computed the rate of p -values that provides statistical differences by comparison of partitions, by response variables, and in a total way.
- (3) From the results obtained in Wilcoxon’s tests we assign to each $P_j(D_i)$ a level of gain with regard to $P_4(D_i)$ for $j = 1, 2, 3$. The mapping is carried out according to Table 4 where $\Delta_{j,t}(Acc_i^k)$ denotes the rate of absolute difference of the median accuracy between operating with the model obtained through HPO on $P_j(D_i)$ and $P_4(D_i)$ (resp., spatial and time complexity). Namely,

$$\Delta_{j,t}(Acc_i^k) = \frac{|Me(Acc_{i,j}^k) - Me(Acc_{i,t}^k)|}{\min[Me(Acc_{i,j}^k), Me(Acc_{i,t}^k)]} \quad (2)$$

$$\Delta_{j,t}(SC_i^k) = \frac{|Me(SC_{i,j}^k) - Me(SC_{i,t}^k)|}{\min[Me(SC_{i,j}^k), Me(SC_{i,t}^k)]} \quad (3)$$

$$\Delta_{j,t}(TC_i^k) = \frac{|Me(TC_{i,j}^k) - Me(TC_{i,t}^k)|}{\min[Me(TC_{i,j}^k), Me(TC_{i,t}^k)]} \quad (4)$$

This correspondence is shown for each algorithm H_k and dataset D_i .

The design of Table 4 was done based on how many response variables show a positive efficiency rate when optimizing the hyperparameter values with a smaller partitions instead of the whole dataset. First, we have created nine levels of gain (9 is the highest level), each of them determined by the number of response variables in which we reach more efficient results prioritizing the accuracy against T.C. and S.C. Also, we have sorted the levels of gain from the best combination to the worst, dropping those combinations in which the loss is larger than the profit. Due to the Python garbage collector, the response variable with a more clear increasing trend should be the time complexity instead of spatial complexity. Then, we suppose that the inequality $T.C.(P_j(D_i)) < T.C.(P_4(D_i))$ holds always true.

- (4) Finally, we compute the average of gain of the smaller partitions in a general way, per datasets, and ML algorithms. These results let us measure the reliability of the conclusions.

4. Results and Discussion

The first research question deals with whether the size of a partition used in the HPO phase influences in certain sense on the efficiency of the algorithm. Once the comparison’s tests described in the above section have been done, we account, for each ML model, how many combinations show statistically significant differences across all the HPO methods and all the datasets. Although we find an influence on the response variables, we do not know whether this influence is positive or not. So, the second research question is focused on

TABLE 5: Statistically significant differences in each dimension for all HPO across all D_i of RF for each comparison between $P_j(D_i)$ and $P_4(D_i)$ and the total (%).

Combination	Accuracy	Time Complexity	Spatial Complexity	(Acc+T.C.+S.C)
P_1 Vs P_4	13/30 = 43.33%	30/30 = 100%	30/30 = 100%	73/90 = 81.11%
P_2 Vs P_4	12/30 = 40%	29/30 = 96.66%	30/30 = 100%	71/90 = 78.88%
P_3 Vs P_4	3/30 = 10%	29/30 = 96.66%	30/30 = 100%	62/90 = 68.88%
$(P_1 + P_2 + P_3)$ Vs P_4	28/90 = 31.11%	88/90 = 97.77%	90/90 = 100%	

TABLE 6: Statistically significant differences in each dimension for all HPO across all D_i of GB for each comparison between $P_j(D_i)$ and $P_4(D_i)$ and the total (%).

Combination	Accuracy	Time Complexity	Spatial Complexity	(Acc+T.C.+S.C)
P_1 Vs P_4	19/30 = 63.33%	29/30 = 96.66%	30/30 = 100%	78/90 = 86.66%
P_2 Vs P_4	20/30 = 66.66%	29/30 = 96.66%	29/30 = 96.66%	78/90 = 86.66%
P_3 Vs P_4	14/30 = 46.66%	28/30 = 93.33%	30/30 = 100%	72/90 = 80%
$(P_1 + P_2 + P_3)$ Vs P_4	53/90 = 58.88%	86/90 = 95.55%	89/90 = 98.88%	

TABLE 7: Statistically significant differences in each dimension for all HPO across all D_i of MLP for each comparison between $P_j(D_i)$ and $P_4(D_i)$ and the total (%).

Combination	Accuracy	Time Complexity	Spatial Complexity	(Acc+T.C.+S.C)
P_1 Vs P_4	4/30 = 13.33%	29/30 = 96.66%	30/30 = 100%	63/90 = 70%
P_2 Vs P_4	5/30 = 16.66%	28/30 = 93.33%	30/30 = 100%	63/90 = 70%
P_3 Vs P_4	1/30 = 3.33%	26/30 = 86.66%	30/30 = 100%	57/90 = 63.33%
$(P_1 + P_2 + P_3)$ Vs P_4	10/90 = 11.11%	83/90 = 92.22%	90/90 = 100%	

the study of this differences and equalities. Next, we analyze the reliability of the results. Finally, we include an overview of the global results that are obtained.

4.1. Research Question 1. In the case of RF, the results included in Table 5 show that the obtained accuracy in the 31.11% of possible combinations between the smaller partitions and the whole dataset can be considered statistically different. However, the time and spatial complexity have a very high amount of statistically significant differences (97.77% and 100%, respectively). Hence spatial and time complexity are affected by the size of the dataset (as expected). But, on the other hand, the size of the dataset does not impact on the accuracy in a critical way.

Also, we can see that the partition $P_1(D_i)$ reaches the highest number of differences in the accuracies, as well as in time and spatial complexity, while the behavior of the partition $P_3(D_i)$ is the more statistically similar to the whole dataset. Note that, in the 90 % of the cases, the accuracy obtained with this partition can be considered equal to accuracy obtained with $P_4 = D_i$. Note that the behavior of the accuracies shows an increasing trend of similarity related to the increasing size of the partition.

In the case of GB, the results included in Table 6 show that the obtained accuracy in the 41.12% of possible combinations among the smaller partitions and the whole dataset can be considered statistically equivalent. However, the time and spatial complexity have a very high amount of statistically significant differences (95.55% and 98.88%, respectively). So,

we can confirm that the dataset's size has an effect on these last dimensions in a strong way, and over the accuracy in a medium level.

Regarding the concrete partitions, we have a greater homogeneity of the results in the GB than in RF, although $P_3(D_i)$ remains as the partition with the most similar accuracy with respect to the whole dataset (53.34 %). If we take into account the three response variables the rate of differences is quite high, $P_3(D_i)$ being the most similar with an 80% of differences. Note that, in this case, there is no increasing trend of similarity in the accuracies as the size grows up.

Finally, in the case of MLP, the results included in Table 7 show that the obtained accuracies in the 88.89 % of possible combinations among the smaller partitions and the whole dataset can be considered statistically equal. Then, the accuracy is not being quite affected by the size of the dataset used in the HPO phase. However, the time and spatial complexities have a very high amount of statistically significant differences (92.22% and 100%, respectively).

It is worth noting that $P_3(D_i)$ has obtained 96.67% of equivalent accuracies, but the behavior of the similarity in the accuracies, sorted by the size of the partition, is not found either in this case.

In general, we have found a high effect of the dataset's size used in the HPO over the time and spatial complexity, for the three ML algorithms. In the case of the accuracy, the ensemble methods (RF and GB) show a medium effect

TABLE 8: Patterns of profit.

ML method	Pattern 1	Pattern 2	Pattern 3	Pattern 4
RF	NM	SMAC, RS	PS, TPE, CMA-ES	
GB	RS	NM, TPE, CMA-ES	SMAC, PS	
MLP		RS, SMAC	CMA-ES	PS, TPE, NM

TABLE 9: Average rate of statistically significant differences in each dimension for all HPO across all D_i for all ML algorithms and for each comparison between $P_j(D_i)$ and $P_4(D_i)$ (%).

Combination	Accuracy	Time Complexity	Spatial Complexity
P_1 Vs P_4	39.99%	97.77%	100%
P_2 Vs P_4	41.10%	95.55%	98.88%
P_3 Vs P_4	19.99%	92.21%	100%
$(P_1 + P_2 + P_3)$ Vs P_4	33.7%	95.18%	99.62%

(around a rate of 40%), while in the MLP, the level of effect is very low.

4.2. Research Question 2. In order to study in depth whether the considered effect is positive or negative when we work with smaller partitions, the evolution of the response variables as the size of the partition grows up is developed. We are going to study how are the encountered differences in each dimension of the efficiency.

In Figures 1, 2, and 3, the charts of the evolution of the behavior of the studied dimensions are shown for each dataset D_i and for RF, GB, and MLP, respectively.

Both the time and spatial complexity appear with an increasing trend, the first one being more highlighted. Also, the case of spatial complexity is more variable in the case of MLP than in the ensemble methods. So, in order to gain efficiency when tuning the hyperparameters with a smaller proportion of data, different levels are assigned according to Table 4. As we have mentioned above, the levels of profit are proposed in terms of the gain in the accuracy and spatial complexity due to the fact that the time complexity shows a clear increasing trend.

Note that, in general, is not true that the accuracy increases as the size of the partition used for the HPO phase does. This can be seen more clearly when the ML model is GB or MLP and, certainly, depends on the dataset. See, for instance, the three charts for D_2 and D_3 in Figures 1, 2, and 3.

The gain's averages obtained in RF, GB, and MLP are included in Figures 4, 5, and 6.

In all the studied ML algorithms, we can obtain a gain when smaller partitions are used to HPO. Also, we can clearly find four different patterns (see Table 8). The first and second one show that the partition P_2 obtains the highest and the lowest profit. The third one is an increasing trend from P_1 to P_3 . In the case of MLP, we also detect another pattern that stabilizes the gain from P_2 .

In the case of RF, the lowest level of gain is 4.5 and the maximum value is 7. The other ensemble method, GB, has obtained levels of profit between 2.5 and 8. Finally, the MLP algorithm shows values between 5 and 7.5. Then, the neural

network is the algorithm in which the HPO phase performs better with smaller partitions, followed by RF and GB.

In addition, for all ML algorithms there is at least one HPO algorithm that obtains a profit of level greater than 6 in the smaller partitions P_1 and P_2 , namely, the NM algorithm. In case of P_3 , it is should be noted that the minimum level of gain is 5.5 for all HPO and ML techniques.

4.3. Research Question 3. If we compute the average of gain in each dataset, we obtain the results shown in Figure 7.

The averages of the profit level for RF are between 3.2 and 8.5, being the largest dataset D_5 the one in which we reach more efficiency working with smallest partition and showing a decreasing trend. On the other hand, the dataset D_3 is the one in which we get less gain of efficiency. The same behavior is presented in MLP up to a little loss of level, between 2.5 and 7.5. In the case of GB, the averages of the profit level are between 3.2 and 7.2, being the largest dataset D_4 the one in which we reach more efficiency working with smallest partition.

Finally, we can conclude that, in a general way, in all datasets we obtain efficiency optimizing the hyperparameter values with smaller partition, although the data were different in terms of features, number of instances, or classes of the target variable. So, the results are consistent and reliable.

4.4. Global Results. In Table 9 we include the global average rate of statistically significant differences in each dimension (see Tables 5, 6, and 7). As we have mentioned above, time and spatial complexity show an increasing trend directly related to the size of the partition. Therefore, the high rate of differences appearing in these variables in Table 9 is intuitive and expected. However, the behavior of the accuracy is different. We can observe that the accuracy that we get with smaller partitions is statistically equivalent to the obtained accuracy with the whole dataset in more than the 60% of the cases. This rate is greater than 80% for P3 (the 50% of the all dataset). This fact should be taken into account, especially in case of big data contexts, in order to optimize the available

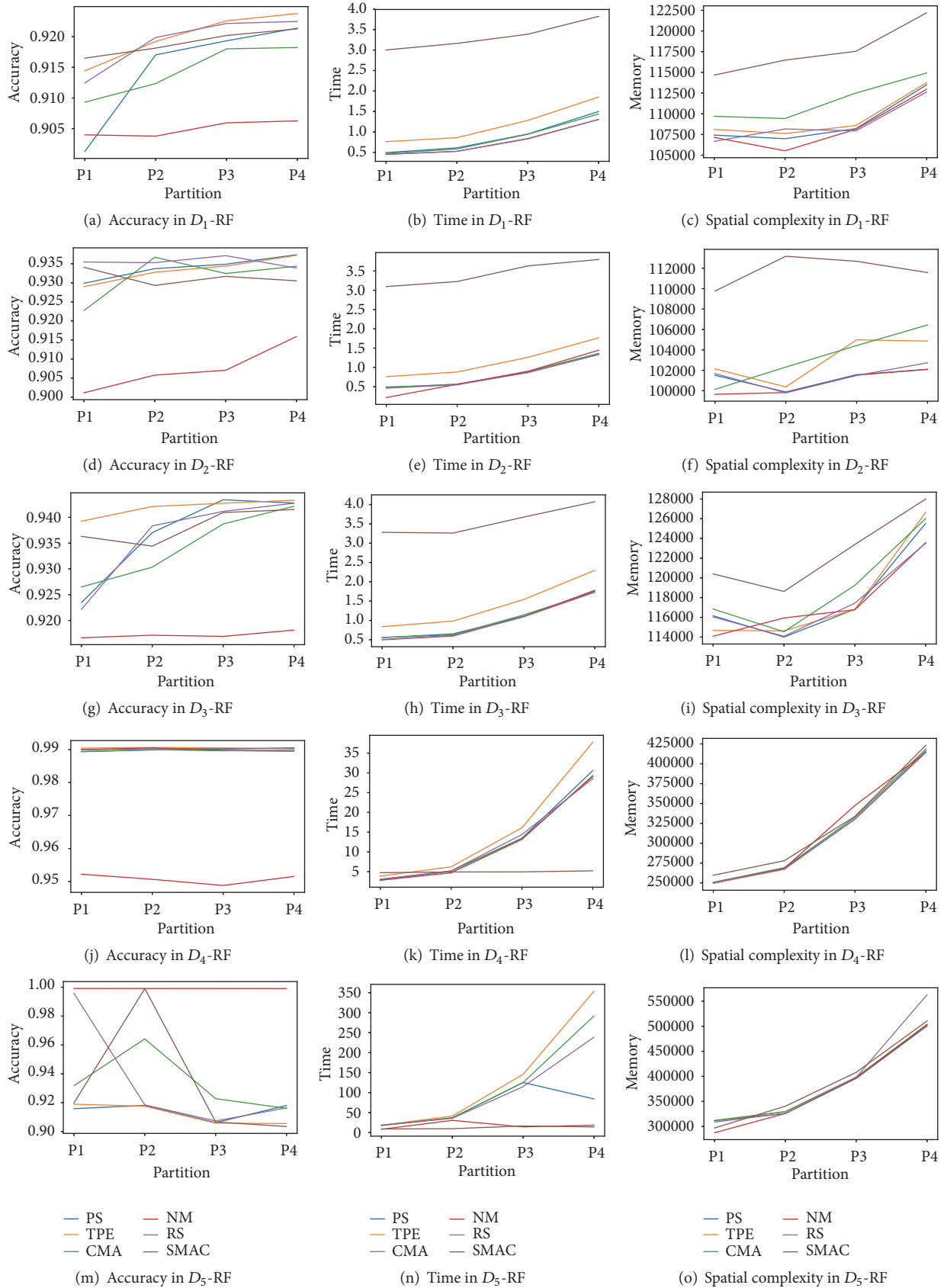


FIGURE 1: Accuracy, time, and spatial complexity of RF.

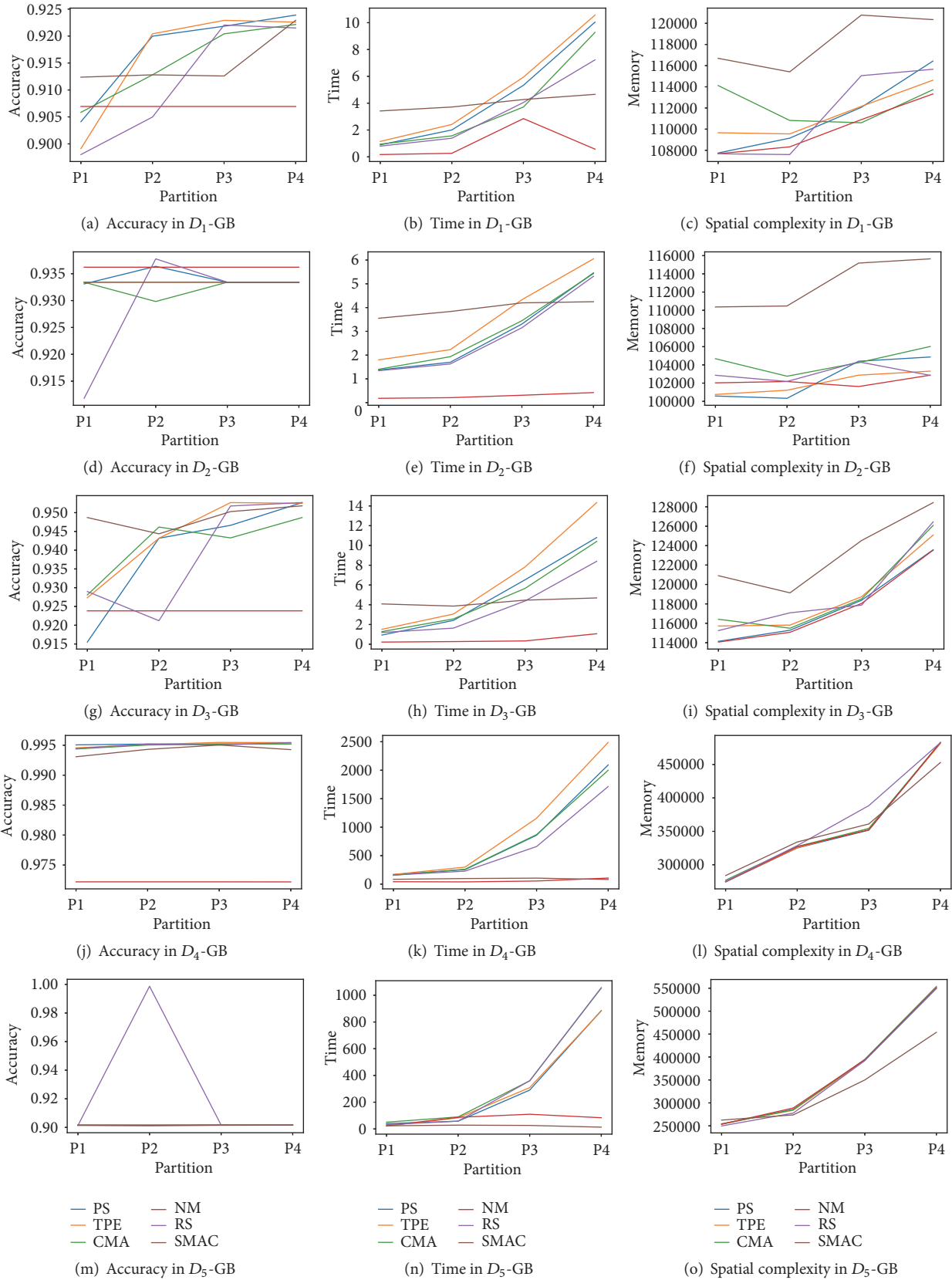


FIGURE 2: Accuracy, time, and spatial complexity of GB.

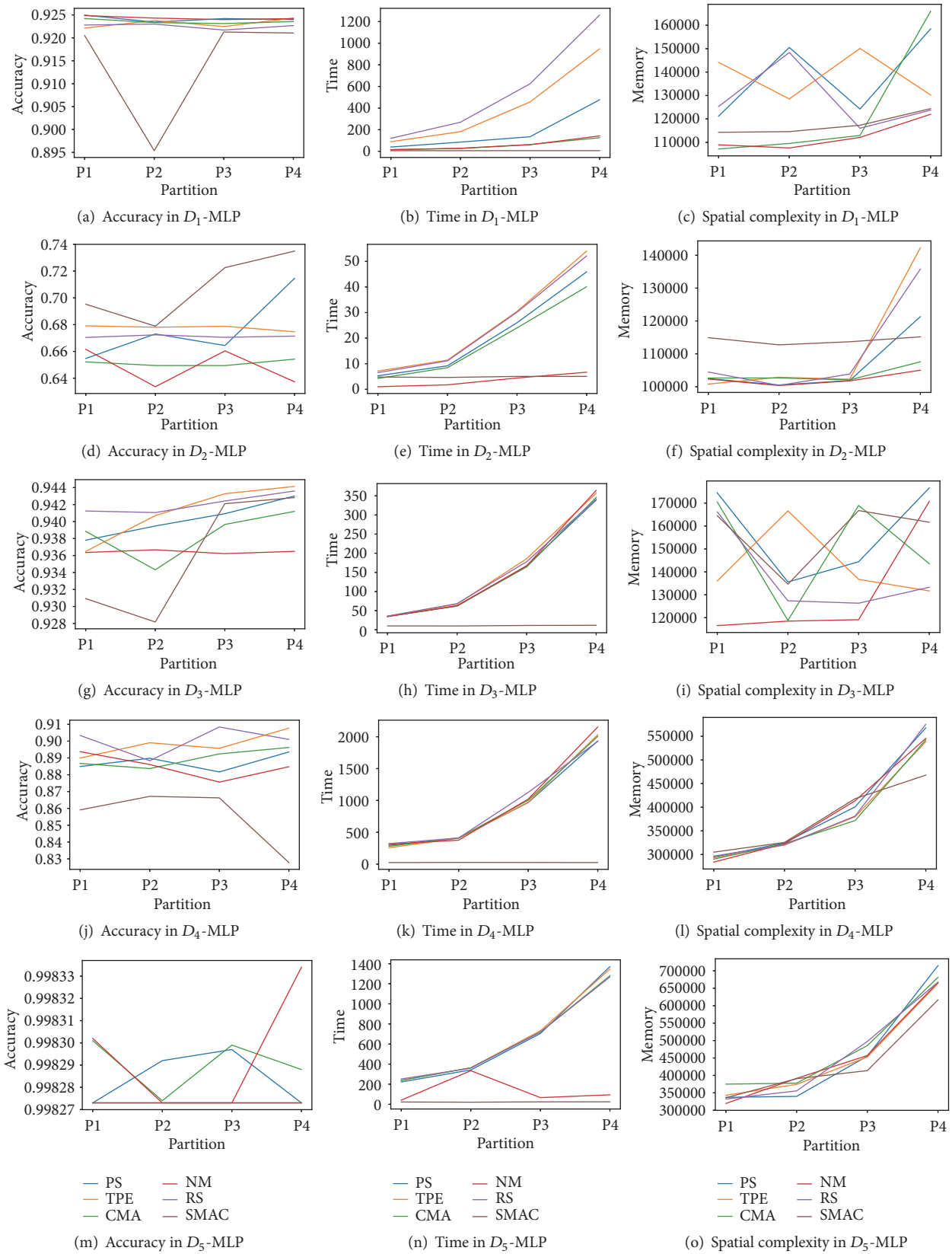


FIGURE 3: Accuracy, time, and spatial complexity of MLP.

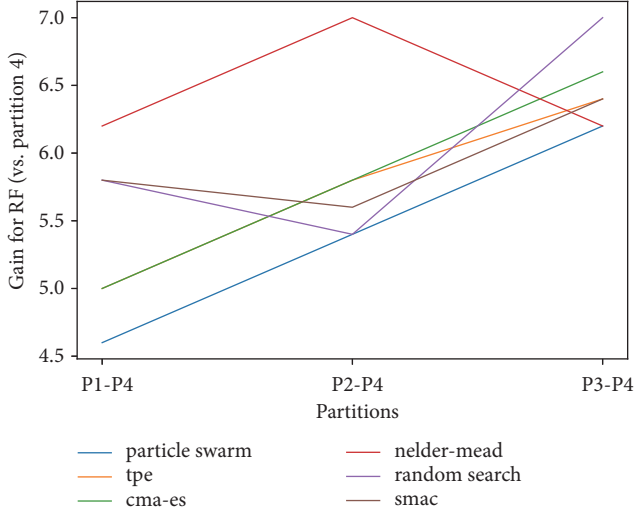


FIGURE 4: Average of gain for each smaller partition with respect to the whole dataset in RF.

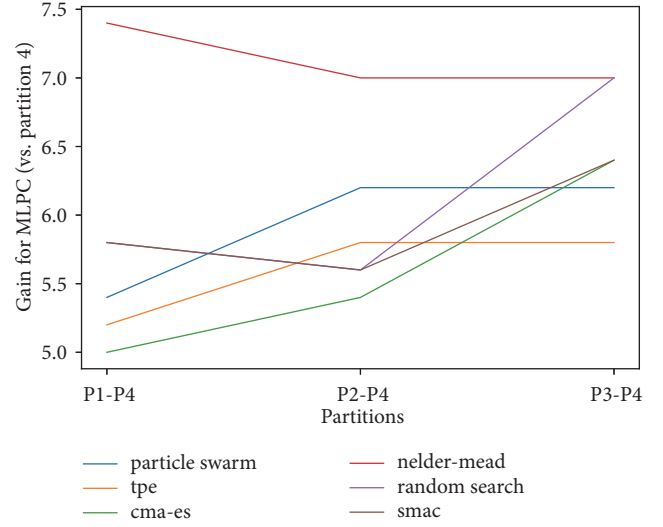


FIGURE 6: Average of gain for each smaller partition with respect to the whole dataset in MLP.

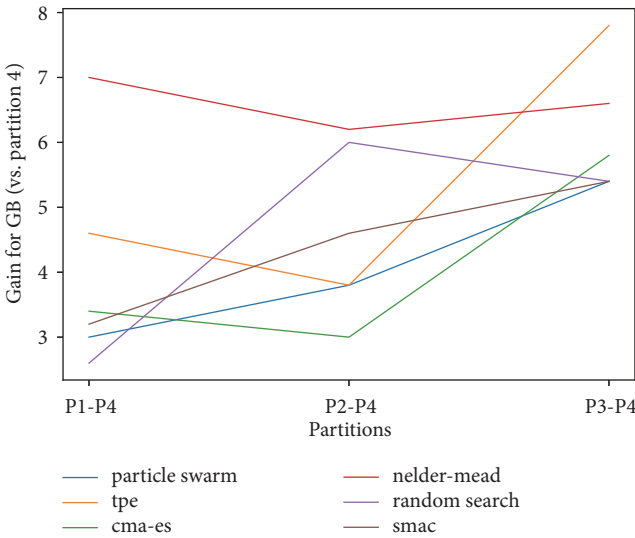


FIGURE 5: Average of gain for each smaller partition with respect to the whole dataset in GB.

resources and avoid losing quality in an eventual solution based on ML.

Once we have statistically and globally analyzed the efficiency of a ML algorithm when we use smaller partitions instead of the whole dataset, the next step is going in depth over those cases in which differences are encountered. It should be noted that these could provide a gain or a loss of effectiveness. Also, a statistically difference in the accuracy, for example, could be due to a variation as low as a few ten thousandth of the total. In these cases, the relevance of this difference is meaningfully related to the order of gain or loss in the time and spatial complexity. The global average of the level of gain, according to Table 4, is included in Table 10 (see Figures 4, 5, 6, and 7).

TABLE 10: Average level of profit for all ML algorithms considered and for each comparison between $P_j(D_i)$ and $P_4(D_i)$ across all HPO and D_i (%).

Combination	HPO	D_i
P_1 Vs P_4	5.04	5.04
P_2 Vs P_4	5.44	5.475
P_3 Vs P_4	6.33	6.175

The obtained global results show an average level of profit between 5.04 and 6.33 over 9 with an increasing trend related to the size of the partition.

5. Conclusions and Future Work

Cybersecurity is a dynamical and emerging research discipline that faces on problems which are increasingly complex and that requires innovative solutions. The value of a database of Cybersecurity is very high due to the actionable knowledge that we extract from it, but in the most cases, we have to deal with a big volume of data that entails expensive costs of resources and time. Artificial Intelligence techniques, such as Machine Learning, are powerful tools that allow us to extract and generate knowledge in Cybersecurity, among other fields.

One of the main issues, in order to reach quality results by Machine Learning, is the optimization of the hyperparameter values of the algorithm. However, the automatic HPO methods suppose a cost in terms of dynamical complexity.

In this work, we have developed a statistical analysis of the fact of using smaller samples of the dataset for this process, and its influence on the effectiveness of the Machine Learning solution. The study was carried out over five different public datasets of Cybersecurity. The results let us conclude that working with smaller partitions turns out to be more efficient than performing the same process with the whole dataset. The obtained gain is different depending on the ML algorithm and

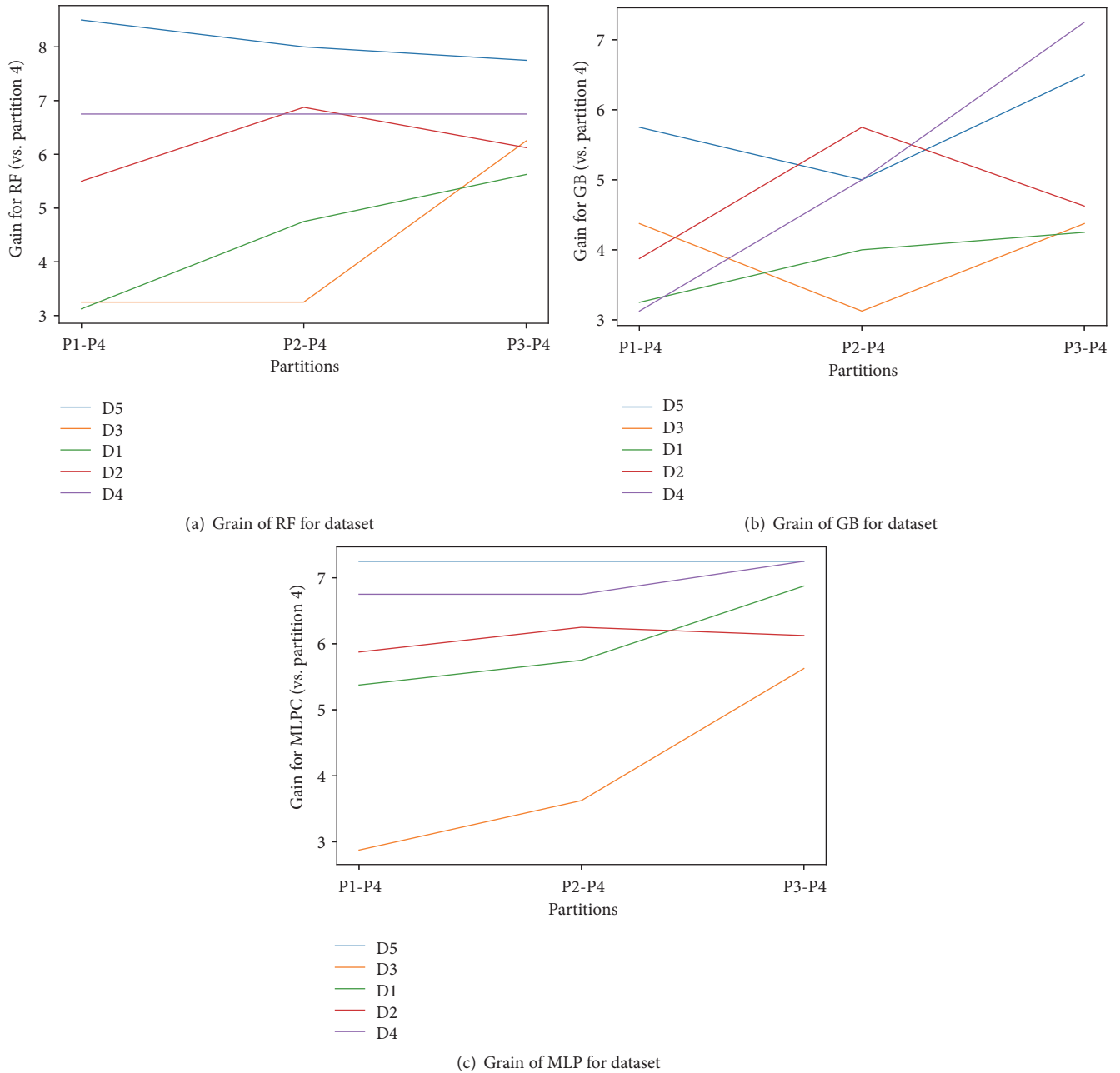


FIGURE 7: Average of gain for each smaller partition with respect to the whole dataset in each dataset D_i .

the HPO technique providing the highest level of profit with the 50% of the dataset.

As future work, the following landmark would be to search what is the optimal partition to obtain the best gain, as well as studying other HPO methods over more types of ML algorithms.

Acronyms

- CMA-ES: Covariance Matrix Adaptation Evolutionary Strategies
- GP: Gaussian Process
- GB: Gradient Boosting

- HPO: Hyperparameters Optimization
- ML: Machine Learning
- MLP: Multilayer Perceptron
- NM: Nelder-Mead
- PS: Particle Swarm
- RBF: Radial Basis Function
- RS: Random Search
- RF: Random Forest
- SMAC: Sequential Model Automatic Configuration
- SMBO: Sequential Model-Based Optimization
- SH: Successive Halving
- TPE: Tree Parzen Estimators.

Data Availability

The datasets supporting this meta-analysis are from previously reported studies and datasets, which have been cited. The processed data are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank the Spanish National Cybersecurity Institute (INCIBE), who partially supported this work. Also, in this research, the resources of the Centro de Supercomputación de Castilla y León (SCAYLE, www.scayle.es), funded by the “European Regional Development Fund (ERDF)”, have been used.

References

- [1] Z. Cheng and Z. Lu, “A Novel Efficient Feature Dimensionality Reduction Method and Its Application in Engineering,” *Complexity*, vol. 2018, Article ID 2879640, 14 pages, 2018.
- [2] I. Czarnowski and P. Jędrzejowicz, “An Approach to Data Reduction for Learning from Big Datasets: Integrating Stacking, Rotation, and Agent Population Learning Techniques,” *Complexity*, vol. 2018, Article ID 7404627, 13 pages, 2018.
- [3] F. Provost, D. Jensen, and T. Oates, “Efficient progressive sampling,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 23–32, San Diego, Calif, USA, 1999.
- [4] W. Ng and M. Dash, “An Evaluation of Progressive Sampling for Imbalanced Data Sets,” in *Proceedings of the Sixth IEEE International Conference on Data Mining Workshops*, Hong Kong, China, 2006.
- [5] F. Portet, F. Gao, J. Hunter, and R. Quiniou, “Reduction of Large Training Set by Guided Progressive Sampling: Application to Neonatal Intensive Care Data,” in *Proceedings of the Intelligent Data International Workshop on Analysis in Medicine and Pharmacology (IDAMAP2007)*, pp. 1-2, Amsterdam, Netherlands, July 2007.
- [6] G. M. Weiss and F. Provost, “Learning when training data are costly: The effect of class distribution on tree induction,” *Journal of Artificial Intelligence Research*, vol. 19, pp. 315–354, 2003.
- [7] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2013)*, pp. 847–855, 2013.
- [8] Y. Bengio, “Gradient-based optimization of hyperparameters,” *Neural Computation*, vol. 12, no. 8, pp. 1889–1900, 2000.
- [9] J. Luketina, M. Berglund, K. Greff, and T. Raiko, “Scalable gradient-based tuning of continuous regularization hyperparameters,” *CoRR*, 2015, <https://arxiv.org/abs/1511.06727>.
- [10] D. Maclaurin, D. Duvenaud, and R. P. Adams, “Gradient-based hyperparameter optimization through reversible learning,” in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning (ICML15)*, vol. 37, pp. 2113–2122, 2015, <http://dl.acm.org/citation.cfm?id=3045118.3045343>.
- [11] L. Mockus, V. Tiesis, and A. Zilinskas, “The application of bayesian methods for seeking the extremum,” *Towards Global Optimization*, 1978.
- [12] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*, *MPS/SIAM Series on Optimization*, 2009, <http://epubs.siam.org/doi/book/10.1137/1.9780898718768>.
- [13] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.
- [14] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.
- [15] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.
- [16] X. C. Guo, J. H. Yang, C. G. Wu, C. Y. Wang, and Y. C. Liang, “A novel LS-SVMs hyper-parameter selection based on particle swarm optimization,” *Neurocomputing*, vol. 71, no. 16-18, pp. 3211–3215, 2008.
- [17] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, 2001.
- [18] F. Fortin, F. De Rainville, M. Gardner, M. Parizeau, and C. Gagné, “DEAP: Evolutionary algorithms made easy,” *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012, <https://github.com/DEAP/deap>.
- [19] E. Hazan, A. Klivans, and Y. Yuan, “Hyperparameter Optimization: A Spectral Approach,” in *In Proceedings of the Sixth International Conference on Learning Representations*, 2018.
- [20] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian optimization of machine learning algorithms,” in *Proceedings of the Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012 (NIPS 2012)*, pp. 2960–2968, 2012.
- [21] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Proceedings of the Neural Information Processing Systems 2011 (NIPS 2011)*, pp. 2546–2554, 2011.
- [22] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential Model-Based Optimization for General Algorithm Configuration,” in *Learning and Intelligent Optimization (LION)*, C. A. C. Coello, Ed., vol. 6683 of *Lecture Notes in Computer Science*, pp. 507–523, Springer, Berlin, Germany, 2011.
- [23] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.
- [24] W. Konen, P. Koch, O. Flasch et al., “Tuned Data Mining: A Benchmark Study on Different Tuners,” in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO-2011)*, SIGEVO/ACM, New York, NY, USA, 2011.
- [25] E. R. Sparks, A. Talwalkar, D. Haas et al., “Automating model search for large scale machine learning,” in *Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC 2015)*, pp. 368–380, 2015.
- [26] I. Ilievski, T. Akhtar, J. Feng, and C. A. Shoemaker, “Efficient hyperparameter optimization for deep learning algorithms using deterministic RBF surrogates,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 822–829, San Francisco, Calif, USA, 2017.

- [27] Y. Ozaki, M. Yano, and M. Onishi, "Effective hyperparameter optimization using Nelder-Mead method in deep learning," in *Proceedings of the IPS Transactions on Computer Vision and Applications*, pp. 9–20, 2017.
- [28] K. G. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016)*, pp. 240–248, 2016.
- [29] B. Recht, *Embracing the random*, 2016, <http://www.argmin.net/2016/06/23/hyperband>.
- [30] B. Recht, *The news of auto-tuning*, 2016, <http://www.argmin.net/2016/06/20/hybertuning/>.
- [31] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: a novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research (JMLR)*, vol. 18, pp. 1–52, 2017.
- [32] P. Koch, B. Wujek, O. Golovidov, and S. Gardner, "Automated Hyperparameter Tuning for Effective Machine Learning," in *Proceedings of the SAS Global Forum 2017 Conference*, SAS Institute Inc, Cary, NC, USA, 2017, <http://support.sas.com/resources/papers/proceedings17/SAS514-2017.pdf>.
- [33] K. Eggenberger, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Efficient benchmarking of hyperparameter optimizers via surrogates," in *Proceedings of the Twenty-Ninth Association for the Advancement of Artificial Intelligence*, pp. 1114–1120, 2015.
- [34] P. Turney, "Types of cost in inductive concept learning," in *Proceedings of the ICML '2000 Workshop on Cost-Sensitive Learning*, pp. 15–21, 2000.
- [35] D. Bogdanova, P. Rosso, and T. Solorio, "Exploring high-level features for detecting cyberpedophilia," *Computer Speech and Language*, vol. 28, no. 1, pp. 108–120, 2014.
- [36] P. Galán-García, J. Gaviria de la Puerta, C. Laorden Gómez, I. Santos, and P. G. Bringas, "Supervised Machine Learning for the Detection of Troll Profiles in Twitter Social Network: Application to a Real Case of Cyberbullying," in *Proceedings of the International Joint Conference SOCO'13-CISIS'13-ICEUTE'13. Advances in Intelligent Systems and Computing*, Á. Herrero et al., Ed., vol. 239, Springer, 2014.
- [37] S. Miller and C. Busby-Earle, "The role of machine learning in botnet detection," in *Proceedings of the 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 359–364, Barcelona, Spain, 2016.
- [38] J. R. Moya, N. DeCastro-García, R. Fernández-Díaz, and J. L. Tamargo, "Expert knowledge and data analysis for detecting advanced persistent threats," *Open Mathematics*, vol. 15, no. 1, pp. 1108–1122, 2017.
- [39] A. Shenfield, D. Day, and A. Ayesh, "Intelligent intrusion detection systems using artificial neural networks," *ICT Express*, vol. 4, no. 2, pp. 95–99, 2018.
- [40] F. Vanhoenshoven, G. Napoles, R. Falcon, K. Vanhoof, and M. Koppen, "Detecting malicious URLs using machine learning techniques," in *Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, Athens, Greece, December 2016.
- [41] M. Zanin, M. Romance, S. Moral, and R. Criado, "Credit Card Fraud Detection through Parenclitic Network Analysis," *Complexity*, vol. 2018, Article ID 5764370, 9 pages, 2018.
- [42] N. DeCastro-García, Á. L. Muñoz Castañeda, M. Fernández Rodríguez, and M. V. Carriegos, "On Detecting and Removing Superfluous Redundancy in Vector Databases," *Mathematical Problems in Engineering*, vol. 2018, Article ID 3702808, 14 pages, 2018.
- [43] V. N. Gudivada, R. Baeza-Yates, and V. V. Raghavan, "Big data: Promises and problems," *The Computer Journal*, vol. 48, no. 3, pp. 20–23, 2015.
- [44] R. Baeza-Yates, "Big Data or Right Data," in *Proceedings of the 7th Alberto Mendelzon International Workshop on Foundations of Data Managements, CEUR Workshop Proceedings 1087*, p. 14, 2013.
- [45] S. García, A. Fernandez, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.
- [46] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [47] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [48] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, Washington, DC, USA, 1961.
- [49] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart, J. L. McClelland, and The PDP research group, Eds., vol. 1, MIT Press, 1986.
- [50] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, <https://arxiv.org/abs/1412.6980>.
- [51] J. Bergstra J, D. Yamins, and D. Cox, "Hyperopt: a Python library for optimizing the hyperparameters of machine learning algorithms," in *In Proceedings of the 12th Python in Science Conference (SciPy 2013)*, pp. 13–20, 2013.
- [52] M. Claesen, J. Simm, D. Popovic, Y. Moreau, and B. De Moor, *Easy Hyperparameter Search Using Optunity*, 2014, <https://github.com/claesnm/optunity>.
- [53] M. Lindauer, K. Eggenberger, M. Feurer et al., *SMAC v3: Algorithm Configuration in Python*, 2017, <https://github.com/automl/SMAC3>.
- [54] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt, "Dataset Spambase," UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml/datasets/spambase>]. Irvine, CA: University of California, School of Information and Computer Science, 2017.
- [55] Á. M. Guerrero-Higueras, N. DeCastro-García, and V. Matellán, "Detection of Cyber-attacks to indoor real time localization systems for autonomous robots," *Robotics and Autonomous Systems*, vol. 99, pp. 75–83, 2018.
- [56] M. Rami, T. L. McCluskey, and F. A. Thabtah, "An Assessment of Features Related to Phishing Websites using an Automated Technique," in *Proceedings of the International Conference For Internet Technology And Secured Transactions (ICITST 2012)*, pp. 492–497, IEEE, London, UK, 2012.
- [57] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Intelligent rule-based phishing websites classification," *IET Information Security*, vol. 8, no. 3, pp. 153–160, 2014.
- [58] R. M. Mohammad, F. Thabtah, and L. Mc-Cluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, vol. 25, no. 2, pp. 443–458, 2014.
- [59] L. Dhanabal and S. P. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, 2015.

- [60] “NSL-KDD Dataset,” https://github.com/defcom17/NSL_KDD, 2015.
- [61] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, “Calibrating Probability with Undersampling for Unbalanced Classification,” in *Symposium on Computational Intelligence and Data Mining (CIDM)*, IEEE, 2015, <https://www.openml.org/d/1597>.

Research Article

Practical Employment of Granular Computing to Complex Application Layer Cyberattack Detection

Rafał Kozik ¹, Marek Pawlicki ¹, Michał Choraś¹, and Witold Pedrycz²

¹*Institute of Telecommunications and Computer Science, UTP University of Science and Technology, Bydgoszcz, Poland*

²*Department Electrical and Computer Engineering, University of Alberta, Canada*

Correspondence should be addressed to Rafał Kozik; rkozik@utp.edu.pl

Received 12 October 2018; Revised 28 December 2018; Accepted 6 January 2019; Published 16 January 2019

Guest Editor: Fernando Sánchez Lasheras

Copyright © 2019 Rafał Kozik et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Network and information security are regarded as some of the most pressing problems of contemporary economy, affecting both individual citizens and entire societies, making them a highlight for homeland security. Innovative approaches to handle this challenge are undertaken by the scientific community, proposing the utilization of the emerging, advanced machine learning methods. This very paper puts forward a novel approach to the detection of cyberattacks taking inventory of the practical application of information granules. The feasibility of utilizing Granular Computing (GC) as a solution to the most current challenges in cybersecurity is researched. To the best of our knowledge, granular computing has not yet been widely examined or used for cybersecurity application purposes. The major contribution of this work is a method for constructing information granules from network data. We then report promising results on a benchmark dataset.

1. Introduction and Rationale

In the lifecycle of any technological advancement, the application domain and the target user base grow, shift and alter. However, the very technological advancements that contribute to the users' well-being can be twisted into submission by malevolent agents. Apart from the obvious uses, which are beneficial to the society, they can also be both the origin and the objective of a cybercrime.

It does not require exhaustive research to notice that the number and the severity of attacks targeted at the application layer is on the rise.

This situation is echoed by the rankings of cyberattacks, e.g., by OWASP [1], where SQLIA (SQL Injection Attack) and XSS (Cross Site Scripting) are on top of the list. Selected examples of critical systems which were successfully attacked through the application layer are the flight management system in Poland or the energy grid in Ukraine [2]. Therefore, we aim at proposing novel methods to effectively detect and counter the cyberattacks in the application layer. The research we are conducting aims at investigating the ability of the emerging methods of machine learning to counter these attacks. Several attempts have been made to utilize innovative

machine learning solutions to tackle the cyberattack problem. Those solutions include lifelong learning intelligent systems or deep learning [3, 4].

We hereby offer, in this paper, the implementation in the domain of cybersecurity of yet another emerging concept, namely Granular Computing.

Therefore, the major contribution of this paper is the application of a granular computing paradigm for cybersecurity.

We propose to extract information granules (which can also be known as tokens) in order to better understand the structure, semantics, and meaning of HTTP requests. Such an approach allows for effective anomaly detection, as is proven by the results we report.

In particular, the main contributions of this paper are

- (i) an innovative anomaly detection algorithm based on extracted information granules;
- (ii) a description of packet structure;
- (iii) an efficient method for request sequence encoding and classification; and
- (iv) evaluation of the proposed approach.

The remainder of the paper is structured as follows: Section 2 is focused on current trends and challenges related to cybersecurity and anomaly detection approaches, Section 3 discusses the application of granular computing in cybersecurity, and Section 4 presents an approach for extracting information granules from network data, while Section 5 presents the results of the conducted experiments. Conclusions are given thereafter.

2. Cybersecurity: A Quick Primer on Challenges and Trends

The rising numbers of attacks aimed at citizens, societies and even seemingly secure systems built for critical infrastructures are easily observable [2, 5]. The inadequacy of signature-based systems in cyberattack detection is one of the primary causes of this situation. Whenever new attacks are created, or even slightly modified families of malware are utilized, those systems, which constitute an industry standard, fall short of properly handling the hazard until new signatures are added. Anomaly detectors, which could tackle either new, or obfuscated attacks, are likely to generate false positives. There is a plethora of solutions that aim at countering attacks targeting the application layer. Many of those products use the signature-based approach. The signature-based category of cyber-attack detection includes Intrusion Prevention and Detection Systems (IDS and IPS) which use a predefined set of patterns (called signatures) to identify an attack. IPS and IDS are deployed to improve the security of computer systems and networks through detection (in the case of IDS) and blocking (in the case of IPS) of the cyberattacks.

Another class of solutions is called WAF (Web Application Firewall [6–8]). Those solutions are based on black and white listing approach. Classification of requests sent from client to server is performed. WAFs work on the basis of regular expressions, patterns, and signatures to detect cyberattacks. The predefined patterns (or rules) are typically compared to the content of incoming requests (either the header or the payload). A very popular IDS/IPS open product is called SNORT [9]. It is an open source project with a community of users who can freely modify the software, providing new rules to the Snort engine.

Injection attacks such as SQL or XSS occur when an improperly validated request containing malicious code is sent to an interpreter as part of a command or query. XSS flaws might occur when the application processes malicious data and sends it to a web browser without proper validation or escaping (the reformatting of ambiguous characters). By its nature, XSS allows attackers to execute scripts in the victim's browser. This leads to hijacking user sessions, defacing web sites, or redirecting the user to malicious sites.

The problem of developing effective signatures for such attacks is a highly complex one, as these attack vectors and patterns lend themselves to obfuscation. Another drawback of many WAF solutions is the fact that those are based on the preliminary (or previously learnt) assumptions regarding the request's structure (e.g., [6, 7]). However, different protocols utilizing HTTP as the transportation protocol are

characterized by different payload structures. For example, the structure sent via a plain HTML form is different from that of a GWT-RPC or a SOAP call. In such cases a number of pre-prepared signatures will not match a differently-structured payload, and in consequence, those attacks will not effectively be detected.

Another approach to HTTP traffic anomaly detection was presented in [10]. The authors applied a DFA (Deterministic Finite Automaton) to compare the requests described by means of tokens. The method based on the *n*-grams applied for anomaly detection and cyber-attacks detection has been presented in [11]. Other systems have also implemented algorithms to analyze the *n*-grams, for example with the use of statistical analysis [12], Self-Organizing Maps [13], Bloom filters [14], and a wide variety of different machine learning classifiers [15]. However, depending on the analyzed protocols and the methodology used to analyze *n*-grams, researchers report very diverse results for *n*-grams techniques. For instance, in [11] authors reported a high number of false positives for various state of the art methods. In contrast, authors in [16] reported a recognition rate of over 85%, while having quite low false positive rate of only 1%.

3. Information Granules in Cybersecurity

One of the most serious challenges of the methods and algorithms used in cybersecurity is being able to reach a correct understanding of the network data. Undeniably, cyberecosystems are quickly changing, as are the characteristics of the data. This fluctuation of properties produces uncertainty and difficulties in data partitioning/clustering. It is profoundly challenging to construct correct generalizations, rules and thresholds, and standard choices greatly decrease the efficiency of typical pattern recognition and anomaly detection algorithms. In addition, many of the used pattern recognition techniques do not try to incorporate or even take into account the semantics of the analyzed network data.

We propose the utilization of the practical elements of Granular Computing for anomaly detection as a solution of the preceding problems.

Granular computing refers to a general data analysis and recognition framework, incorporating data partitioning into so called information granules.

Granular Computing emerged as a general structure of data processing and knowledge discovery utilizing items called information granules. The very concept of granulation appeared independently in an array of fields, including fuzzy and rough sets or cluster analysis [17]. Granules are alignments of elements drawn together by their similarity, closeness or functionality [18]. A granule which occurs at a particular granularity level conveys a certain aspect of the modelled issue [19]. In situations with a certain degree of uncertainty granules can provide a convenient solution. This property can be translated into a certain economy when dealing with intricate problems. The tolerance for uncertainty bears a degree of resemblance to human thinking itself [18]. What follows is the utilization of Granular Computing in designing

real-life smart systems. The hierarchical nature of Granular Computing in conjunction with the basics of human reasoning conveyed in its premise makes it a perfect match for meaningful abstraction on various levels of detail [20].

Granules are essentially tiny parts of a larger construct, which describe a particular facet of that construct, when viewed from a particular level of granulation [21]. As an illustration of this principle one can consider how in cluster analysis objects can be grouped together based on similarity or distance functions. Since objects grouped in one cluster should exemplify a strong degree of similarity, clusters can be considered as granules [20]. Granules can be, thus, amassed into larger collections, which are then perceived as new, larger granules or divided into smaller pieces, which are more specific [21].

Ideally, the extracted information granules should comply with the Principle of Justifiable Granularity (PJG) [22]. PJG is a guideline for information granules to best comply with two competing requirements: justifiability and specificity. It stipulates that the constructed granules cover the relevant portion of the data, but should not be highly dispersed across the dataset. This can be achieved by selecting granules that resemble relevant semantics describing the data. Typical practical methods of granular computing are fuzzy sets [23], rough sets [24–26], and intuitionistic sets [27].

Granular computing allows for better data understanding through the incorporation of semantic aspects, similarities and uncertainties. Granular computing has been used recently for the analysis of spatiotemporal data [28], to concept-cognitive learning from large and multi-source data in formal concept analysis [29].

Granular Computing has been used to estimate a power plant's electrical power output [30]. The principles of granular computing are utilized to cluster the data with regard to the distance between granules, and the density of the newly constructed granules. Data prepared this way is fed into an Adaptive Neuro-Fuzzy Inference System (ANFIS). The procedure has been tested and proven to be a close fit [30].

Another application of granular computing allows for the recognition of faces that were surgically altered. Multiple levels of granules are created, some granules contain information about the whole face, some about specific regions, and some about the fine detail of specific features. Those granulation levels are then directed to classifiers [31].

The utilization of granular computing allowed for the creation of more accurate medical classifiers, which has an appreciable value in medical procedures. Data is granulated on multiple levels with regards to the Euclidean distance of the data points to the centroids. This distance constitutes the level of granulation, with higher granulation achieved through enlarging the distance. Data on multiple levels is then served to classifiers, which return a value between 0 and 1. The values are then introduced to a final stage classifier [32].

One of the recent research papers proposed a system introducing granular computing to financial markets. The proposed method of time series forecasting bested the state-of-the-art benchmark algorithms. Clustering is performed with an adaptation of the possibilistic fuzzy c-means algorithm, supplemented with the ability to process intervals. The

algorithm recursively gauges cluster centers as it brings in novel data [33].

Granular computing can be used to increase the calculating speed when solving the economic dispatch problem in order to reach the minimal running cost of a power grid. The described method breaks down a large power grid into smaller components which are treated as equivalent power stations. The process can be repeated, obtaining a finer level of granulation with each iteration. Determining optimal values on each level, before trying to reach a global optimum, makes the procedure both quicker and easier [34].

We apply our own methods to construct a practical solution based on information granules obtained from the network data.

To the best of our knowledge, granular computing has not yet been widely examined nor adapted for cybersecurity application purposes. One of the rare published papers is authored by Napoles et al. [35]. The authors addressed the problem of modeling and classification for network intrusion detection by utilizing a recently proposed granular model named Rough Cognitive Networks (RCN). The authors both proposed and defined RCN for detection of atypical (abnormal and potentially dangerous) patterns in the network traffic. RCN has been delineated as a sigmoid FCM (Fuzzy Cognitive Map). Map concepts denote information granules corresponding to the RST (Rough Set Theory) -based positive, boundary and negative regions of decision classes. Learning mechanisms for RCN are based on a self-adaptive Harmony Search algorithm. The proposed model has been evaluated with the NSL-KDD dataset (https://github.com/defcom17/NSL_KDD) and is shown to be a suitable and promising approach for detecting abnormal traffic in computer networks. Future work will address validation and further evaluation of the model based on real network traffic.

In the upcoming section we present our innovative methods for extracting information granules to counter cyberattacks in the application layer.

4. Proposed Approach for Extracting Information Granules from Network Data in Cybersecurity

Hereby we propose a new method for the clustering of multiple HTTP sequences utilizing a machine-learning classifier and granular computing approach.

As a way of grasping the semantics and the granularity, we use the information about the request structure and the statistical measurements of the structure content to detect anomalous behavior of untrusted sessions between client and server.

An overview of the proposed algorithm is presented in Figure 1, while a general overview of the granulation procedure utilized in our approach is presented in Figure 2.

As can be seen later in Section 5, the conducted experiments confirm the promising results and we can report that the proposed approach and method is competitive with other state-of-the-art solutions.

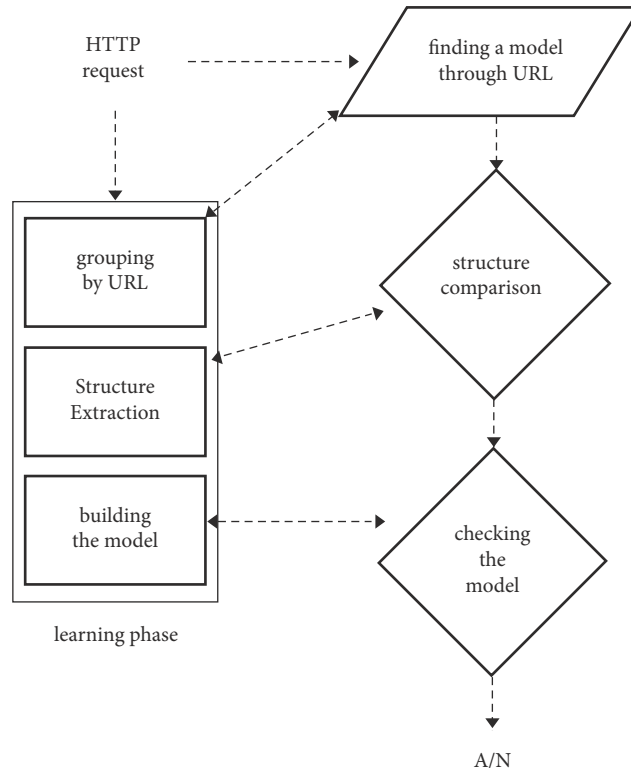


FIGURE 1: High-level overview of the proposed algorithm.

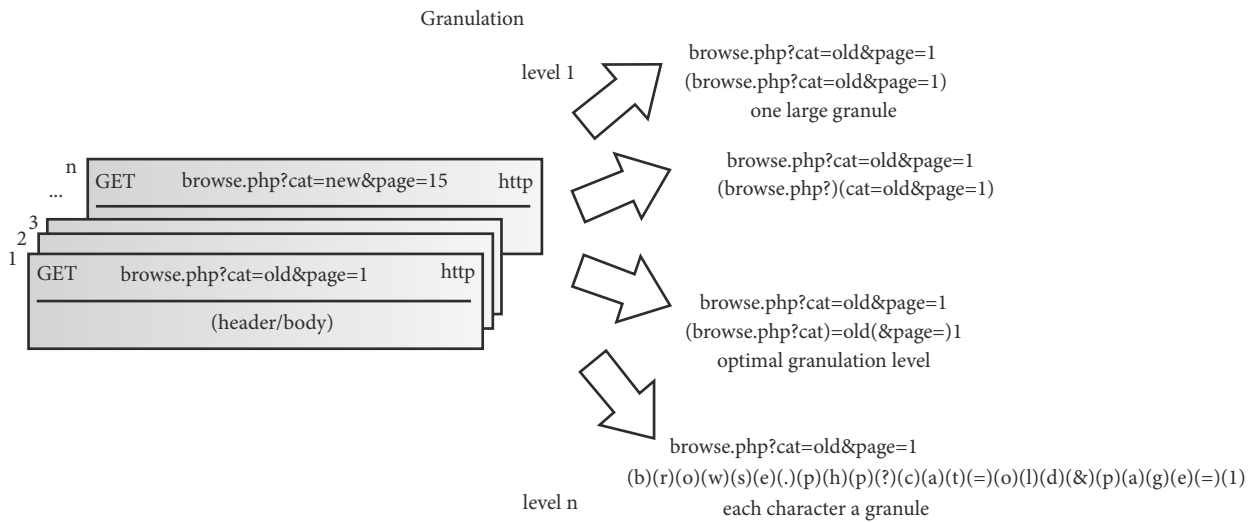


FIGURE 2: The overview of the granulation algorithm. The HTTP requests undergo a textual semantic segmentation of similar requests. The optimal level of granulation reveals data that can be used to calculate the feature vector.

The crucial advantage of the method proposed in this paper is that it is invariant to the underlying protocol stack (the method is protocol agnostic). In other words, it does not need to be tuned to any of the used protocols or application interfaces using HTTP for transport (e.g., the RESTful API, and SOAP). Hypertext Transfer Protocol (HTTP) is now frequently used due to its simplicity and reliability in assuring communication between computers in

distributed networks and allowing for increased usage of the web applications.

In our method, we apply a granular computing approach and extract information granules from HTTP requests.

An information granule in this approach is defined as a recurring sequence of information, which shares semantics for all the requests sent to the same resource or server (in Figure 2).

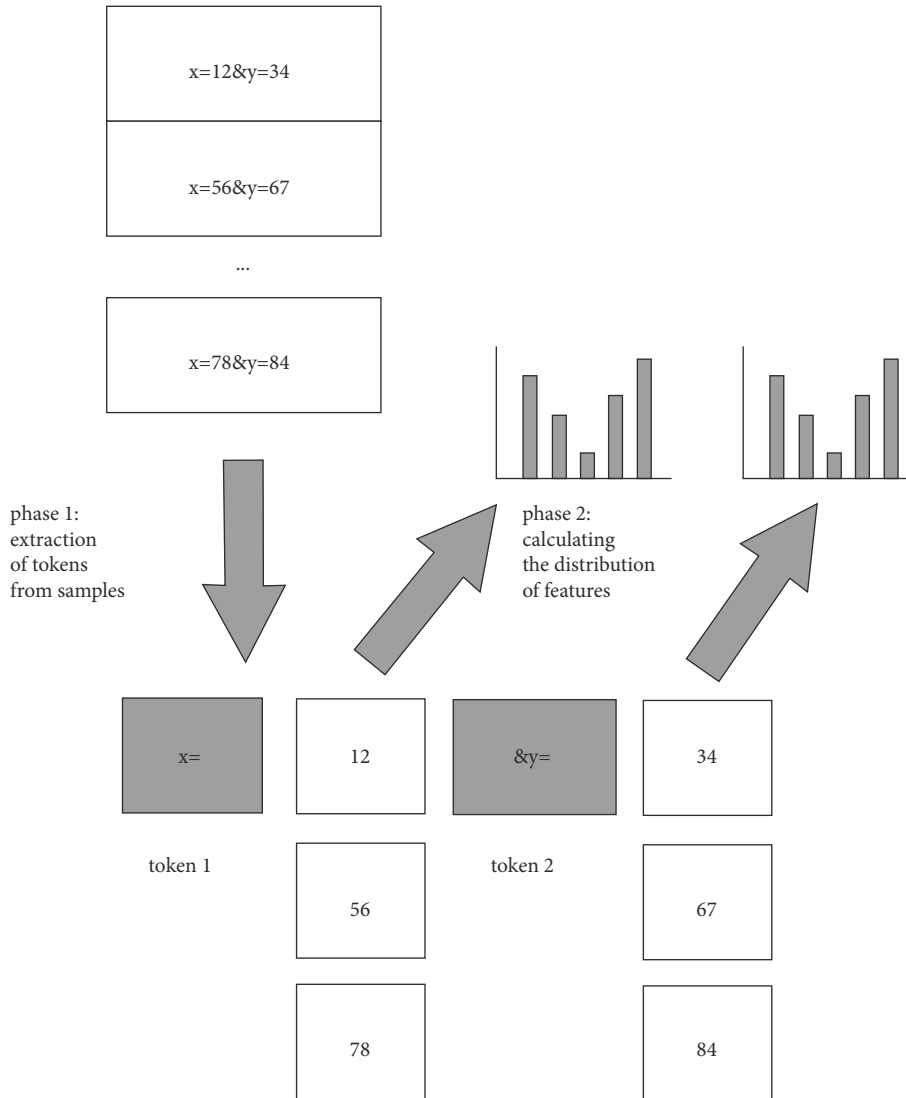


FIGURE 3: The parameter encoding approach.

The analysis of each single request can result in several granules (tokens). Our goal is to extract granules that will identify in the analyzed requests the delimiters of regions representing positions injected with cyberattacks or malware.

After these tokens are found, we have to describe the sequences between them by calculating their statistical properties (these tokens are represented on the optimal granulation level in Figure 2). We propose a two-step segmentation of HTTP requests. In the first step we divide the request space into smaller subsets to perform further calculations in the parallelized manner. In the second step we perform the actual segmentation of requests in order to identify their structure. The structure is described and represented by the extracted information granules. Those information granules (shaded boxes in Figure 3) allow us to identify the clusters in the feature space consisting of the feature vectors extracted from the granularized data.

Afterwards, when feature vectors are assigned to the appropriate clusters, we apply a machine-learning classifier. The classification task is to assign them either to the normal or anomalous class. In our experiments we have used various supervised methods. These have been explained in the experiments section.

In order to extract the granules we implemented the known LZW compression method (Lempel-Ziv-Welch [36, 37]). Firstly, we create the LZW dictionary D which allows us to transform the textual input (e.g., logs) into natural numbers (see the following):

$$D : word \rightarrow \{i : i \in N\} \quad (1)$$

The algorithm performs scanning within the set S in order to find the successively longer subsequences. This step is performed until the algorithm finds a sequence that does not yet belong to the dictionary. The found substring is added to the dictionary unless it is already represented. The described

```

Data: Set of HTTP payloads  $S$ 
Result: Dictionary  $D$ 
 $s =$  empty string
while there is still data to be read in  $S$  do
  |  $ch \leftarrow$  read a character
  | if  $(s + ch) \in D$  then
  | |  $s \leftarrow s + ch$ ;
  | else
  | |  $D \leftarrow D \cup \{s + ch\}$ ;
  | |  $s \leftarrow ch$ ;
  | end
end

```

FIGURE 4: The algorithm for creating the dictionary D .

procedure repeats until the entire dataset is scanned. The described algorithm is shown in Figure 4.

At the end of the processing a set S of HTTP payloads, the dictionary D , containing a list of sequences is created.

This dictionary has the form of an unordered list. Positions in the list can be taken only by one word. The dictionary D is implemented as a hash-table to achieve $O(1)$ lookup time.

Of course, as in LZW method, creating the dictionary D , also allows for compressing the data. The algorithm replaces words by numbers corresponding to the position of the sequence in the dictionary.

It is worth noticing that even after the single scan of the data, we can extract a reasonable number of candidates for appropriate information granules. Of course, it is not a trivial task, given the need to achieve balance between the specificity and justifiability. Another advantage is the compression of the data.

Still we have to further process the dictionary in order to obtain the collection of information granules. First condition is that we remove all the candidates that do not appear in all the samples used for structure extraction. In the next step we remove the sequences that also appear elsewhere as the sub-sequences of others.

The HTTP requests have the form of character sequences, whose lengths vary.

Moreover, single granule can appear at different positions in consecutive HTTP requests. Also the distance between granules may vary and, additionally, it happens that sometimes one granule is a subset of another information granule.

In our approach, we propose to use IDC (Idealized Character Distribution) method. We calculate it in the training phase from normal requests sent to a web application (the assumption is that the requests are normal, and manual inspection is needed in this step). The IDC is calculated as the mean value of all the character distributions. During the detection phase, we calculate and evaluate the probability that the character distribution of a sequence is an actual sample drawn from its ICD. Hereby, we use the well-known Chi-Square metric.

Equation (2) is used for computing the value of the Chi-Square metric $D_{chisq}(Q)$ for a sequence Q :

$$D_{chisq}(Q) = \sum_{n=0}^N \frac{(ICD_n - h_n(Q))^2}{ICD_n} \quad (2)$$

where N indicates the number of bins in the *histogram* (in our approach we used $N=9$), ICD the distribution established for all the samples, and $h()$ the distribution of the sequence Q that is being tested.

In order to calculate the distributions we count the number of characters that fall into each of the range of the ASCII table. We use the following ranges for this distribution count: $\langle 0,31 \rangle$, $\langle 32,47 \rangle$, $\langle 48,57 \rangle$, $\langle 58,64 \rangle$, $\langle 65,90 \rangle$, $\langle 91,96 \rangle$, $\langle 97,122 \rangle$, $\langle 123,127 \rangle$, and $\langle 128,25 \rangle$. The chosen ASCII ranges represent different types of signs such as numbers, quotes, letters, or special characters and in result represent well the distribution. The histogram that is used here will have 9 bins (due to 9 ranges).

5. Experiments

The CSIC10 benchmark dataset [38] was used for the experiments. It contains several thousand HTTP protocol requests which are organized in a form similar to the Apache Access Log. The dataset was developed at the Information Security Institute of CSIC (Spanish Research National Council) and it contains the generated traffic targeted to an e-Commerce web application. For convenience, the data was split into anomalous, training, and normal sets. The dataset contains approx. 36000 normal and 25000 anomalous requests. The anomalous requests are not always cyberattacks. They might refer to some anomalies (e.g., requesting unavailable resource), but more importantly they contain a wide range of application layer attacks, such as SQL injection, buffer overflow, information gathering, files disclosure, CRLF injection, XSS, server side include, and parameter tampering. To understand the results it is important to remember that the requests targeting hidden (or unavailable) resources are also considered anomalies. Some examples of such anomalies are requests for configuration files, default files, or session IDs in a URL (symptoms of an http session takeover attempt). Moreover, the requests with an appropriate format (e.g., a telephone number composed of letters) are also labeled as anomalies. As authors of the dataset explained, such requests may not have a malicious intent but nevertheless they do not follow the normal behavior of the web application. Still, there is no other appropriate, publicly available dataset for the web attack detection problem where we could reliably compare our results.

To verify our method based on information granule extraction we checked how our solution handles different quantities of learning data. As it is the typical case, in our experiments we also used the 10-fold approach.

The 10-fold cross-validation, also known as rotation estimation, is a model validation technique applied for evaluation of a machine learning model effectiveness in generalising the model to an unforeseen dataset. The method is utilised in spotting problems like overfitting or selection bias. It provides an overview of how the model might perform. In general, k-fold cross-validation achieves results less biased than other methods based on splitting the dataset into training and testing data subsets (e.g., repeated random sub-sampling). Cross-validation averages the results of all the

TABLE 1: True positive rate and false positive rate for different learning algorithms.

Algorithm	True Positive Rate	False Positive Rate
ICD	97,8%	8,1%
DS AdaBoost	93,7%	0,1%
RepTree	93,1%	0,3%
Random Forest	91,9%	0,7%

folds to come up with a more accurate assessment of a model performance.

In our case, the data used in learning and evaluation purposes is divided randomly into 10 parts (folds). One part of the data (10% of the entire dataset) is used for evaluation while the remaining 90% is used for training (e.g., establishing model parameters). The whole procedure is repeated 10 times, so each time a different part of the dataset is used for evaluation and a different part is used for training. The results for all 10 folds are averaged to yield an overall error estimate.

5.1. Comparison of Different Classification Techniques. In this experiment we have compared the effectiveness of various machine learning methods. As it was explained earlier first the granules are extracted and data for each granule is encoded used histograms. These histograms are used to train the algorithms.

It must be noticed that ICD is purely anomaly detection method and it can be trained on normal data. Other algorithms require both normal and anomalous data. As it is shown in Table 1 the ICD algorithm achieves the highest anomaly detection (TPR). However, the number of false alarms in contrast to other methods is relatively high.

5.2. Assessment of Training Dataset Size Impact of Classification Effectiveness. For each fold we deliberately picked only a subset of data to train the classifier. In such approach, we still have the same number of testing samples (common baseline for comparison) even if we have used only a fraction of the available training data. The entire 10-fold cross validation is repeated for different proportions of the training data, namely, 1%, 10%, 20%, and 100%. Results are presented in Table 2.

To obtain a better overview of the effectiveness of our method we calculated and present the ROC curves. The ROC curve for 300 learning samples is presented in Figure 5, while the curve for 32400 samples is presented in Figure 6.

6. Conclusions

In this paper we have proposed using the elegant theory of Granular Computing (GC) as the new approach to cybersecurity and network anomaly detection. The major contribution and innovation of this work is the first practical

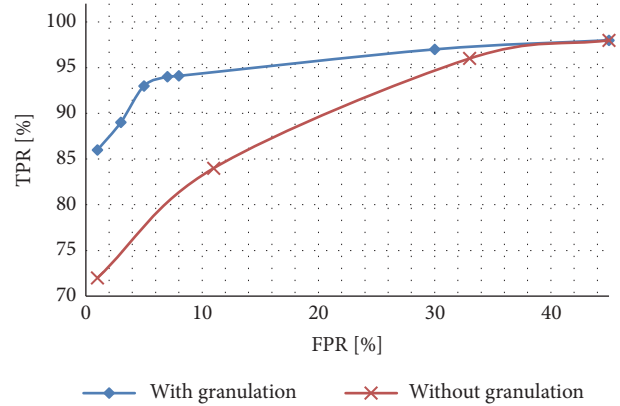


FIGURE 5: ROC curves for the Chi-Square metric comparing effectiveness of anomaly detection when granules for payload are extracted and otherwise. The experiment was conducted for an algorithm trained on 300 samples.

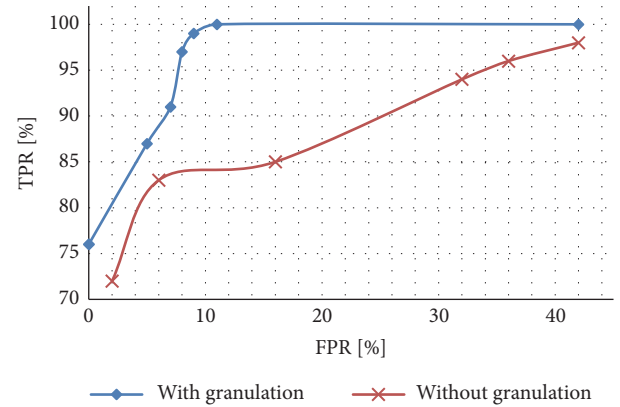


FIGURE 6: ROC curves for the Chi-Square metric comparing effectiveness of anomaly detection when granules for payload are extracted and otherwise. The experiment conducted for an algorithm trained on 32000 samples.

implementation of the method to extract information granules in order to detect cyberattacks. The proposed solution is designed to work with a typical HTTP-based, request-response web application. It can be described as an anomaly detection tool that receives HTTP requests, analyses their content, extracts information granules, and classifies those either as normal or as anomalies. We conducted the set of experiments on a standard benchmark dataset and typical evaluation scenarios. We report promising results, which demonstrate the efficiency of our approach and motivate our further research in applying Granular Computing to the cybersecurity domain (e.g., for other types of attacks in other layers).

Data Availability

The data used to support the findings of this study are included within the article.

TABLE 2: True positive rate and false positive rate for different numbers of learning samples.

TP Rate [%]	FP Rate [%]	Data Set Size	Number of samples
86,6	1,8	1%	300
95,6	5,8	10%	3000
96,9	6,8	20%	6000
97,7	8,1	100%	32400

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] "OWASP Top 10, 2013, OWASP project homepage," 2018.
- [2] M. Choraś, R. Kozik, A. Flizikowski, W. Hołubowicz, and R. Renk, "Cyber Threats Impacting Critical Infrastructures," in *Managing the Complexity of Critical Infrastructures*, R. Setola, V. Rosato, E. Kyriakides, and E. Rome, Eds., vol. 90, pp. 139–161, Springer International Publishing, Cham, Switzerland, 2016.
- [3] M. Choraś, R. Kozik, R. Renk, and W. Hołubowicz, "The concept of applying lifelong learning paradigm to cybersecurity," *Intelligent Computing Methodologies*, pp. 663–671, 2017.
- [4] D. Ariu, I. Corona, R. Tronci, and G. Giacinto, "Machine Learning in Security Applications," *Transactions on Machine Learning and Data Mining*, vol. 8, no. 1, 2015.
- [5] R. Kozik, M. Choraś, A. Flizikowski, M. Theocharidou, V. Rosato, and E. Rome, "Advanced services for critical infrastructures protection," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, no. 6, pp. 783–795, 2015.
- [6] "SCALP, Project homepage," <http://code.google.com/p/apache-scalp/>, 2018.
- [7] "PHPIDS, Project homepage," 2018.
- [8] "OWASP Stinger, Project homepage," <https://www.owasp.org/index.php/Category:OWASPStingerProject>, 2018.
- [9] "SNORT, Project homepage," <http://www.snort.org/>, 2018.
- [10] K. L. Ingham, A. Somayaji, J. Burge, and S. Forrest, "Learning DFA representations of HTTP for protecting web applications," *Computer Networks*, vol. 51, no. 5, pp. 1239–1255, 2007.
- [11] D. Hadžiosmanović, L. Simionato, D. Bolzoni, E. Zambon, and S. Etalle, "N-Gram against the Machine: On the Feasibility of the N-Gram Network Analysis for Binary Protocols," in *Research in Attacks, Intrusions, and Defenses*, pp. 354–373, 2012.
- [12] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *Recent Advances in Intrusion Detection*, vol. 3224, pp. 203–222, Springer, Berlin, Germany, 2004.
- [13] D. Bolzoni, S. Etalle, P. Hartel, and E. Zambon, "POSEIDON: A 2-tier anomaly-based network intrusion detection system," in *Proceedings of the 4th IEEE International Workshop on Information Assurance, IWIA 2006*, pp. 144–156, UK, April 2006.
- [14] K. Wang, J. J. Parekh, and S. J. Stolfo, "Anagram: a content anomaly detector resistant to mimicry attack," in *Recent Advances in Intrusion Detection*, vol. 4219, pp. 226–248, Springer, Berlin, Germany, 2006.
- [15] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "McPAD: a multiple classifier system for accurate payload-based anomaly detection," *Computer Networks*, vol. 53, no. 6, pp. 864–881, 2009.
- [16] K. L. Ingham and H. Inoue, "Comparing Anomaly Detection Techniques for HTTP," *Recent Advances in Intrusion Detection*, pp. 42–62, 2007.
- [17] T. Y. Lin and C. J. Liau, "Granular computing and rough sets," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds., pp. 535–561, Springer, Boston, Mass, USA, 2005.
- [18] L. A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy Sets and Systems*, vol. 90, no. 2, pp. 111–127, 1997.
- [19] A. Bargiela and W. Pedrycz, "The roots of granular computing," in *Proceedings of the IEEE International Conference on Granular Computing*, pp. 806–809, 2006.
- [20] Y. Yao, "A partition model of granular computing," *Transactions on Rough Sets I*, vol. 1, pp. 232–253, 2004.
- [21] Y. Y. Yao, "Granular Computing," in *Proceedings of the 4th Chinese National Conference on Rough Sets*, vol. 31, pp. 1–5, 2004.
- [22] W. Pedrycz and W. Homenda, "Building the Fundamentals of Granular Computing: A Principle of Justifiable Granularity," *Applied Soft Computing*, vol. 13, no. 10, pp. 4209–4218, 2013.
- [23] C. Wagner, S. Miller, J. M. Garibaldi, D. T. Anderson, and T. C. Havens, "From interval-valued data to general type-2 fuzzy sets," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 248–269, 2015.
- [24] F. Gong, M.-W. Shao, and G. Qiu, "Concept granular computing systems and their approximation operators," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 2, pp. 627–640, 2017.
- [25] Y. H. Qian, J. Liang, and C. Y. Dang, "Incomplete multigranulation rough set," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 40, no. 2, pp. 420–431, 2010.
- [26] M. I. Ali, B. Davvaz, and M. Shabir, "Some properties of generalized rough sets," *Information Sciences*, vol. 224, pp. 170–179, 2013.
- [27] B. Huang, Y. Zhuang, and H. Li, "Information granulation and uncertainty measures in interval-valued intuitionistic fuzzy information systems," *European Journal of Operational Research*, vol. 231, no. 1, pp. 162–170, 2013.
- [28] M. Song, W. Shang, L. Wang, and W. Pedrycz, "Analysis of spatiotemporal data relationship using information granules," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 5, pp. 1439–1446, 2017.
- [29] J. Niu, C. Huang, J. Li, and M. Fan, "Parallel computing techniques for concept-cognitive learning based on granular computing," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 11, pp. 1785–1805, 2018.
- [30] W. Sun, J. Zhang, and R. Wang, "Predicting electrical power output by using Granular Computing based Neuro-Fuzzy modeling method," in *Proceedings of the 27th Chinese Control and Decision Conference, CCDC 2015*, pp. 2865–2870, China, May 2015.

- [31] K. Vimitha and M. Jayasree, "Recognizing faces from surgically altered face images using granular approach," in *Proceedings of the 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 463–466, Chennai, India, March 2017.
- [32] M. Al-Shammaa and M. F. Abbod, "Granular computing approach for the design of medical data classification systems," in *Proceedings of the IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2015*, pp. 1–7, Canada, August 2015.
- [33] L. Maciel, R. Ballini, and F. Gomide, "Evolving granular analytics for interval time series forecasting," *Granular Computing*, vol. 1, no. 4, pp. 213–224, 2016.
- [34] X. Li and L. Fang, "Research on economic dispatch of large power grid based on granular computing," in *Proceedings of the 2016 IEEE PES Asia Pacific Power and Energy Engineering Conference, APPEEC 2016*, pp. 1130–1133, China, October 2016.
- [35] G. Nápoles, I. Grau, R. Falcon, R. Bello, and K. Vanhoof, "A Granular Intrusion Detection System Using Rough Cognitive Networks," *Recent Advances in Computational Intelligence in Defense and Security*, vol. 621, pp. 169–191, 2016.
- [36] T. A. Welch, "A Technique for high-performance data compression," *The Computer Journal*, vol. 17, no. 6, pp. 8–19, 1984.
- [37] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.
- [38] C. Torrano-Gimnez, A. Prez-Villegas, and G. Alvarez, "The HTTP dataset CSIC 2010," <http://users.aber.ac.uk/pds7/csic-dataset/csic2010http.html>.