Evaluating the impact of different Feature as a Counter data aggregation approaches on the performance of NIDSs and their selected features

ROBERTO MAGÁN-CARRIÓN*, *Network Engineering & Security Group, Department of Signal Theory, Communications and Telematics, CITIC-University of Granada, 18014, Granada, Spain.*

DANIEL URDA**, *Grupo de Inteligencia Computacional Aplicada (GICAP), Departamento de Digitalización, Escuela Politécnica Superior, Universidad de Burgos, Av. Cantabria s/n, 09006, Burgos, Spain.*

IGNACIO DIAZ-CANO†, *Applied Robotics Group, Department of Automatic, Electronic, Computer Architecture and Com. Net. Engineering, University of Cádiz, 11519, Puerto Real, Cádiz, Spain.*

BERNABÉ DORRONSORO††, *Graphical Methods, Optimization & Learning (GOAL) Group, Department of Computer Engineering, University of Cádiz, 11519, Puerto Real, Cádiz, Spain; School of Computer Science, Faculty of Engineering, The University of Sydney, 2008, Darlington, NSW, Australia.*

Abstract

There is much effort nowadays to protect communication networks against different cybersecurity attacks (which are more and more sophisticated) that look for systems' vulnerabilities they could exploit for malicious purposes. Network Intrusion Detection Systems (NIDSs) are popular tools to detect and classify such attacks, most of them based on ML models. However, ML-based NIDSs cannot be trained by feeding them with network traffic data as it is. Thus, a Feature Engineering (FE) process plays a crucial role transforming network traffic raw data onto derived one suitable for ML models. In this work, we study the effects of applying one such FE technique in different ways on the performance of two ML models (linear and non-linear) and their selected features. This is the Feature as a Counter approach. The derived observations are computed from either with the same number of raw samples, (batch-based approaches) or by aggregating them by time intervals (timestamp-based approach). Results show that there is no significant differences between the proposed approaches neither

*E-mail: rmagan@ugr.es
**E-mail: durda@ubu.es
†E-mail: ignacio.diaz@uca.es
††E-mail: bernabe.dorronsorodiaz@sydney.edu.au

Vol. 32, No. 2, © The Author(s) 2024. Published by Oxford University Press.

Advance Access published 16 March 2024 https://doi.org/10.1093/jigpal/jzae007

in the performance of the models nor in the selected features that validate our proposal making it feasible to be widely used as a standard FE method.

Keywords: Machine learning, feature engineering, feature selection, NIDS, cybersecurity, network security, information security.

1 Introduction

The Cisco Annual Internet Report (2018–2023) [1] forecasts a huge increment on the number of Internet connected devices. It foresees more than 29 billions of connected devices, over three times the global population. Moreover, this will be boosted by the real deployment of new communications technologies e.g. 5G or 6G, allowing heterogeneous devices from IoT (Internet of Things) ecosystems for an easy and affordable Internet access. Mainly because of the previous (inter-) connectivity capabilities, new applications and services can be conceived. However, from the point of view of the security, this scenario increases the likelihood of being threaten by malicious actors thanks to the many new ways and entry points to perform and execute attacks. Even more, according to the ENISA Threat Landscape 2020 report (ETL2020) [14], the sophistication of threat capabilities was seriously increased in 2019, where over 200,000 daily new variants of malware targeting diverse objectives were detected. To deal with them, security solutions are needed to prevent, detect and react against malicious attempts whichever their goals are. For that, the Intrusion Detection Systems (IDSs) are one of the most relevant defense lines nowadays, specially to provide early attack detection and thus reducing its impact on the target system.

IDSs make use of different techniques and methods for modeling and classifying heterogeneous data for diverse application contexts. In particular, Machine Learning (ML)-based IDSs [3] are popular techniques for fighting against malicious security events in networks and systems, where large amounts of data need to be processed in real time. The evaluation of this kind of systems relies on the use of pre-defined and -built datasets usually gathered from the application scenario. Obviously, inherent dataset characteristics influence algorithms or techniques upon which the IDS are based [6]. Apart from that, the IDS performance evaluation needs of relevant, some of them mandatory, pre- and post-processing task thus building the so called ML evaluation pipeline.

This work focuses on two of the most relevant steps for assessing ML-based IDS as it was stated in [22]: the Feature Engineering (FE) and the Feature Selection (FS) processes. The first one transforms and adapts raw information into useful derived data to feed the ML model accordingly. Hence, the FE method must be able to deal with variables of different types, as numerical and categorical ones. The second one provides meaningful data to the ML model according to the objective of the IDS. In regards to the FS, removing useless features improves the detection performance and save computation and storage resources making models feasible to be deployed in production environments. Specifically, we assess in this work the impact of the FE process in both the number of selected features and the system performance. For that, the *ff4ml* framework [22] will be used for a fair evaluation of supervised classification ML-based Network IDSs (NIDSs). The authors of the work rely on the use of the Feature as a Counter (FaaC) as FE approach, which has been tested an validated for network anomaly detection systems in previous works [10, 11, 24]. Briefly, FaaC transforms raw dataset features into new variables, which are counters of the originals. To do this, a 1-minute aggregation time interval is used to transform raw dataset observations into new derived ones. Both the number of derived variables (counters of the original ones) and the time aggregation window can be customized on demand, making it an extremely versatile and adaptable

tool. Moreover, it homogenizes and structures the ML-based model input data from raw and probably unstructured data (including variables of different types), making it suitable to be used with different network data sources to combine them for achieving robust NDISs [24], for instance. However, FaaC was traditionally restricted to be used with timestamp-based datasets e.g. the UGR'16 [20] dataset. Because of that, the use of FaaC is limited to the previous kind of datasets making it useless to process timestamp-less datasets like the well-known and widely used NLS-KDD network dataset [31].

The contributions of the present work can be summarized as follows:

- A new aggregation technique of the raw observations dataset for the FaaC FE process that in this proposed case, unlike its traditional use, now does not use timestamp information is proposed. Two approaches are considered with different batch observation sizes.
- The influence of using or not timestamp information in the generation of the derived datasets on two well-known ML models for NIDS is studied.
- An extension of the results provided in [23] to analyse the impact of the proposed methodology on another timestamp based dataset: the UNSW-NB15 dataset.
- An exhaustive study on the most meaningful features, according to the performance of the ML models, and how the different FE approaches impact on the feature selection.

Experimental results show that no significant differences can be found, neither using timestamp-based nor batch-based FE aggregation approaches, both on the detection performance and the selected features. Therefore, this study makes feasible the use of FaaC FE technique on timestamp-less datasets thus allowing comparison against state-of-the-art NIDS FE methods proposed on those datasets. Moreover, it provides a flexible and sophisticated way to homogenize the input information of a model making possible a fair comparison among NIDS systems since there is no consensus about which FE procedures should be used yet and why in the research community.

The rest of the paper is organized as follows. Section 2 introduces and describes works about how the research community tackle the problem addressed here. Section 3 describes ML based methods and the considered datasets to evaluate them. Then, in Section 4 the FaaC fundamentals is introduced as well as the proposed batch-based aggregation technique to deal with timestamp-less network datasets. In this section, evaluation strategy to assess the performance of ML models is introduced too. Finally, results and discussion are addressed in Section 5 and some conclusions and future works are outlined in Section 6.

2 Background

This section presents and discusses some of the most state-of-the-art interesting articles dealing with the detection of network attacks using ML-based techniques, especially focused on feature engineering methods and how to transform, in some way, raw network data onto meaningful derived data.

Thus, the authors in [16] propose a method for classifying attacks based on the hybrid combination of two swarm intelligence algorithms. For the division of the training dataset, they use the FCM algorithm [5] on the original dataset, creating several subsets of data. Other similar works as the one introduced in [30] also use swarm intelligence algorithms together with genetic algorithms for the classification of attacks in an intrusion detection system. In this case, a three tier system is proposed where, in the first tier, a careful process is applied to choose the most appropriate features for anomaly detection. In the second tier, the classification algorithms are used while, in the third tier, the authors validate the model using a 10-fold cross validation approach. In the study [17] the authors

do not apply any Feature Engineering techniques nor do they mention anything about how they pre-process the raw data. The training and testing processes are carried by selecting batches of a number of observations. The batch size is calculated by applying the Optimum Allocation scheme method on the original dataset that determines the size of each training subgroup using a 99% confidence interval. None of these methods consider the use of a FE technique, compromising their applicability to other datasets with different structure.

The authors in [27] generate a new dataset useful for evaluating NIDSs. To do this, they have used a Characteristics Engineering method based on the CICFlowMeter tool, a public domain software from the Canadian Institute of Cybersecurity [19] for synthetic network traffic generation. To select the most relevant features, they use the RandomForestRegressor method from the scikitlearn library. The authors do not perform any previous pre-processsing task for training and/or testing the models they consider. Additionally, the authors in [28] propose an alternative workflow for the detection of attacks in a classic dataset, the ISCX-IDS2012. It consists on the execution of a data pre-processing and transformation task based on the use of the well-known SMOTE technique to be afterwards followed by a correlation based feature selection process. Finally the ML-based models are evaluated. The data pre-processing and transformation performed is limited to simply removing observations with incomplete/incorrect values and applying data normalization. Different from the FaaC approach, what the authors propose does not provide a way to homogenize raw data onto derived one.

In [18] the authors propose a NIDS that is validated through the UNSW-NB15 dataset. To do this, they perform a data pre-processing phase with two consecutive steps. First, they reduce the original dataset by creating fifteen clusters according to the Silhouette coefficient and then select those with the most presence of attacks. Second, the authors make a reduction of the original feature space by using a information gain approach. Therefore, no FE method is applied.

Belouch *et al.* evaluate in [4] the detection performance of ML-based NIDSs with UNSW-NB15 dataset. There is no data transformation or pre-processing and no feature selection is performed. They use the Apache Spark, a big data processing tool, to implement the system. Their aim is to compare four well-known ML-based algorithms implemented in the MLib library included in the Apache Spark suite.

A deep learning approach for network anomaly detection is presented in [32], where the authors apply feature engineering techniques to improve the detection performance. Two well-kown datasets are used: the NSL-KDD and UNSW-NB15. Before model training, they perform data pre-processing. They apply the Probabilistic Mass Function (PMF) method to convert nominal features into numeric ones. After that, a data normalization process is carried out. Then, the network is trained, where a batch of 100 observations is chosen in each iteration. Similarly, the authors in [2] introduce several improvements to the UNSW-NB15 dataset making deep learning based NIDSs better in terms of detection performance. First, a data sanitization is done. Next, a normalization of the data is carried out using the min-max approximation. The models training is done afterwards by feeding them with batches of 100 raw observations. The two proposals perform some kind of FE on the data, but the FE applied do not generate homogenized data, when applied to different datasets.

A novel study is presented in [33]. It provides a deep learning based image recognition system, using transfer learning techniques, to detect network attacks and classify network attacks. To this end, it uses the already known dataset UNSW-NB15. A feature data normalization is initially performed to afterwards transform them into four image channels. The authors carry out several experiments but no feature engineering and feature selection processes are used.

A deep learning based NIDS is also proposed by the authors of [26], based on the use of a bi-directional long short term memory with drop out, dense output layers and a novel time distributed

layers. The proposed workflow does not show the application of any feature engineering method. However, they normalize and pre-process the data but no details about how they performed these were provided. They test the system by combinations of different activation functions in two of known network data sets the UNSW-NB15 and the KDDCUP-99, concluding that SoftMax lead the system to better performance.

Authors of [36] present a novel NIDS that combines the classification paradigm of well-known supervised and unsupervised ML-based algorithms. They employ several well-known datasets, used in other network attack learning studies, and a 10-fold cross-validation approach. Before that, they carry out a feature selection procedure using the information retrieval method. The authors conclude that the proposed system reduces miss-classifications and improves attack detection in six of the seven datasets used in the experimentation. No FE technique is applied in this work.

In summary, many articles have been found in the literature where the authors propose different and heterogeneous feature selection approaches. However, current NIDSs solutions do not often include FE processes in the evaluation pipeline or, at least, the proposed solutions are not enough elaborated and rely on the use of only data pre-processing approaches like the standard normalization. Additionally, no strong enough efforts can be found in the literature first, to provide sophisticated and flexible FE approaches like the one introduced here and second, to evaluate them in terms of their impact on the NIDS performance and their kind and number of selected features.

3 Datasets and methods

Throughout this section, the data sets and network models used in this study are introduced and described.

3.1 Datasets

3.1.1 UGR'16 dataset UGR'16 [20] is a dataset built from real anonymized network traffic flows (NetFlow) collected over four months. These data have been collected within a Spanish Internet Service Provider (ISP). The dataset was divided by its authors into two different parts. One of them is CAL, and it is made up of network flows captured from March to June 2016. The other group, TEST, was collected the last week of July to August of the same year. Throughout this entire period, network attacks were planned, created and synthetically launched in conjunction with normal network traffic captured on the ISP's own infrastructure. To achieve this purpose, updated tools were used to generate the attacks, which covered attacks that were difficult to detect and, in other cases, harmful, such as *port scanning*, or *low and high rate DoS*. Other types of network attacks, such as *Botnet*, were painstakingly added to the UGR'16 dataset using network communication flow traces from other existing network datasets. Additionally, the authors went so far as to label attacks that were detected as network anomalies by state-of-the-art detectors, after doing an exhaustive analysis of the traffic. Thus, *SSH scanning*, *spam campaigns* and *UDP port scanning* were identified. Lastly, some streams were tagged as *blacklist*, if their IP was found in blacklists open sources.

Table 1 shows the number of traffic flows and their corresponding percentage, according to the distribution of each of them in the TEST part of the data set. It was expected that within the NIDS problems there would be differences in the number of observations of the different attacks, between those that come from attacks and those that belong to normal network traffic (*Background*). Thus, the attack with the highest number of observations belongs to the *spam* dataset, and it represents only 1.96% of the total traffic flows. It is true that all the attack samples exist in both the CALC and

TABLE 1. Number of raw observations per class in TEST partition of the UGR'16 dataset.

Class	Background	Blacklist	Botnet	DoS	SSH scan	Scan	Spam	UDP scan
raw obs.	$\sim 4,000M$	$\sim 18M$	$\sim 2M$	$\sim 9M$	64	$\sim 6M$	$\sim 78M$	$\sim 1M$
%	97.14	0.46	0.04	0.23	~ 0	0.14	1.96	0.03

TABLE 2. Number of raw observations per class of the UNSW-NB15 dataset.

Class	Background	Generic	Exploits	Fuzzers	DoS	Reconn.	Shellcode	Analysis	Backdoor	Worms
raw obs.	$\sim 2,21M$	$\sim 215K$	~ 44	$\sim 5K$	$\sim 16K$	$\sim 2K$	223	$\sim 3K$	$\sim 2K$	174
%	87.35	8.48	1.75	0.2	0.64	0.7	~ 0	0.1	0.09	~ 0

TEST groups, but in the latter all the network attacks exist. This is what leads us to choose the TEST partition to carry out the proposed study.

3.1.2 UNSW-NB15 dataset UNSW-NB15 [25] is a public dataset that comprises raw network traffic. To generate it, the authors use the IXIA 'PerfectStorm' tool created by the CyberRange Laboratory of the Australian Center for Cybersecurity (CASS). Through this tool the authors built a network architecture composed by various routers and firewalls that, in turn, interconnect several servers and clients too. A total of 47 characteristics of the studied gathered network traffic (PCAP files) were obtained. As a result 2,540,044 of traffic were collected and were publicly available in the form of of four CSV files.[1] Nine different attacks were recognized and labeled: *Analysis (Port Scanning)*, *Fuzzers*, *DoS*, *Backdoors*, *Generic*, *Exploits*, *Shellcode*, *Reconnaissance* and *Worms*. Table 2 shows the distribution of the above mentioned classes both in number of traffic flow observations and in percentage. As we observed in the UGR'16 dataset, there are a large number of *Background* traffic flows compared to other classes of attacks. Among the attacks, *Generic* is the one with the highest number of number of observations.

3.2 Methods

3.2.1 Feature selection algorithm: LASSO Commonly, ML methods tend to perform better in scenarios with many more samples than features (i.e. when $N >> P$). Besides, in the AI/ML community it is well-known that models' performance decrease when irrelevant or noisy features are used in the training process. Therefore, feature selection arises as a natural and necessary step in order to select the best subset of features that could potentially lead to optimal predictive performance. In the literature, there are several techniques published categorized as wrapper, filter and embedded methods), which could be used for this purpose. Among the first category of FS methods, forward feature selection, backward feature elimination, recursive feature elimination or evolutionary-based methods are the ones commonly used, although they are typically time consuming and require high computing resources. On the opposite hand, filter methods such as the Pearson's correlation coefficient, ANOVA, Chi-square test, mutual information or information gain tend to be much faster but rarely outperforms wrapper ones. Finally, embedded methods such as

[1] UNSW-NB15: https://researchdata.edu.au/unsw-nb15-dataset

the Least Absolute Shrinkage and Selection Operator (LASSO) family of methods (LASSO, Ridge and the Elastic Net), decision trees or Random Forest are methods that take advantage of the main characteristics of wrapper and filter ones, i.e. they tend to perform well by selecting the most relevant features in a reasonable time and with no need of high computing resources.

However, making an extensive comparison of feature selection methods is not within the scope of this work. In this sense, we decided to use the LASSO [15] to deal with high dimensional settings based on the authors' previous experience [23, 24, 34, 35]. Besides, LASSO is a simple embedded feature selection method, fast in terms of execution time, supervised (i.e. it takes into account the class in order to select the most relevant features), and that has been successfully used for this purpose to address problems in several domains. Therefore, the features selected by the LASSO model are used to train the classifiers by solving the minimization problem depicted in Equation 1, which is the classic logistic regression problem that now includes a regularization term (l_1-penalty):

$$\hat{\boldsymbol{\beta}}_{\lambda} = \arg\min_{\boldsymbol{\beta}} \ ||\boldsymbol{y} - f(\boldsymbol{\beta} X^T)||_2^2 + \lambda ||\boldsymbol{\beta}||_1 \ . \tag{1}$$

LASSO has one single hyper-parameter, λ, which helps to control the strength of the feature selection procedure: the lower the value of λ, the weaker the regularization would be (i.e. less noisy or irrelevant features will be eliminated) and vice versa.

3.2.2 Machine learning models: linear regression and random forest Let us assume that the *ff4ml* framework [22] has been employed onto a raw network dataset and the corresponding derived dataset has been obtained. In this setting, a derived dataset $\mathcal{D} = \{\boldsymbol{x_i}, y_i\}_{i=1}^n$ of n samples can be used to train different supervised ML methods, since for each i-th sample, the input feature vector x_i that maps onto the given class y_i is available. This work considers two well-known ML models, a multinomial logistic regression model and a Random Forest, which are a good representation of the main ML categories (linear and non-linear methods) allowing us to analyse their performance on this problem. While multinomial logistic regression is the simplest classifier possible that can provide us a baseline model, Random Forest is a well-known ensemble method that allows capturing non-linear relationships in data and that addresses the same problem from a different perspective, thus providing a good evaluation framework from the ML point of view.

- Multinomial logistic regression (LR). For a binary classification problem, logistic regression is the simplest linear model available that aims at learning the class (y_i) as a linear combination of the input feature vector x_i, as depicted in Equation 2:

$$y_i = f(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) \ , \tag{2}$$

  where $f(\boldsymbol{\beta} X^T)$ is the logistic or sigmoid function. In this scenario, the minimization problem shown in Equation 3 is solved in order to learn the parameters' vector $\boldsymbol{\beta}$, which optimize the classifier performance:

$$\boldsymbol{\beta^*} = \arg\min_{\boldsymbol{\beta}} \ ||\boldsymbol{y} - f(\boldsymbol{\beta} X^T)||_2^2 \ . \tag{3}$$

  An extension of this simple classifier allows to deal with multi-label classification tasks where more than 2 labels or categories are available. For this purpose, a common approach consists of training k logistic regression models, where each of these models specializes in separating or classifying one individual label from the rest. This approach is known as *One-vs-All*. Since this extension provides an output probability vector of size k where probabilities do not sum up to 1, there is a need to use the *softmax* function depicted in Equation 4 in order to normalize

this probability vector and to compute the overall prediction of the LR model:

$$Pr(y_i = k|X) = \frac{Pr(y_i = k|X)}{\sum_{z=1}^{k} Pr(y_i = z|X)} \ , \tag{4}$$

where the final class predicted by the model, for a given input feature vector, would be the one with higher probability in this normalized probability vector.

- Random Forest (RF). It is a bagging ensemble of decision trees that allows capturing non-linear relationships in the data. In particular, each single decision tree is trained using a specific view of the data [7], which means that each decision tree uses a random subset of samples to build the model, at the same time that each node within the tree employs a random subset of input features to determine the feature to use to expand the tree and which maximizes the entropy. The final RF prediction for an input feature vector x_i is calculated as the average (or weighted average according to the performance of each single decision tree on the out-of-bag samples), or the majority vote of the individual output provided by the bag of decision tree models.

4 Experimental design

As it has been mentioned in the previous section, ML techniques for NIDSs cannot be applied directly on the raw network datasets, but they need to be transformed onto derived datasets, suitable for the application of these techniques. In [23], FaaC was proposed as a suitable FE technique to generate the derived dataset. Particularly, it was used in UGR'16 for aggregating all raw samples in a 1-minute time interval onto an only one derived observation. This way, the whole derived dataset is built repeating this process every minute in the original dataset. In UGR'16 dataset, one minute interval comprises around 85, 000 samples, in general, but this number can be considerably reduced for networks with low traffic, to the point of being insignificant for a meaningful classification. Aggregating data following a batch-based technique allows generating meaningful derived datasets, where all samples are generated from the same number of observations in the original dataset.

Table 3 shows the relationships among selected raw features and derived features (counters). Last two columns in the table show how many derived features are extracted from one specific raw variable and the derived features description. FaaC counts the number of times each feature category (within brakes in the table) appears in a batch of raw observations from the specific raw feature. For instance, the derived feature `dport_http` will contain how many network HTTP connections have been performed to a web server. A total of 134 derived features are created from the original ones. It is worth mentioning that this method and the chosen set of derived features have been demonstrated to be useful for NIDSs [8, 9, 12, 13, 21, 22, 24], and therefore it has been adopted here too. Further details can be found in [22].

As previously explained, our previous work [23] explored the possibility of splitting the data into a fixed number of observations, regardless the timestamp, and its effect on the detection of attacks of the system was studied for UGR'16 dataset. Here, we extend that work firstly by considering another well-known timestamp-based dataset as USNW-NB15, and secondly by analysing the impact of the aggregation approaches on the feature selection process. Three different settings will be considered for the two datasets:

- *Timestamp-based*. The derived dataset is built by aggregating the samples of the raw dataset in 1-minute time intervals. In the case of UGR'16, around 40, 000 observations are generated, while this number is reduced to 1, 441 for USNW-NB15.

TABLE 3. Raw and derived features relationships.

Raw features	**Derived**	
	#	**Features**
Src. IP	3	`srcip_`{private, public, default}
Dst. IP	3	`dstip_`{private, public, default}
Src. port	52	`sport_`{http, smtp,..., reserved}
Dst. port	52	`dport_`{http, smtp,..., reserved}
Protocol	5	`protocol_`{tcp, udp,..., other}
Flags	6	`tcpflags_`{ACK, SYN,..., URG}
ToS	3	`tos_`{0, 192, other}
# packets	5	`npckts_`{verylow, low,..., veryhigh}
# bytes	5	`nbytes_`{verylow, low,..., veryhigh}

TABLE 4. Class distributions (%) for UGR'16 dataset per derived dataset corresponding to the application of the FaaC procedure using the three described settings: timestamp- and batch-based ones.

Class	timestamp-based ($\sim 40,000$ obs.)	batch-based (10,000 obs.)	batch-based (20,000 obs.)
Background	63.65	55.97	58.23
Spam	33.76	39.86	38.23
Botnet	1.26	1.66	1.34
DoS	0.88	1.41	1.3
Scan	0.37	0.79	0.62
SSH scan	0.06	0.24	0.12
UDP scan	0.02	0.08	0.05

- *Batch-based (*10,000 *observations)*. This approach fixes the number of observations of the derived dataset to 10,000. Each observation has 395,082 and only 254 raw samples for UGR'16 and UNSW-NB15 datasets, respectively.
- *Batch-based (*20,000 *observations)*. It differs from the previous one in the number of observations generated for the derived dataset, which is set to 20,000 in this case. Therefore, batches are generated from 197,541 raw samples for UGR'16 and 127 for UNSW-NB15.

Tables 4 and 5 show the class distribution on the three derived datasets, for UGR'16 and USNW-NB15, respectively. They reflect minor differences on the class distributions, despite the fact that the final size, in terms of the number of observations, is significantly different across the datasets (implying the use of different batch sizes to compute the aggregated counter variables). Therefore the different derived datasets result as similar problems for the ML techniques. However, smaller datasets are expected to be more challenging for the ML models, which typically benefit from a high number of samples.

TABLE 5. Class distributions (%) for USNW-NB15 dataset per derived dataset corresponding to the application of the FaaC procedure using the three described settings: timestamp- and batch-based ones.

Class	timestamp-based (1,441 obs.)	batch-based (10,000 obs.)	batch-based (20,000 obs.)
Background	44.14	35.49	35.59
Generic	26.44	38.06	35.82
Exploits	20.26	16.80	16.85
Fuzzers	8.88	7.14	7.56
DoS	0.28	1.02	1.75
Reconnaissance	–	1.47	2.2
Shellcode	–	–	–
Analysis	–	0.01	0.07
Backdoor	–	0.01	0.08
Worms	–	–	–

In this work, we used *ff4ml* framework on each of the derived datasets considered (generated using FaaC) to evaluate the performance of the selected ML models. In order to find concluding results, we performed a 5-fold cross-validation strategy (5-CV) in the training and testing processes of the models. For that, we followed the methodology described next on each of the six considered derived datasets. First, the dataset is splitted into five non-overlapped blocks (also called folds). They are generated in a way that it is guaranteed that they have the same size and class distribution. Then, the training process of the models is done five times on these folds, using one different fold for testing each time, and the remaining four ones for training the model (as described in Section 3.2). This process of splitting the dataset into folds and applying the 5-CV procedure is repeated 20 times with different partitions. This is done to avoid any non-representative result produced by the randomness of the partitioning process or the presence of some bias that could be introduced in the partition procedure.

The evaluation of the models was made using the Area Under the Receiving Operation Characteristic Curve (*AUC*) performance metric. This is a well-known metric widely used in ML literature, and suitable for unbalanced multi-classification problems, as it is the case of the considered one. We also calculate the *Weighted avg.* metric [22], computing a weighted average that considers the relative presence of each class, together with the accuracy of the model on that class.

With respect to the parameters of the models considered in this work, and to allow reproducing the results shown in Section 5, the search of the optimal number of trees employed in RF is set on the range of [500,600], while the maximum number of features to use on the split of each node is set on the range of [2,16]. The optimal values for these hyper-parameters are found using Bayesian optimization [29] strategy. Different from some other well-known hyper-parameter selection strategies like Grid or Random Search, Bayesian optimization assures a pseudo-optimal hyper-parameters setting in a few number iterations thus saving both computation resources and execution time. On the other hand, LR does not have any hyper-parameter to be tuned and the λ hyper-parameter of the LASSO model is learned from data using nested cross-validation through the R implementation provided in the *glmnet* package [15].

5 Results and discussion

Through this section we assess the impact of the different data aggregation approaches, timestamp- and batch-based ones, on the two of the most concerned and relevant aspects for building and evaluating NIDSs according to [22]: the model performance and the feature selection.

5.1 Performance analysis

In order to corroborate the viability of the batch-based aggregation approaches in comparison to the timestamp-based one, we carried out several experiments where the performance of the selected ML models (Section 3.2) is evaluated when considering two widely used network datasets: the UGR'16 and the UNSW-NB15. For that, we follow the methodology proposed in [22] comprising what the authors consider essential steps to be taken into account in whatever the ML-based NIDS evaluation pipeline.

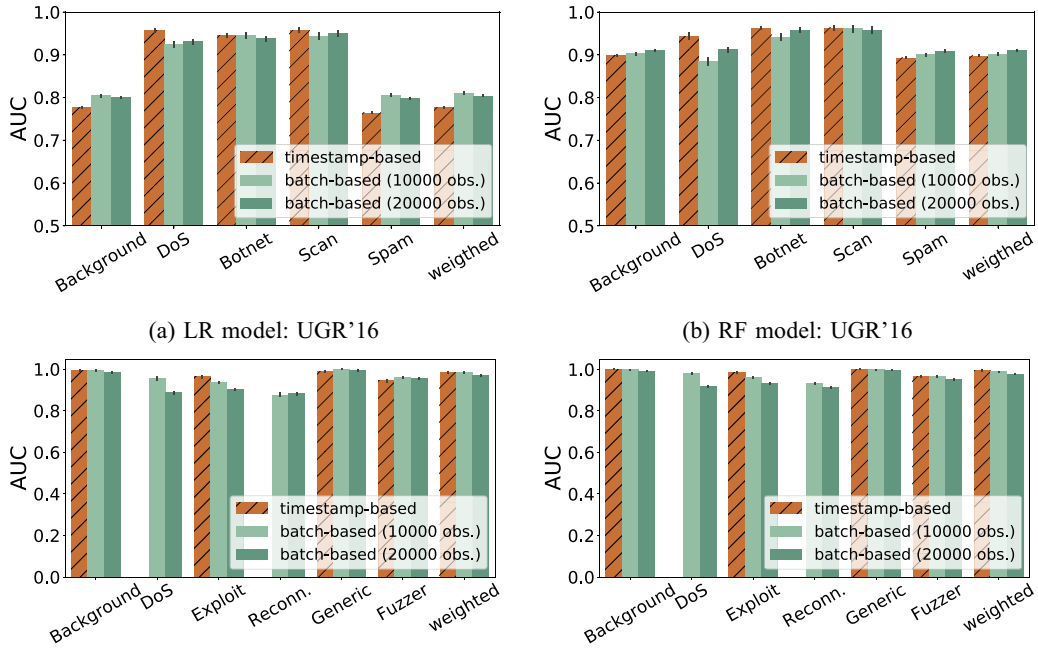
These results are both numerically and graphically represented in Table 6 and Figure 1 for each dataset and ML-based models, respectively. In the table, the AUC values of those classes with too low representation in the datasets were replaced by '−' symbol, meaning that their performance could not be obtained following a 5-fold cross-validation approach as recommended in [22]. Additionally and for the sake of the easy reading of the table, best results are highlighted in **bold font** for each dataset and class.

From the results, we can see that the models generally offer high accuracy on the attacks detection for the different versions of the two considered datasets, being the worst results those obtained by LR for UGR'16 dataset on *Background* and *Spam* classes, offering an AUC of around 0.8. In general, we can observe that there is no relevant impact on the ML models performance for the different aggregation techniques. Therefore, this result opens the door towards the application of FaaC to timestamp-less datasets, and the comparison of the performance of the ML models on different datasets, regardless they are timestamp-based or not. For instance, we could evaluate the performance of our studied models in well-known timestamp-less network datasets like KDDCup and its improvement, NSL-KDD, just by generating their derived datasets with FaaC, without any further modifications [24]. Additionally, we can see how the non-linear method performs better than the linear one for every dataset and class in general. The difference on the performance of the models becomes higher in UGR'16 dataset, where LR performs AUC values under 0.8 for *Background* and *Spam* classes, and the weighted average is in the same range of values too.

Another interesting conclusion, that could conduct the choice of one or another aggregation technique, is the number of observations each approach produces. Theoretically, ML models should benefit from the use of a high number of observations in the training process for a more accurate performance. However, this behavior does not always hold in our study, depending on the class type and the dataset. Indeed, the accuracy of the studied models is generally higher for the UNSW-NB15 dataset than for UGR'16, despite the fact the former has $1,441$ observations in the timestamp-based dataset and the latter has around $40,000$. Regarding the differences produced by the class types, batch-based aggregation approaches outperform timestamp-based ones for real classes like *Background* and *Spam* in comparison with synthetically generated ones, the *DoS, Scan* and *Spam* in the UGR'16 dataset, where the number of observations is divided by 2 and 4 (for the two batch-based derived datasets) with respect to the timestamp based version. It slightly differs from the classes in the UNSW-NB15 that were synthetically generated. In this case, even having similar behavior in terms of performance when the timestamp-based approach is used, the number of observations got after applying the FaaC with a one minute aggregation interval was $1,441$, which is significantly

TABLE 6. Average AUC performance for the different ML models, datasets and classes considered.

Dataset	Model	UGR'16		UNSW-NB15	
		Class	AUC	Class	AUC
timestamp-based	LR	Background	0.776	Background	**0.994**
		DoS	**0.957**	DoS	-
		Botnet	0.945	Exploit	0.962
		Scan	0.958	Reconnaissance	-
		Spam	0.764	Generic	0.988
				Fuzzer	0.945
		Weighted avg.	*0.776*	*Weighted avg.*	*0.981*
	RF	Background	0.898	Background	**0.999**
		DoS	0.943	DoS	-
		Botnet	**0.962**	Exploit	**0.984**
		Scan	**0.963**	Reconnaissance	-
		Spam	0.893	Generic	**0.999**
				Fuzzer	**0.966**
		Weighted avg.	*0.898*	*Weighted avg.*	*0.994*
batch-based (10,000 obs.)	LR	Background	0.805	Background	0.994
		DoS	0.924	DoS	0.954
		Botnet	0.946	Exploit	0.937
		Scan	0.943	Reconnaissance	0.875
		Spam	0.806	Generic	**0.999**
				Fuzzer	0.962
		Weighted avg.	*0.805*	*Weighted avg.*	*0.982*
	RF	Background	0.903	Background	0.998
		DoS	0.844	DoS	**0.978**
		Botnet	0.941	Exploit	0.961
		Scan	0.961	Reconnaissance	**0.930**
		Spam	0.90	Generic	**0.999**
				Fuzzer	0.965
		Weighted avg.	*0.901*	*Weighted avg.*	*0.988*
batch-based (20,000 obs.)	LR	Background	0.801	Background	0.985
		DoS	0.931	DoS	0.885
		Botnet	0.938	Exploit	0.904
		Scan	0.951	Reconnaissance	0.882
		Spam	0.798	Generic	0.994
				Fuzzer	0.956
		Weighted avg.	*0.804*	*Weighted avg.*	*0.968*
	RF	Background	**0.911**	Background	0.992
		DoS	0.912	DoS	0.920
		Botnet	0.958	Exploit	0.933
		Scan	0.958	Reconnaissance	0.911
		Spam	**0.908**	Generic	0.996
				Fuzzer	0.953
		Weighted avg.	*0.910*	*Weighted avg.*	*0.977*



(a) LR model: UGR'16

(b) RF model: UGR'16

(c) LR model: UNSW-NB15

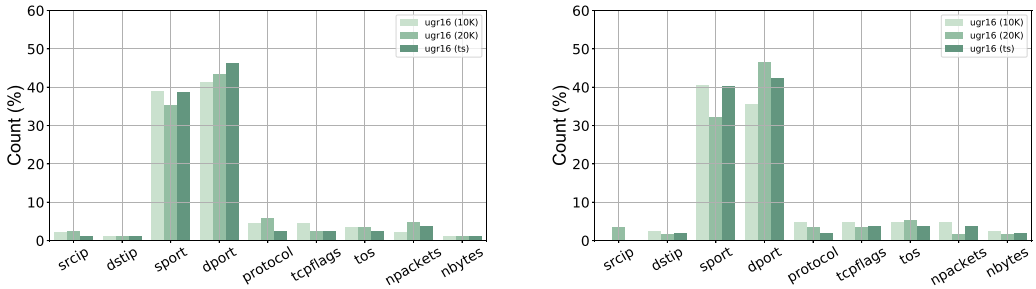
(d) RF model: UNSW-NB15

FIGURE 1. AUC performance results comparison for each class and dataset.

lower than in the case of the batch-based approaches. However, using timestamps for observations aggregation and counting outperforms batches mainly due the synthetic nature of the dataset.

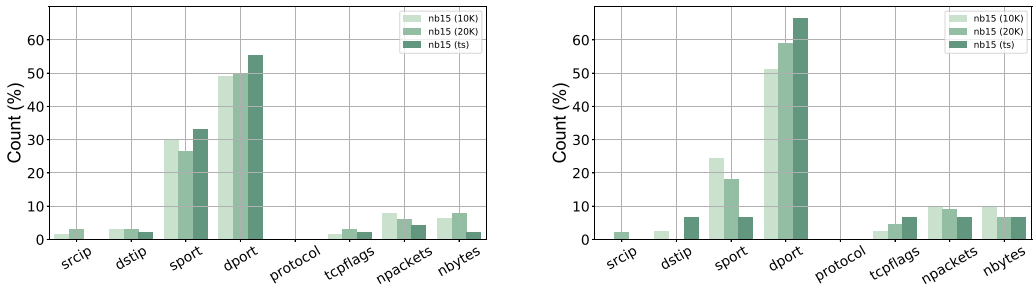
5.2 Feature selection analysis

Analysing the feature relevance in how ML-models are trained for classification is definitively important. As previously mentioned, LASSO was used as a feature selection procedure. To address such an analysis, we establish a threshold in such a way that those features that were selected less times than the threshold are discarded, computed over the 100 combinations from the cross-validation procedure (20 repetitions of 5-folds each). From that analysis, we can conclude that from a total of 134 derived features, the number of variables LASSO selects depends on the number of observations of the datasets. We observed that the higher the number of observations is, the larger the set of chosen variables. For instance, the number of features that were selected in all the 100 different cross-validation (threshold equal to 1) training for the UGR'16 {ts, 10K, 20K} and UNSW-NB15 {ts, 10K, 20K} derived datasets, were {52, 42, 56} and {15, 41, 44}, respectively. More in detail, Figures 2 and 3 show the selected features by group, threshold and derived datasets for UGR'16 and UNSW-NB15 datasets, respectively. These figures represent the percentage of selected variables from the total set by group. Overall, variables representing network services, e.g. source and destination ports, play a relevant role in training the ML-models. It is slightly more evident for the destination ports in the USNW-NB15 with higher threshold values maybe motivated by the synthetic nature of the dataset.



(a) Threshold: 0.6 (b) Threshold: 1

FIGURE 2. Relevant inter-group selected variables for the UGR'16 derived datasets.



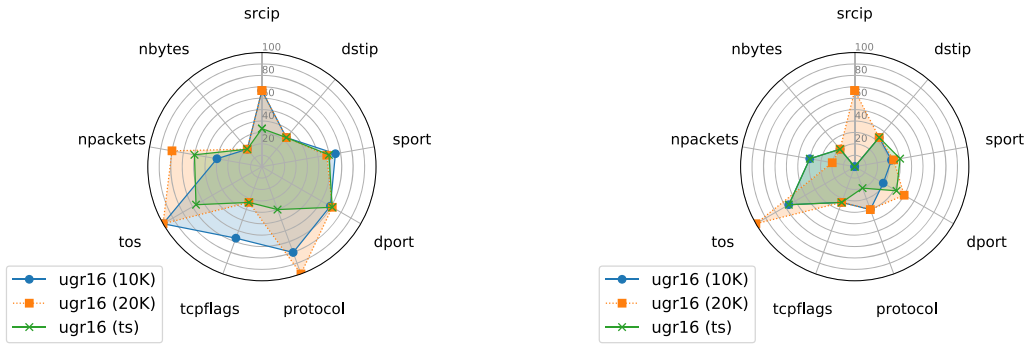
(a) Threshold: 0.6 (b) Threshold: 1

FIGURE 3. Relevant inter-group selected variables for the UNSW-NB15 derived datasets.

Different from the previous analysis, Figures 4 and 5 show the percentage of the selected variables by group from the total number of variables each specific group comprises. As previously introduced in Section 4, the FaaC transformation divides it in groups of derived variables with different group sizes. Here we can observe, that timestamp-based transformations reduce the number of selected variables in comparison with the batch-based approach for every threshold and every dataset. As expected, the higher the threshold is, the lower the number of selected variables. Considering only figures where the shown variables were always selected (i.e. setting up a threshold equal to 1), the most relevant ones, we can observe that IPs are also important in training the ML-models apart from the communication ports. IPs and communications ports are useful in the detection of distributed attacks like *botnets, scanning* or *DDoS*. Moreover, the size of the network communication flows is also relevant, useful for detecting attack patterns too.

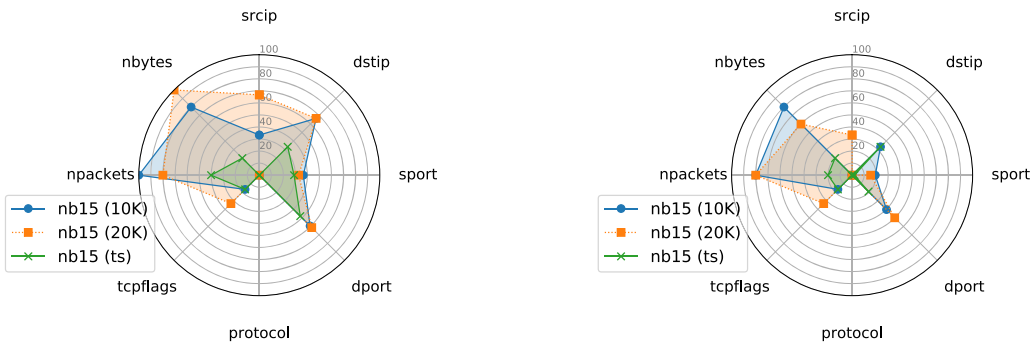
6 Conclusions and future work

We present in this work a new data aggregation methodology for Network Intrusion Detection Systems (NIDSs), based on Feature as a Counter (FaaC) feature engineering technique. The method is suitable for being used in real environments because the FaaC approach makes it simple the normalization of the system input data, required for Machine Learning (ML) models to avoid any biases produced by the magnitudes of the differences in the data. Additionally, the proposed method



(a) Threshold: 0.6 (b) Threshold: 1

FIGURE 4. Relevant intra-group selected variables for the UGR'16 derived datasets.



(a) Threshold: 0.6 (b) Threshold: 1

FIGURE 5. Relevant intra-group selected variables for the UNSW-NB15 derived datasets.

is interesting for the fair comparison of NIDS models for two main reasons. First, it allows generating derived datasets from the original one with a pre-specified number of observations, whatever the size of the original dataset. Second, it can be equally applied to timestamp-based or timestamp-less datasets, so it allows the comparison of NIDS solutions on different datasets, regardless they are timestamp-based or not, without any further adaptations of the models.

Two well-known datasets were used for the experiments in this work (namely, UGR'16 and UNSW-NB15 datasets), and two derived datasets were generated from each of them using the proposed technology: the batch-based derived datasets with 10,000 and 20,000 observations. Additionally, two widely spread ML models were evaluated, a linear and a non-linear ones (Linear Regression and Random Forest, respectively). From the results, it can be concluded that the two models generally perform similarly for all derived datasets considered, i.e. the two batch-based ones and the derived dataset obtained using timestamp information (aggregating samples from every minute intervals in the original raw dataset). Moreover, the batch-based datasets allow the comparison of the performance of the models on datasets with or without timestamp, a significant advantage. Additionally, the derived batch-based datasets are composed by exactly the same number

of observations, despite the size of the original ones. The selection of one or another aggregation approach depends on the nature of the dataset and its inner classes to effectively tune the FaaC method to maximize the obtained performance. Moreover, the number of the output observations in the case of the batch-based approach and the time interval for the timestamp-based one should be adequately chosen, optimized and dynamically adapted for the deployment of NIDS on real production networks.

Another study was made to analyse the relevance of the different features on the learning process of the considered models. For that, all features were grouped into several categories, and it was analysed the percentage of features in each category chosen as relevant for the models (using LASSO). We considered two different cases, those where the percentage of chosen features was either 100% or higher than 60%. As the main conclusion of this study, those features related to network services (as source and destination ports) play a relevant role in the decisions made by the ML models. It was also observed that the models generally require a higher number of representative variables for the batch-based datasets than for the timestamp-based ones.

As future work, we find it will be interesting to expand this study with the inclusion of timestamp-less datasets, in order to extend the comparison of the performance of the models to other well-known datasets that do not include timestamp information. This comparison has not been done yet in the literature, but the data aggregation methodology proposed in this work allows it.

Acknowledgements

This work was supported by the Spanish Ministerio de Ciencia, Innovación y Universidades and the ERDF (RTI2018-100754-B-I00, RTI2018-098160-B-I00 and PID2020-114495RB-I00), ERDF under project FEDER-UCA18-108393 (OPTIMALE) Junta de Andalucía and ERDF (GENIUS–P18-2399). B. Dorronsoro acknowledges 'ayuda de recualificación' funding by Ministerio de Universidades and the European Union-NextGenerationEU. Additionally, this publication results from the project NetSEA-GPT (C-ING-300-UGR23) funded by Consejería de Universidad, Investigación e Innovación and the European Union through the ERDF Andalusia Program 2021-2027.

References

[1] Cisco Annual Internet Report (2018–2023). White Paper. https://bit.ly/3jpAgNx, 2020. [Online; Accessed January 27, 2023].

[2] A. M. Aleesa, M. Younis, A. A. Mohammed and N. Sahar. Deep-intrusion detection system with enhanced unsw-nb15 dataset based on deep learning techniques. *Journal of Engineering Science and Technology*, **16**, 711–727, 2021.

[3] R. Ali, A. Ali, F. Iqbal, A. M. Khattak and S. Aleem. A systematic review of artificial intelligence and machine learning techniques for cyber security. In *Big Data and Security, Communications in Computer and Information Science*, Y. Tian, T. Ma and M. K. Khan., eds, pp. 584–593. Springer, Singapore, 2020.

[4] M. Belouch, S. El Hadaj and M. Idlianmiad. Performance evaluation of intrusion detection based on machine learning using Apache Spark. *Procedia Computer Science*, **127**, 1–6, 2018.

[5] J. C. Bezdek, R. Ehrlich and W. Full. FCM: the fuzzy c-means clustering algorithm. *Computers & Geosciences*, **10**, 191–203, 1984.

[6] M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita. Network anomaly detection: methods, systems and tools. *IEEE Communications Surveys Tutorials*, **16**, 303–336, 2014.

[7] L. Breiman. Random Forests. *Machine Learning*, **45**, 5–32, 2001.

[8] J. Camacho, G. Maciá-Fernández, J. Díaz-Verdejo and P. García-Teodoro. Tackling the Big Data 4 vs for anomaly detection. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 500–505, 2014.

[9] J. Camacho, G. Maciá-Fernández, N. M. Fuentes-García and E. Saccenti. Semi-supervised multivariate statistical network monitoring for learning security threats. *IEEE Transactions on Information Forensics and Security*, **14**, 2179–2189, 2019.

[10] J. Camacho, J. M. García-Giménez, N. M. Fuentes-García and G. Maciá-Fernández. Multivariate Big Data Analysis for intrusion detection: 5 steps from the haystack to the needle. *Computers & Security*, **87**, 1–11, 2019.

[11] J. Camacho, A. Pérez-Villegas, P. García-Teodoro and G. Maciá-Fernández. Pca-based multivariate statistical network monitoring for anomaly detection. *Computers & Security*, **59**, 118–137, 2016.

[12] J. Camacho, A. Pérez-Villegas, P. García-Teodoro and G. Maciá-Fernández. PCA-based multivariate statistical network monitoring for anomaly detection. *Computers & Security*, **59**, 118–137, 2016.

[13] J. Camacho, J. M. García-Giménez, N. M. Fuentes-García and G. Maciá-Fernández. Multivariate Big Data Analysis for intrusion detection: 5 steps from the haystack to the needle. *Computers & Security*, **87**, 101603, 2019.

[14] ENISA. *ENISA Threat Landscape Report 2020*. [Online; Accessed 29-apr-2022] https://bit.ly/3gdsB1O.

[15] J. Friedman, T. Hastie and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, **33**, 1–22, 2010.

[16] V. Hajisalem and S. Babaie. A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Computer Networks*, **136**, 37–50, 2018.

[17] E. Kabir, H. Jiankun, H. Wang and G. Zhuo. A novel statistical technique for intrusion detection systems. **79**, 303–318, 2018.

[18] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey and R. T. Goswami. An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset. **23**, 1397–1418, 2020.

[19] A. H. Lashkari, G. D. Gil, M. S. I. Mamun and A. A. Ghorbani. Characterization of tor traffic using time based features. *ICISSP 2017–Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, **2017**, 253–262, 2017.

[20] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro and R. Therón. UGR'16: a new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers & Security*, **73**, 411–424, 2018.

[21] R. Magán-Carrión, J. Camacho, G. Maciá-Fernández and Á. Ruíz-Zafra. Multivariate statistical network monitoring–sensor: an effective tool for real-time monitoring and anomaly detection in complex networks and systems. *International Journal of Distributed Sensor Networks*, **16**, 155014772092130, 2020.

[22] R. Magán-Carrión, D. Urda, I. Diaz-Cano and B. Dorronsoro. Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches. *Applied Sciences*, **10**, 2020.

[23] R. Magán-Carrión, D. Urda, I. Diaz-Cano and B. Dorronsoro. Assessing the impact of batch-based data aggregation techniques for feature engineering on machine learning-based network IDSs. In *In 14th International Conf. on Comp. Intelligence in Security for Information Systems*, pp. 116–125. Springer, 2022.

[24] R. Magán-Carrión, D. Urda, I. Diaz-Cano and B. Dorronsoro. Improving the reliability of network intrusion detection systems through dataset integration. *IEEE Transactions on Emerging Topics in Computing*, **10**, 1717–1732, 2022.
[25] N. Moustafa and J. Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference (MilCIS)*, 1–6, 2015.
[26] T. S. Pooja and S. Purohit. Evaluating neural networks using bi-directional LSTM for network IDS (intrusion detection systems) in cyber security. *Global Transitions Proceedings*, 2021.
[27] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani. *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*. 2018.
[28] M. Siddiqi and W. Pak. Efficient filter based feature selection flow for intrusion detection system. In *International Workshop on Emerging ICT*, **9**, 2020.
[29] J. Snoek, H. Larochelle and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, **25**, 2012.
[30] B. A. Tama, M. Comuzzi and K. H. Rhee. TSE-IDS: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE Access*, **7**, 94497–94507, 2019.
[31] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.
[32] Q. Tian, D. Han, K.-C. Li, X. Liu, L. Duan and A. Castiglione. An intrusion detection approach based on improved deep belief network. **50**, 3162–3178, 2020.
[33] J. Toldinas, A. Venčkauskas, R. Damaševičius, Š. Grigaliūnas, N. Morkevičius and E. Baranauskas. *A Novel Approach for Network Intrusion Detection Using Multistage Deep Learning Image Recognition*. **10**, 2021.
[34] D. Urda, J. Montes-Torres, F. Moreno, L. Franco and J. M. Jerez. Deep learning to analyze rna-seq gene expression data. In *Advances in Computational Intelligence*, I. Rojas, G. Joya and A. Catala., eds, pp. 50–59. Springer International Publishing, 2017.
[35] D. Urda, F. Aragón, R. Bautista, L. Franco, F. J. Veredas, M. G. Claros and J. M. Jerez. BLASSO: integration of biological knowledge into a regularized linear model. *BMC Systems Biology*, **12**, 94, 2018.
[36] T. Zoppi and A. Ceccarelli. Prepare for trouble and make it double. Supervised and unsupervised stacking for anomaly based intrusion detection. *Journal of Network and Computer Applications*, **189**, 103106, 2022.

Received 20 May 2022