Received 28 October 2021; revised 18 March 2022; accepted 20 May 2022.
Date of publication 2 June 2022; date of current version 6 December 2022.

Digital Object Identifier 10.1109/TETC.2022.3178283

Improving the Reliability of Network Intrusion Detection Systems Through Dataset Integration

ROBERTO MAGÁN-CARRIÓN, DANIEL URDA, IGNACIO DIAZ-CANO, AND BERNABÉ DORRONSORO

Roberto Magán-Carrión is with the Network Engineering and Security Group (NESG), Department of Signal Theory, Telematics and Communications, School of Computer Engineering and Telecommunications, University of Granada, 18071 Granada, Spain
Daniel Urda is with the Grupo de Inteligencia Computacional Aplicada (GICAP), Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad de Burgos, 09001 Burgos, Spain
Ignacio Diaz-Cano and Bernabé Dorronsoro are with the Superior Engineering School, University of Cadiz, 11519 Cádiz, Spain

CORRESPONDING AUTHOR: ROBERTO MAGÁN-CARRIÓN (rmagan@ugr.es)

This work was supported in part by Spanish Ministerio de Ciencia, Innovación y Universidades and the ERDF under Grants RTI2018-100754-B-I00 (iSUN), RTI2018-098160-B-I00 (DEEPAPFORE) and PID2020-114495RB-I00 (SICRAC), in part by ERDF under Grant FEDER-UCA18-108393 (OPTIMALE), and in part by Junta de Andalucía and ERDF under Grant GENIUS – P18-2399.

ABSTRACT This work presents Reliable-NIDS (R-NIDS), a novel methodology for Machine Learning (ML) based Network Intrusion Detection Systems (NIDSs) that allows ML models to work on integrated datasets, empowering the learning process with diverse information from different datasets. We also propose a new dataset, called UNK22. It is built from three of the most well-known network datasets (UGR'16, USNW-NB15 and NLS-KDD), each one gathered from its own network environment, with different features and classes, by using a data aggregation approach present in R-NIDS . Therefore, R-NIDS targets the design of more robust models that generalize better than traditional approaches. Following R-NIDS, in this work we propose to build two well-known ML models for reliable predictions thanks to the meaningful information integrated in UNK22. The results show how these models benefit from the proposed approach, being able to generalize better when using UNK22 in the training process, in comparison to individually using the datasets composing it. Furthermore, these results are carefully analyzed with statistical tools that provide high confidence on our conclusions. Finally, the proposed solution is feasible to be deployed in network production environments, not usually taken into account in the literature.

INDEX TERMS Robust network intrusion detection systems, NIDS, network security, machine learning, data aggregation, data integration

I. INTRODUCTION

Several technical reports announce the notable growing of the devices connected to Internet Protocol (IP) networks nowadays and in a near future. For instance, the Cisco Annual Internet Report (2018–2023) [1] forecasts that its number will exceed 29 billions, >3 times the global population. This fact can be motivated by the advent of the real deployment of communications technologies like 5G, providing easy and affordable Internet access for a huge number of heterogeneous devices from IoT (Internet of Things) ecosystems. Such an (inter-) connectivity allows to develop new applications and to offer new services inconceivable before. However, both the security risk and the attack surface grow as the number of connected devices, communications and services increase. Consequently, malicious actors have many ways and exposed entry points to perform and execute attacks. According to the ENISA Threat Landscape 2020 report (ETL) [2], the sophistication of threat capabilities seriously increased in 2019, having detected over 200,000 daily new variants of malware targeting diverse objectives.

In this scenario, there is an major need for novel methods and techniques for an early attack detection, which is of upmost importance to reduce the impact of the attacks on the target system or communication network. These new solutions must be robust, resilient, and adaptable to highly dynamic scenarios. According to the ETL report [2], the detection of an attack takes around 6 months in average thus, a short detection time reduces the attack's impact.

IDSs (Intrusion Detection Systems) have been widely used to address this problem in general, being the NIDSs (Network

IDSs) used for monitoring and detecting malicious actions in communications networks. A large number of Machine Learning (ML)-based solutions have been proposed in the literature for NIDSs [3]–[7]. These methods are generally trained and validated on one single dataset. However, despite the fact that some of the existing datasets were carefully built with realistic network traffic and attacks, such kind of studies does not allow providing generic conclusions about the performance of the proposed methods on other network datasets. Indeed, the trustworthiness of a method that is trained and validated on a single dataset is seriously compromised, and this is a major issue nowadays in the literature of ML-based NIDS methods. Besides, this is a critical concern in the topic, because these methods must detect network attacks (ideally, including new attacks) in any scenario.

More reliable solutions could be designed if several network datasets are jointly considered during the training process of the ML model. In principle, this may allow capturing the fundamentals of heterogeneous attacks observed and/or artificially generated in different network environments where each single network dataset was created. Unfortunately, integrating data from several datasets is a difficult task, given that all existing ones in the literature present important differences. For instance they differ in the number of features, their values, the annotated attacks, whether they include timestamp or not in the observations, etc. This makes it unfeasible to apply a given solution on different datasets with no additional methods or techniques.

Besides, ML-based techniques typically require the normalization of input data to avoid any undesirable bias due to existing differences in the magnitudes of the variables' values. In this sense, samples fed to ML methods must be normalized in order to achieve an acceptable generalization performance under the assumption of both the development and test sets having the same distribution. However, this assumption does not always hold since the data used to train a ML model may differ from the one present in the testing scenario, a well-known issue in ML-based approaches commonly known as data shift [8], [9] (e.g., differences on the class labels definition —shift on the output—, differences on the distribution of input features —shift on the input—). In fact, this issue is more likely to be present in network datasets for NIDSs since each of them are built on different network environments and under different experimental settings (e.g., real or synthetic network traffic, variety of attacks samples both real or manually generated, etc.). Consequently, one needs to solve the challenge of normalizing data belonging to different network datasets in order to put ML models that generalize well into production, in terms of accurately classifying the traffic of a real network.

Camacho *et al.* proposed in [10] a data aggregation technique for network datasets, called *Feature as a Counter* (FaaC). It aggregates a batch of raw network traffic observations into one single sample by translating the dataset features into a new set of features that represent counters of the previous ones. FaaC was later successfully applied to NIDS in [5], [11]–[13], and the authors applied it to the UGR'16 dataset [14], a timestamp-based network dataset built from real network traffic. It is reasonable to think that time distribution (i.e., network flows within a certain time slot) might contain relevant information to detect attacks and, consequently, aggregating raw information by time intervals is relevant in this field. However, its applicability to other well-known previous network datasets, such as USNW-NB15 [15] or NLS-KDD [16], the latter with no timestamp information, is not a priory possible, or at least as the FaaC technique was conceived.

In this sense, a novel methodology to apply FaaC to timestamp-less datasets was recently proposed in [17]. The paper shows that there are no major differences between the performance of ML models trained on UGR'16 dataset and two timestamp-less versions. Based on this result, we propose in this work a novel methodology that allows using and combining several network datasets (with and without timestamp information), each one with its own set of input features, to train reliable ML-based NIDSs solutions that generalize better across different network environments, in contrast to the single dataset-based solution that is typically found in the literature. This combination of heterogenous datasets is possible thanks to the use of FaaC.

Additionally, this novel methodology allows normalizing the derived datasets in an easy and flexible way (no matter the number of samples, the values in their variables nor the network environment used to build the raw dataset), a key feature that makes feasible the use of the previously built ML models in a production scenario [18], in comparison to other existing ML-based NIDSs solutions. Our main hypothesis is that the resulting ML model developed using this novel methodology may benefit from the overall attacks information contained across different datasets, which gives hope to achieve ML-based NIDSs solutions that outperform classical approaches that are only trained in one single dataset. The proposed methodology also considers all the steps required for a correct practice when using ML [5], including data normalization, feature engineering and selection, hyper-parameters selection (we use Least Absolute Shrinkage and Selection Operator, LASSO for short) within a cross-validation approach for model classification assessment. Although this methodology allows adopting many different ML models, it is not the scope of this work to perform such a large comparison (an interesting future work). Instead, we chose two of the most representative models of the main ML categories, namely linear and non-linear methods, to compare their performance. They are Linear Regression [19] and Random Forest [20], respectively. Altogether, the proposed methodology supposes the first step towards the design of a new generation of reliable ML-based NIDSs that can perform accurately in generic network scenarios.

The main contributions of this work are:

- The deployment of a complete methodology, called Reliable-NIDS (R-NIDS), that allows building trustworthy ML models for NIDSs, as well as evaluating

TABLE 1. Methodology comparison for NIDS evaluation.

Work	Dataset	Methodology					
		FEFS	DP	HS	ML	PM*	SSA
Hajisalem *et al.* [6]	NSL-KDD, UNSW-NB15	– ✓	–	–	other	A, TFR	–
Kabir *et al.* [7]	KDDCup'99	– ✓	–	–	classic	F1, R, P	–
Sharafaldin *et al.* [21]	CICIDS2017	✓ ✓	–	–	classic	F1, R, P	–
Siddiqi *et al.* [22]	NLS-KDD, ISCX12	✓ ✓	mean	–	classic	A, F1	–
Kumar *et al.* [23]	UNSW-NB15	✓ ✓	normalization	–	other	A, R, P	–
Tian *et al.* [24]	UNSW-NB15, NSL-KDD	✓ ✓	normalization	–	deep learning	R, O	–
Verma *et al.* [25]	CIDDS-001, UNSW-NB15, NSL-KDD	✓ –	–	✓	classic, deep learning	A, TFP, AUC	Friedman, Nemenyi
Riyaz *et al.* [26]	KDDCup'99	– ✓	numerical values	–	deep learning	A	–
Aleesa *et al.* [27]	UNSW-NB15	✓ ✓	normalization	✓	deep learning	A, P, ROC, AUC	–
Toldinas *et al.* [28]	UNSW-NB15	✓ ✓	normalization	–	deep learning	A	–
Pooja *et al.* [29]	KDD'99, UNSW-NB15	– –	normalization	–	classic, deep learning	A	–
Zoppi *et al.* [4]	NLS-KDD, UGR16, CICIDS17, UNSW-NB15	✓ ✓	normalization	✓	deep learning	A, TFR, F1, R, P	–
Sarhan *et al.* [30]	USNW-NB15, BoT-IoT, ToN-IoT, CSE-CIC-IDS2018, NF-UQ-NIDS	✓ –	normalization	–	classic	AUC, F1, TFR	–
Injadat *et al.* [31]	CICIDS17, UNSW-NB15	✓ ✓	normalization	✓	classic	A, FP, R, P	–
Magán-Carrión *et al.* [5]	UGR'16	✓ ✓	normalization	✓	classic	F1, R, P, AUC	Friedman, Wilcoxon

** A (Accuracy), TFR (TP –True Positive Rate–, FP –False Positive Rate–, TN –True Negative Rate–, or FN –False Negative Rate–), F1 (F1-score), R (Recall), P (Precision), ROC (Receiver Operating Characteristic curve), AUC (Area Under the Curve), O (Others).*

the results in a reliable way, including steps for (i) preprocessing and transforming raw network datasets, (ii) performing data integration of different network datasets, (iii) training robust ML models —feature selection, hyper-parameter selection, evaluation strategy, performance metrics—, and (iv) testing statistical significance. Besides, R-NIDS methodology allows to design, implement and make fair and honest comparisons of ML-based solutions for NIDSs which is not commonly seen in the literature.

- The introduction of a novel data integration approach, that allows to jointly consider different network datasets through a row-wise aggregation step. A new dataset, UNK22, is proposed combining network traffic from 3 previous and well-known network datasets: UGR'16, UNSW-NB15 and NSL-KDD.
- The implementation of a reliable ML-based solution using the new UNK22 dataset for NIDS, which is shown to considerably generalize better than previous existing ones that were based on one single network dataset. Particularly, this solution can be used in production in generic network scenarios thanks to the data aggregation and normalization approach presented in this paper. Furthermore, it can analyze network communication flows regardless of the aggregation strategy employed for the FaaC technique, i.e., analyzing traffic within a certain frequency (timestamp-based) or within a set of a certain size of consecutive network flows (batch-based).
- The study of how ML models generalize using validation datasets different from those used for training. This is, to the best of our knowledge, the first attempt to analyze the reliability of ML-based NIDSs.

The rest of the paper is organized as follows. In Section II, the most relevant works carried out to date regarding the evaluation of NIDS are studied. In Section III, the datasets that have been used in this study are introduced (UGR'16, USNW-NB15 and NSL-KDD). The proposed framework, called R-NIDS, is introduced and described in detail in Section IV. Next, the experimental design employed in this work to develop and evaluate reliable ML-based solutions for NIDSs is presented in Section V. Finally, Section VI presents and discusses the results obtained in the analysis, and the conclusions and future work are shown in Section VII.

II. BACKGROUND

Throughout this section, we present and discuss some of the most interesting articles dealing with the detection of network attacks using ML-based techniques. Table 1 provides an overview and comparison of these works. It mainly shows information about key factors involved when implementing ML-based NIDSs solutions which are, among other, the datasets they work with and the stages that should be found in a work pipeline to train and test ML modes as recommended in [5]. They are the Feature Engineering (FE), the Data Preprocessing (DP), the Feature Selection (FS), the Hyperparameters Selection (HS), the Machine Learning (ML) model and the Performance Metrics (PM) used. Additionally,

we consider a new and relevant stage for the Significance Statistical Analysis (SSA). The '✓' indicates that the corresponding work is considering a specific stage (existing or proposed by the authors) within their approach while the '–' shows right the opposite (i.e., the stage is not included within the work or the authors miss or omit to mention it). Rest of values in the table are self-explanatory.

A considerable amount of works have been recently published related to ML-based NIDSs. For instance, in [6] the authors proposed a NIDS solution for anomaly detection that relies on the joint use of swarm-intelligence-based optimization heuristics: Artificial Fish Swarm (AFS) and Bee Colony Optimization (BCO). This hybrid algorithm, is able to detect anomalies using a reduced subset of characteristics. The proposed system implements an adequate feature selection procedure, though it does not introduce any of the other what we consider recommended steps. Finally, this study does not provide details about the method used, thus making it difficult to compare it against other solutions.

The authors in [7] proposed a Least Squares Support Vector Machine (LS-SVM)-based NIDS. In this work, they used the Optimal Allocation technique to efficiently handle large datasets through the selection of the most representative samples, organized according to the number of samples and the confidence interval to be achieved. Despite the LS-SVM model is not often used for intrusion detection, the authors considered this technique interesting to define the entire population from a series of representative examples.

In [21], the authors introduce and highlight the existing problems caused by the increasing number and variety of network attacks. They are in turn worsened due to the lack of appropriate datasets for training ML-based NIDSs. Therefore, they review a significant amount of network datasets from recent years, listing the problems that each one contains like: freshness, insufficient number of attacks, etc. As a result, they proposed a new dataset that solves, or at least mitigates these problems. They tested several classical ML models to corroborate it. Similarly, authors in [22] addressed this issue by remarking the importance of making an effective feature selection for an efficient training and testing of ML models. For this purpose, they proposed to implement and evaluate the effects of normalizing samples before performing feature selection, concluding that this strategy provides better results than the existing ones.

In [23] the authors proposed a system to detect five conventional attacks. The author built a new dataset that is afterwards compared with the USNW-NB15 dataset. In this sense, they employed the gain information technique over the set of features on the original USNW-NB15 dataset and used a misuse-based approach to create this new dataset. Such dataset was used to train ML models making them to perform better in terms of precision, attack detection rate, false positive rate, among other performance metrics.

The authors of [24] addressed a well-known problem in ML classification tasks: the overfitting problem. To overcome this problem, they proposed an approach based on improved Deep Belief Network (DBN) composed of multiple hidden layers and connections. Together with DBN, the dataset used was pre-processed using the Probabilistic Mass Function (PMF) and the Min-Max normalization technique that simplifies this procedure. Both techniques allowed to achieve a small improvement in the performance compared to other studies compared in this work. The authors considered all the stages we recommend here for a correct ML models training and testing except the hyper-parameter selection and the significance statistical analysis stages.

NIDS solutions are also being applied in IoT ecosystems, such as the one reported by the authors in [25]. This work proposes the use of classification algorithms to detect DoS attacks, one of the most damaging attacks for IoT platforms. The performance tests were made on ML popular datasets and using a Multi-Layer Perceptron (MLP). Although authors perform a hyper-parameter selection, other important phases, such as feature selection, were ignored. Additionally, they employed significance tests, specifically Friedman and Nemenyi, to statistically analyze the performance obtained, which should be a mandatory step in order to have confidence in the results obtained [32].

A Convolutional Neural Network (CNN) is used in [26] for intrusion detection in wireless networks. They use KDD'99 Cup dataset, whose non-numerical features have been transformed into numerical values. The work proposes an interesting feature selection mechanism, called conditional random field and linear correlation coefficient-based feature selection. The model is evaluated and compared against others from the literature according to their accuracy.

A Deep Learning (DL)-based NIDS is featured in [27]. The authors proposed using a popular dataset, the USNW-NB15 to perform a binary and multi-class classification. In the stages of their methodology, they included all the steps suggested in our proposal except the significance statistical analysis. Subsequently, they compared the performance of the trained models with other published works, obtaining a performance rate close to 100%. Similarly, the authors in [28] proposed a DL-based NIDS using the same dataset. This work tried to enhance state of the art NIDSs by using of image recognition techniques through different stages. However, they did not mention anything about hyper-parameters selection. With respect to the dataset, they transformed it into four-channel images (red, green, blue, and alpha), which were subsequently used to train and validate the proposed model. Another DL-based system using bidirectional LSTM (Long Short-Term Memory) is introduced by the authors in [29]. The experiments are carried out on classic datasets, and data normalization is only applied within the data preprocessing stage. The remaining stages were not contemplated by the authors, where only a performance study based on the considered activation function within the DL model was included.

In [4] the authors noted that supervised learning has been very effective for known threats, although it does not work well at detecting unknown attacks that are commonly called

zero-day attacks. In the study, the authors proposed a semi-supervised learning system, combining two layers, to detect known and unknown threats, and they address all the recommended steps for an adequate methodology in the study of network attacks, except the use of SSA.

A methodology for ML-based NIDS is proposed in [31]. It compares several methods for feature and hyper-parameter selection. The results of two classic ML models (KNN and RF) are analyzed for CICIDS17 and UNSW-NB15 datasets using different performance metrics, and no statistical significance analysis was performed. The work emphasizes an important limitation found in the literature, where all papers use the same dataset for training and testing, while the generalization capability of the models on other datasets (different dataset from the one used for training) should be analyzed. However, they do not perform this study either. Our work is pioneer in this sense, and we prove that ML models trained in one dataset are not able to generalize to different situations, a desirable feature for a ready-production NIDS system.

Another interesting study about the need of building network datasets comprising a standard feature set for NIDS evaluation is introduced in [30]. The authors transform 4 state-of-the-art network datasets into NetFlow-based versions where a common feature space is shared for fair NIDS comparison. Moreover, it allows to merge datasets for improving, potentially, how NIDSs generalize, which is a desired characteristic to deploy them in real network environments different from the ones used for training. In fact, the authors built a new dataset named NF-UQ-NIDS which merges all the previous ones. For assessing the suitability of the datasets for NIDS classification, they use a classic ML-model (ExtraTree) following a traditional one-single dataset approach i.e., training and testing with the same dataset. However, first, they forgot or elude to mention what we consider relevant steps for a fair NIDS evaluation and, in the end, for a fair comparison. They are the FS, HS and SSA steps. Second, though it is a really nice approach for training robust NIDSs ready to be used in production network environments, the authors fail in validating it following a one-single dataset evaluation approach. Conversely, our work carries out a deep analysis of how classic ML-models, both linear and non-linear, generalize. We conclude that training ML-models with an aggregated dataset comprising single and heterogeneous ones outperform significantly their classification performance than in the case of following a classic one-single-dataset approach for training and testing them.

Therefore, considering Table 1 and the guidelines proposed in [5], we can conclude that most of the works still fail in the use of HS and SSA stages, two of the most important steps for a reliable and honest design and comparison of NIDS solutions. It is also worth noting the recurrent use of the KDD-related datasets for validating NIDSs still nowadays, even though it is undoubtedly outdated (created almost 20 years ago in a field that significantly evolves within a year time). Moreover, authors tend to use the accuracy measure to validate their models, although it is widely known as a non-fair performance metric for highly imbalanced classification problems, such as the context of the problem addressed here. Finally, no works can still be found in the literature comprising the issue of making reliable ML-based NIDSs, aiming at performing well in generic network environments. In this work, we test our main hypothesis based on the idea that integrating network datasets would empower the learning process with diverse information from different datasets, at the same time that ML-based systems would be more adaptable in order to put them in production in generic network environments.

III. NETWORK DATASETS

Network security researchers continuously deal with a well-known issue concerning the existence of an adequate number of datasets to evaluate NIDS. Thus, several efforts have been made in recent years in order to obtain good enough network datasets in some sense. For instance, some datasets stand out and differ from others in terms of the data format used (network flows or packets), the way in which the data is recorded (real or synthetic), the duration of the data recordings used in the dataset and their freshness [14], [33].

Therefore, building the perfect dataset is not a trivial task, as stated in [33], [34] and which is mainly supported by two compelling reasons: 1) new attacks samples are continuously observed [2], [35], [36], which makes it hard to have a permanently up-to-date dataset; and 2) the fact that specific applications context of ML-based models need specific datasets [4].

Even in the case that no perfect dataset exists, in this study we hypothesized that ML-based solutions trained using a combination of datasets, could perform better than classic solutions involving only one dataset for training and testing. Moreover, this approach is more likely to be suitable for its implementation in real and generic production environments than others built using only one dataset.

Next, we briefly introduce the three network datasets used in this work, to be afterwards combined into a single one called UNK22 (see Section V). They are the UGR'16, USNW-NB15 and NSL-KDD, and were gathered in different network scenarios and also including a variety of attack samples generated with different tools.

A. UGR-16

The UGR'16 [14] dataset contains anonymized network traffic flows (NetFlow) collected during 4 months at the facilities of an Internet service provider (ISP) in Spain. Thus, the authors of this dataset have divided its content into two different parts: CAL, which refers to data obtained from March to June 2016, containing inbound and outbound ISP network traffic; and TEST, corresponding to traffic obtained from July to August 2016 mostly with attack streams that were either synthetically generated or found by detectors.

To guarantee the correct functioning of the ISP and not adulterate the netflows with the synthetic attacks, they were

launched under a controlled environment. Most of these attacks were created using updated tools that cover difficult to detect harmful attacks, like *DoS (both low and high rate)*, *Scan (Port Scanning)* or *Botnet*. However, UGR'16 does not include only synthetic attacks but also real attacks, which are detected and labeled by the authors. These detected attacks are *UDP port scan*, *SSH scan* and *Spam* campaigns. Finally, and not less relevant, those streams generated by IP addresses in open blacklist sources were tagged as *Blacklist*.

The dataset is composed by 23 files, one per week. From them, 17 were selected for CAL subset and the other 6 for TEST. The size of each file is around 14 GB (compressed) and they can be downloaded in *nfcapd* or CSV formats.[1]

We summarize in Table S1 of the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TETC.2022.3178283 the class distribution of the TEST part, together with the number of flows in each class. As it could be expected, the number of attack traffic flows is notably unbalanced with respect to normal traffic (*Background*). Among the attacks, *Spam* is the one with the highest number of flows, representing only 1.96% of the flows in TEST. In addition, TEST contains flows of all mentioned attacks, both synthetically generated and discovered from anomalous behaviors. For the mentioned reasons, we decided to use this part of the UGR'16 dataset in this work.

B. USNW-NB15

UNSW-NB15 [15] is a public dataset that collects raw network traffic using the IXIA tool "PerfectStorm" in the Cyber-Range laboratory of the Australian Center for Cybersecurity (CASS). This set up consist of several network devices (firewalls and routers) connecting servers and hosts. In concrete, servers are in charge of generating both normal and malicious network traffic, from end to end devices, traversing a firewall.

After using the IXIA tool under this set-up, 47 characteristics were obtained from the data. The *TCPdump* tool was installed on one of the routers to capture the PCAP files where a total of 2,540,044 records (traffic flows) were recorded an stored in four CSV files.[2] Nine different classes of attacks were identified: *Fuzzers*, *Analysis*, *Backdoors*, *DoS*, *Exploits*, *Generic*, *Reconnaissance (Port Scanning)*, *Shellcode*, and *Worms*. For this study, the four CSV files of traffic flow records have been merged, and the number of samples that exist of each class have been counted, as shown in Table S2 of the supplementary material, available online. Similarly to the UGR'16 dataset, there is a large percentage of background traffic in comparison to the other classes, followed by the the Generic attack. Other attack samples have a lower presence in the datasets.

C. NLS-KDD

NSL-KDD is a dataset built from the KDDCup'99 dataset. It was conceived to solve some of the well-kown flaws it raises e.g., the duplicity on the records [16]. However, this new version of the KDD dataset still inherits some of the problems of the original one and it may not be considered a good representation of real traffic in communication networks. It is considered in this study to analyze its influence with respect to the other two involved datasets that are, in principle, more suitable according to their characteristics.

The NLS-KDD dataset comprises two CSV files[3] for training and testing purposes. In our case, we have merged the two files into only one. Table S3 shows the number of flows of each class category. *Probe (Port Scanning)* class comprises the following kind of attacks: *psweep, nmap, portsweep, satan, saint, mscan*. Similarly, for the *DoS* class, it comprises *back, land, neptune, pod, smurf, teardrop, apache2, mailbomb, udpstorm, processtable* attacks. *R2L (Remote to Local)* attacks which tries to get local access to remote target by exploiting certain vulnerabilities comprise the following type of attacks: *ftp_write, guess_passwd, multihop, phf, imap, spy, warezclient, warezmaster, named, xsnoop, xlock, sendmail, worm, snmpgetattack, snmpguess*. Finally, the *U2R (User to Root)* class comprises the following privilege escalation attacks: *perl, rootkit, loadmodule, buffer_overflow, httptunnel, ps, sqlattack, xterm*. Besides, Table S3 of the supplementary material, available online, shows that NSL-KDD notably differs from the other networks datasets in terms of the attacks and normal traffic distributions, representing a more balanced problem than these other datasets, as opposite to what one observes in a real network environment.

IV. RELIABLE NETWORK IDS (R-NIDS) METHODOLOGY

Due to the continuous growing of security threats, more sophisticated methods and ways need to be devised to counteract against new samples of malicious security attacks, specially in the detection stage. Reliable-NIDS (R-NIDS) proposes a new methodology to achieve trustworthy and reliable NIDS solutions. On one hand, it provides a flexible and adaptable framework for a fair evaluation and comparison of NIDS solutions. On the other hand, and thanks to the new data integration approach proposed, makes it suitable to built enhanced ML models that perform correctly in network production environments. Modules and stages of R-NIDS are shown in Figure 1.

On the right part of the figure, the main modules and stages to evaluate and compare NIDSs are shown. They are: the Feature Engineering (FE), the Feature Selection (FS), the Data Preprocessing (DP), the Hyper-parameter Selection (HP), the Machine Learning model (ML), the Performance Metric (PM) used and the Significance Statistical Analysis (SSA) performed. As it is emphasized in [5], almost all the

[1]The UGR'16 dataset: https://nesg.ugr.es/nesg-ugr16/
[2]UNSW-NB15: https://researchdata.edu.au/unsw-nb15-dataset
[3]The NSL-KDD dataset: https://www.unb.ca/cic/datasets/nsl.html

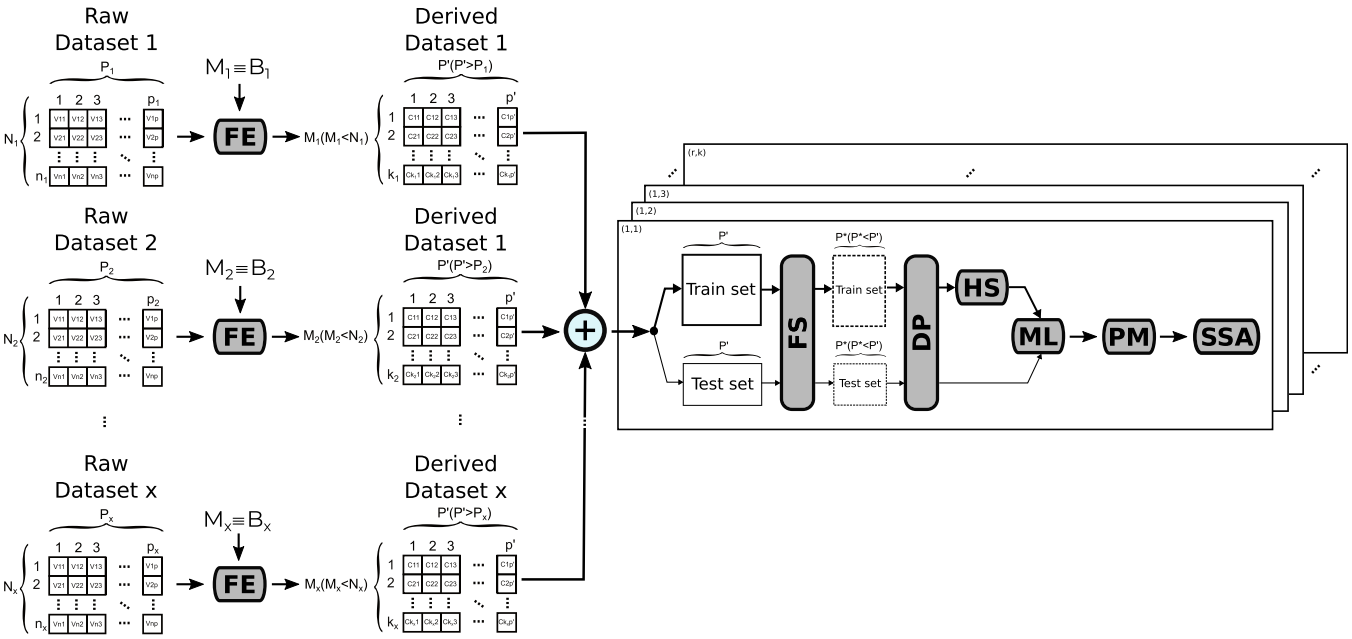


FIGURE 1. **R-NIDS framework and their main stages. FE: *Feature Engineering*, FS: *Feature Selection*, DP: *Data Preprocessing*, HS: *Hyperparameters Selecion*, ML: *Machine Learning model*, PM: *Performance Metric* and SSA: *Statistical Significance Analysis*.**

analyzed previous works either missed to include some of them or elude mentioning them, making it difficult to compare different NIDS solutions.

A. BATCH-BASED FE FOR MULTIPURPOSE DATA AGGREGATION AND INTEGRATION: A ROW-WISE APPROACH

As depicted in the left-hand side of Figure 1, R-NIDS is able to manage heterogeneous network datasets to perform a data aggregation and to integrate them into only one overall NIDSs dataset. This is achieved thanks to the Feature as a Counter (FaaC) method [10] applied in the FE module. FaaC converts the dataset into a derived dataset with completely new variables, which are counters (P' in the figure) on the values of the variables in the original dataset ($P_1, P_2, \ldots, P_x$ in the figure) for any given batch of raw network flows (e.g., a batch consisting of input flows withing each minute, or a batch consisting of a set of consecutive input flows). FaaC outputs data matrices of $M_x \times P'$ dimension where the number of observations depends on the M_x parameter but the number of derived variables, P', remains equals no matter the raw dataset used. M_x will consequently determine the batch size, B_x, of raw network flows that are aggregated using the FaaC technique in order to obtain a derived dataset of M_x observations. Thus, the higher the values of M_x, the smaller the size of the batch, and vice versa [17]. Furthermore, each feature P' in the derived dataset is normalized by the corresponding batch size used to process the raw dataset in order to guarantee that each new feature now lies in the [0,1] range.

For a given classification task in supervised learning, the main goal is to accurately classify as many classes as the dataset has, which is the context of application in the current work. Concerning the dependent variable or output class in the derived dataset, the aggregation performed by the FaaC technique on a given batch of raw observations needs to account for specific situations, since the batch is likely to comprise more than one raw observations of different classes and, consequently, not making it trivial how to choose the class that the aggregated observation belongs to. In this work, we performed a simple procedure that consists of choosing the most predominant attack class counted from the raw observations within a batch. If no attack is present at all, then the aggregated observation is labeled as normal or background traffic. This solution penalizes those raw network datasets with balanced classes which, in fact, is not realistic since network traffic communications are imbalanced by nature (i.e., there should be much more normal or background traffic than attacks). It is the case of the NLS-KDD dataset. Instead, this procedure inherently balances positive (attacks) and negative (background) observations as we will see in the next section.

All previous methods allow having a rectangular matrix per dataset where the variables' values are normalized counters in the [0,1] range, making them easy to be managed as input by any ML model. Besides, this procedure provides the opportunity of customizing the way of integrating all the derived datasets into one joint dataset. The latter key functionality allows us to somehow mix different and heterogeneous network datasets to achieve reliable ML-based solutions. In this work, a simple row-wise data integration approach was employed as a case study, although any other way of combining different datasets could be set up thanks to the data aggregation and integration capabilities of the R-NIDS framework. In this way, one can think in to apply a column-wise approach or to smartly re-arrange and combine both, features and observations [37].

B. FEATURE SELECTION

This is an important module to take into account when developing ML-based solutions in which training samples consist of a high number of input features, as it is already known in the AI/ML community that irrelevant features may introduce noise and cause a negative impact in the predictor performance. For this purpose, there exist many different methods in the literature which classify into one of the three well-known families of feature selection techniques: filter, wrapper and embedded methods. Since the main goal of this work is not the comparison and evaluation of different feature selection methods on the given task, authors propose the use of the Least Absolute Shrinkage and Selection Operator (LASSO) to deal with high dimensional settings based on their previous experience [38], [39]. Moreover, LASSO is an embedded feature selection method which exploits the main benefit of filtering methods (they have less computational costs) and the benefit of wrapper ones (they are supervised methods which take into account the existing relationship between inputs and output features to perform the selection).

Therefore, LASSO is used in this work as a FS method in order to reduce the dimensionality of the datasets employed to train the classifiers, where only input features that the method considers relevant for the target outcome are retained. For this purpose, as a linear model the LASSO [40] adds an l_1-penalty term to the minimization problem solved in a logistic regression model, as Equation (1) defines

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

where λ is a hyper-parameter of the model which controls the strength of the regularization, i.e., the bigger the value of λ, the stronger the regularization and, consequently, less input variables would be retained, and vice versa.

C. DATA PRE-PROCESSING

In this work, a standard normalization procedure is applied to each numeric feature, as shown in Equation (2), in such a way that normalized features have zero mean and unit variance

$$z_j = \frac{\boldsymbol{x_j} - \mu_j}{\sigma_j} \qquad \forall j \in [1, P] \ , \tag{2}$$

where $\boldsymbol{x_j}$ is a raw input variable, μ_j and σ_u are the mean and standard deviation of the given variable, and $\boldsymbol{z_j}$ is the standardized feature.

D. HYPER-PARAMETER SELECTION

Depending on the ML model used, the process of fitting a fairly optimized set of values for every model's hyper-parameters is not a trivial task. Traditionally, two well-known strategies have been used for this purpose: *Grid Search* [41], [42] and *Random Search* [43]. Both approaches have some drawbacks in relation to computation efficiency and the limited exploration of the hyper-parameters search space. Therefore, they are not really suitable for real problems where accurate and efficient solutions are needed.

To overcome the limitations of the above strategies, the Bayesian hyper-parameter optimization [44] is used in this work. In this strategy, hyper-parameters search is modeled by an underlying Gaussian process in such a way that, on each step of the iterative search, the next hyper-parameters setting to be tested is the one with more uncertainty (higher variance) defined by this Gaussian process. Therefore, this strategy ensures that a pseudo-optimal hyper-parameters setting will be achieved in a few number iterations.

Hyper-parameters optimization must be done before training the corresponding ML model, and the results depend on the dataset used in the training process. Therefore, in the context of R-NIDS methodology, this Bayesian hyper-parameter optimization algorithm is applied in each training fold of the cross-validation process, in order to get the most accurate model in each case. The hyper-parameters of the considered models are given in the following section.

E. MACHINE LEARNING MODELS

Given a set of n observations and p variables describing each observation, in supervised ML, and more specifically in any classification task, the goal of a model is to learn in the best possible way the existing relationship between the dependent variable or output class $y^{(i)}$, and the independent variables $x_1^{(i)}, \ldots, x_p^{(i)}$, for any ith sample. In the context of this work, $y^{(i)}$ corresponds to the label or class associated to the ith observation, thus resulting in a multi-classification problem where $y^{(i)} \in [0, \ldots, k]$ (i.e, background or normal traffic is mapped to 0, and each attack is mapped to consecutive numbers starting from 1). In this work, we chose two well-known classifiers representing a linear and a non-linear model. They are described next.

1) MULTINOMIAL LOGISTIC REGRESSION (LR)

It is the simplest linear model [19] possible, in which labels are obtained from a linear combination of the input features and the vector $\boldsymbol{\beta}$ of parameters, i.e., the dependent variable is modeled as $y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$. This model does not really have any critical hyper-parameter to tune, thus the training of the model is straightforward and consists of estimating the $\boldsymbol{\beta}$ coefficients through a common technique such as Ordinary Least Squares (OLS). In a binary classification task, this parameter vector is learned by solving the minimization problem depicted in Equation (3), which aims at estimating the optimal $\boldsymbol{\beta}$ coefficients that minimize the error of the model across the training samples

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

where $f(\boldsymbol{\beta}X^T)$ is the logistic or sigmoid function. In order to deal with the multi-classification setting, the well-known *One-vs-All* approach is employed in such a way that $k+1$ LR models (although slightly abusing with the notation, let

us consider Z LR models in such a way that $Z = k + 1$) are fitted to data, each one focusing on a specific instance of the data that corresponds to a binary classification problem (e.g., one LR model to distinguish background or normal traffic from everything else, another one to distinguish a port scanning attack from the rest, etc.). Since each of these Z LR models provides as output a probability $Pr_z \in [0, 1]$, one needs to employ the *softmax* function depicted in Equation (4) in order to normalize these probabilities and, therefore, ensure that the final output of the model is still a probability in the [0,1] range and $\sum_{z=1}^{Z} Pr_z = 1$.

$$Pr(y^{(i)} = k|X) = \frac{Pr(y^{(i)} = k|X)}{\sum_{z=0}^{k} Pr(y^{(i)} = z|X)} \quad , \tag{4}$$

2) RANDOM FOREST (RF)

It is a classification algorithm which consists of an ensemble of multiple decision trees that are fitted to different instances of the data (i.e, a random sample of the training observations and the set of predictors) [20]. In RF, each individual decision tree outputs a class prediction, thus the class or label predicted by RF for any new unseen observation is the class with the majority of the votes (i.e., the class most predicted by the individual decision trees).

One important characteristic of any ensemble method is the diversification of the models that compose the ensemble. RF ensures it in two ways: first, by using a bagging strategy in which each individual decision tree is trained over a slightly different set of training samples (e.g., by sampling with replacement or by randomly sampling a subset of instances from the training set) which can result in significantly different tree structures; and second, by considering feature randomness at the time of splitting a node in any of the individual decision trees, which forces even more variation amongst the trees in the model, resulting in lower correlation across trees and more diversification.

Although RF has several hyper-parameters to tune, in this work authors focus only on two of them which are more critical when training this model: the maximum number of predictors from which to choose and split each node of any individual decision tree (*mtry*), and the maximum size of each node in any individual decision tree (*nodesize*). The former is linked to the feature randomness strategy previously mentioned and, on each node, it randomly chooses a subset of input features from which to pick the one that maximizes the entropy, i.e., that produces the largest separation between the observations in the left node versus the ones in the right node. The latter is a hyper-parameter which controls how much each individual decision specializes on the training data, thus being used to prevent overfitting: the lower the value of this hyper-parameter, the more we force the model to continue splitting nodes within individual decision trees and the more likely they will overfit the data.

In this work, the R package *randomForest* was used to train a RF model and these two hyper-parameters (*mtry* and *nodesize*) are tuned following the strategy described in Section D. The number of individual decision trees (*ntree*) that composes the ensemble is fixed to 500, and the rest of the hyper-parameters of RF were set to their default values.

F. PERFORMANCE METRICS

Evaluating and comparing a ML-based NIDS solution relies on the use of performance metrics. However, the suitability of a metric depends on the application context. In this case, we are addressing a problem with a clear imbalance in the data. In fact, this is particularly the case of realistic network datasets which have great differences among the number of the positive classes (attacks) in comparison with the negative one (normal traffic or background). In such scenarios, it is highly not recommended to employ metrics like the accuracy, since a model may achieve high performance rates but is only capable of classifying correctly the class that is over-represented, thus not being able to correctly classify the minority class which is typically the most interesting one (i.e., any of the attacks in this work). Instead, it is better to employ performance metrics that are more suitable when dealing with imbalanced data. Therefore, in this work we chose the Area Under the Curve (AUC) to evaluate ML-based NIDSs, as this metric meets the requisite previously mentioned and it has been previously and successfully used in similar works [11], [13], [18], [45], [46].

The AUC measures the area under the Receiver Operating Characteristic (ROC) [47], which draws the evolution of the TP (True Positive) rate versus the FP (False Positive) rate for different values of the classifying threshold. Thus, AUC=1 means that the solution is able to correctly classify all the observations while AUC $\sim$ 0.5 means that the classifier performs randomly.

Besides, an AUC weighted average is computed as follows: for each class $i = 0, \ldots, k$, the $weighted\_avg(AUC_i)$ computes the weighted average of AUC_i times q_i (the number of true observations of each class), with Q being the total number of observations. Equation (5) introduces the AUC weighted average. Note that this equation can be easily extended to other performance metrics.

$$weighted\_avg(AUC_i) = \frac{\sum_{i=0}^{k} AUC_i \times q_i}{Q}. \tag{5}$$

G. SIGNIFICANCE ANALYSIS

In this work, a statistical study on the obtained results was performed to assess statistical certainty on our conclusions, at 95% confidence level. To correctly address it, we followed the recommendations given in [32] for the fair experimental evaluation of classifiers. In particular, we performed the Wilcoxon Signed-Rank Test [48] in the pairwise comparisons of the accuracy of the two studied classifiers on all classes. Wilcoxon Signed-Rank Test is a non-parametric version of the paired T-test. In this sense, statistically significantly better results with respect to the compared classifier, at 95%

TABLE 2. 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}

confidence level, are highlighted in *bold font* in all tables included in Section VI.

V. EXPERIMENTAL DESIGN

This sections describes the set-up of the R-NIDS framework introduced in Section IV at the same time that provides a detailed description of the evaluation strategy employed to test the performance of the R-NIDS we propose.

A. R-NIDS SET-UP

As the proposed R-NIDS methodology establishes, we first apply the FaaC feature engineering method on the original network datasets to obtain the derived ones. The application of FaaC requires building batches of observations, wherein the counters on the values of the variables are applied. With the aim of making a fair comparison on the performance of the ML-based solutions built in this study, authors decided to go for two versions of the derived datasets in such a way that the generated number of observations $(M_1, \ldots, M_x)$ is fixed: either 10,000 or 20,000 observations. Since the number of samples in the raw network datasets is different, the batch sizes —$(B_1, \ldots, B_x)$— used to aggregate the raw observation into new ones, that are counters of these ones, differ. These batch sizes are summarized as follows:

- 10,000 observations derived datasets: the batch size used to generate this derived dataset was {395.082, 254, 14} for the UGR'16, UNSW-NB15 and NSL-KDD raw datasets, respectively.
- 20,000 observations derived datasets: the batch size used to generate this derived dataset was {197.541, 127, 7} for the UGR'16, UNSW-NB15 and NSL-KDD raw datasets, respectively.

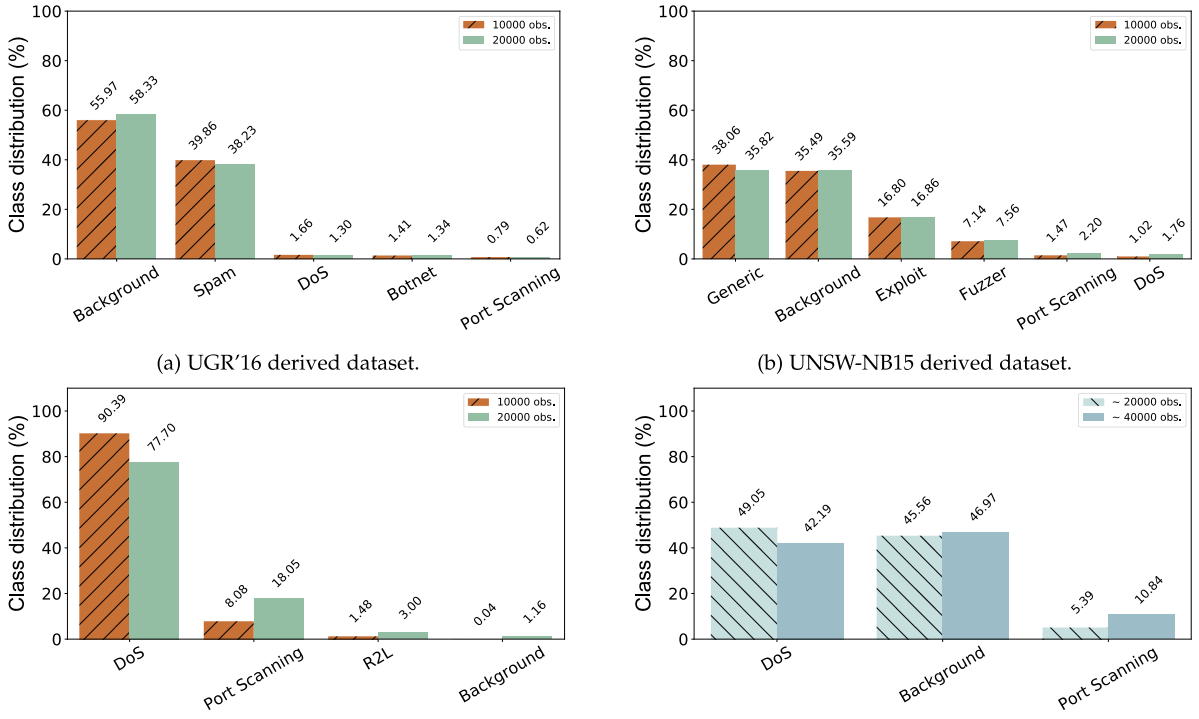
Although the number of derived observations (M_x) differ from dataset to dataset, the number of derived features (P') remains equal having $M_1 \times P', \ldots, M_x \times P' \,|\, x = 1, 2, 3$ different derived datasets. This is a direct consequence of using FaaC, which makes feature homogenization among heterogeneous datasets. Table 2 shows the relationships among selected raw features and derived features (counters). Last column in the table first shows how many derived features are extracted from one specific raw variable and, second, the derived features. The latter count the values (which are the ones within brakes in the table) they found within 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 NIDS [5], [10]–[13], [18], and it has been adopted here too. Further details can be found in [5].

Furthermore, the application of the FaaC technique using the setting described on each network dataset will also modify the class distribution within each dataset (see Section A). In this sense, Figures 2(a), 2(b), and 2(c) visualize the class distribution on both derived datasets for each of the three raw datasets. We can see how FaaC tends to balance them for the corresponding derived datasets of the UGR'16 and UNSW-NB15, that is, the percentages of all positive (attacks) classes and the negative (background) one are similar. However, we find the opposite behavior for the NSL-KDD dataset, mainly represented by observations of *DoS* attack and almost no background or normal traffic. This may be an indication which supports the procedure used to create this dataset using synthetic, i.e., unrealistic network flows.

Next, the R-NIDS framework aims at creating one single derived dataset that combines all the derived datasets generated by using the FaaC technique. For this purpose, only observations that correspond to *Background*, *DoS* or *Port Scanning* classes (i.e., the ones in common among the three datasets) are considered during the aggregation process. Then, and thanks to the variables homogenization achieved with the application of the FaaC technique, a simple row-wise aggregation procedure is used to merge all the observations of these three derived datasets, which met the criteria mentioned before, onto a single new integrated dataset that is hereafter called UNK22,[4] taking its name from the capital letters of the three involved network datasets. Two versions of this dataset are obtained depending on the number of observations that the derived datasets have. Thus, a UNK22 ∼20,000 observations dataset was built by following a row-wise data aggregation of the 10,000 observations derived datasets. Similarly, for the UNK22 ∼40,000 observations—note that only classes that are present in the three datasets are considered to generate UNK22—. The class distribution in UNK22 is shown in Figure 2(d). There is a balanced representation of the *Background* and *DoS* classes, although the *Port scanning* class is still underrepresented.

Finally, a last and key normalization step is performed in such a way that each variable of any observation within the UNK22 dataset is normalized by the batch size used to generate it according to the original raw network dataset. This

[4]The UNK22 dataset will be available for downloading at https://github.com/ucadatalab/ff4ml/tree/master/data/unk22.



(a) UGR'16 derived dataset.

(b) UNSW-NB15 derived dataset.

(c) NSL-KDD derived dataset.

(d) UNK22 row-wise aggregated dataset.

FIGURE 2. Class distribution on the derived datasets obtained by applying the FaaC method to the raw datasets included in Section III —see (a), (b) and (c)— and on the UNK22 row-wise aggregated dataset proposed in this paper —see (d)—.

procedure is performed as shown in Equation (6)

$$norm(\boldsymbol{x}) = \begin{cases} \frac{\boldsymbol{x}}{B_{UGR'16}}, & \text{if } \boldsymbol{x} \in \text{UGR'16 dataset} \\ \frac{\boldsymbol{x}}{B_{UNSW-NB15}}, & \text{if } \boldsymbol{x} \in \text{UNSW-NB15 dataset} \\ \frac{\boldsymbol{x}}{B_{NSL-KDD}}, & \text{if } \boldsymbol{x} \in \text{NSK-KDD dataset} \end{cases}, \tag{6}$$

where $\boldsymbol{x} = x_1, \ldots, x_p$ corresponds with the derived variables (counters) to be normalized and the $B_{dataset}$ is the batch size used to normalize them.

The presented normalization step guarantees that variables' values of all the observations within the UNK22 dataset are normalized counters in the [0,1] range, regardless their dataset of origin. Consequently, the ML-based solution build on this UNK22 dataset could be a more robust and reliable NIDS than others using the classic one-single-dataset approach. Moreover, this system will be ready to use in production and for generic network environments, since the application of this normalization procedure ensures that variables within new unseen observations will also lie in the same range as the expected by the trained models (i.e., in the [0,1] range), essential for their correct performance.

B. EVALUATION STRATEGY

In this study, two evaluation settings were considered

- Analysis of one single dataset: in this setting where ML-based solutions for NIDSs are built and tested on a single network dataset, the entire analysis was carried out using a 5-fold cross-validation strategy ($k = 5$ in Figure 1), where the complete dataset is divided into 5 folds of equal sizes, assuring that the class distribution remains the same within each fold. This strategy trains the ML models proposed as classifiers (which includes a feature selection through LASSO and a z-score data normalization steps before fitting the model to the data) in 4 out of the 5 folds and finally evaluates its performance on the test fold (the fifth one) left apart. Then, this procedure is repeated iteratively by rotating the folds used to train and test the classifiers. Furthermore, the fold partitioning process is repeated 20 times ($r = 20$ in Figure 1) in order to guarantee the randomness of this process.
- Analysis across datasets: the analysis carried out in this setting is much more simple, since it aims at training the ML models proposed as classifiers in one out of the three network datasets considered, and evaluate their performance in the remaining two datasets, not used to train the models. This procedure is repeated iteratively by rotating the complete dataset used to train the classifiers.

The former setting aims at performing the classical ML-based NIDS evaluation that is typically found in the literature, where the community focuses on developing ML-based solutions using one single network dataset, thus forgetting to double-check how well the ML-based solution generalizes with different datasets (i.e., gathered in a different network environments). The latter setting tries to complete the big picture by showing the performance of these kind of ML-

TABLE 3. AUC performance for the models trained on every dataset and evaluated on the others.

Size	Model	Train data \ Test data	UGR'16			USNW-NB15			NSL-KDD		
			Background	DoS	Port Scanning	Background	DoS	Port Scanning	Background	DoS	Port Scanning
10,000 obs.	LR	UGR'16	0.894	0.977	0.980	0.490	0.500	0.494	0.500	0.500	0.500
		USNW-NB15	0.342	0.499	0.780	0.994	0.954	0.875	0.712	0.489	0.434
		NLS-KDD	-	0.500	0.282	-	0.869	0.473	0.983	0.761	0.731
	RF	UGR'16	0.962	0.987	0.997	0.695	0.731	0.565	0.663	0.629	0.474
		USNW-NB15	0.508	0.832	0.499	0.998	0.978	0.930	0.894	0.571	0.543
		NLS-KDD	-	0.586	0.150	-	0.942	0.157	0.950	0.723	0.698
20,000 obs.	LR	UGR'16	0.893	0.985	0.988	0.497	0.500	0.499	0.478	0.500	0.500
		USNW-NB15	0.293	0.502	0.870	0.985	0.885	0.882	0.623	0.595	0.427
		NLS-KDD	0.747	0.689	0.325	0.623	0.740	0.524	0.985	0.749	0.710
	RF	UGR'16	0.962	0.987	0.997	0.506	0.392	0.536	0.329	0.460	0.562
		USNW-NB15	0.630	0.657	0.120	0.992	0.920	0.911	0.756	0.554	0.549
		NLS-KDD	0.579	0.454	0.398	0.827	0.460	0.562	0.979	0.738	0.707

Results with gray background are obtained on the same dataset used for trainig, following the cross-validation setting shown in Section S.II, available online.

based solutions when these models are used to predict attacks on a different network environment than the one used to train them. In both of these settings, the feature and hyper-parameter selection were carried out following the methods introduced in Sections B and D, respectively. The former relying on LASSO technique while the latter through Bayesian optimization both in the training set. Finally, the AUC previously introduced in Section F is chosen as a performance metric because it is found suitable when dealing with imbalanced network datasets for comparing and evaluating NIDSs.

VI. RESULTS AND DISCUSSION

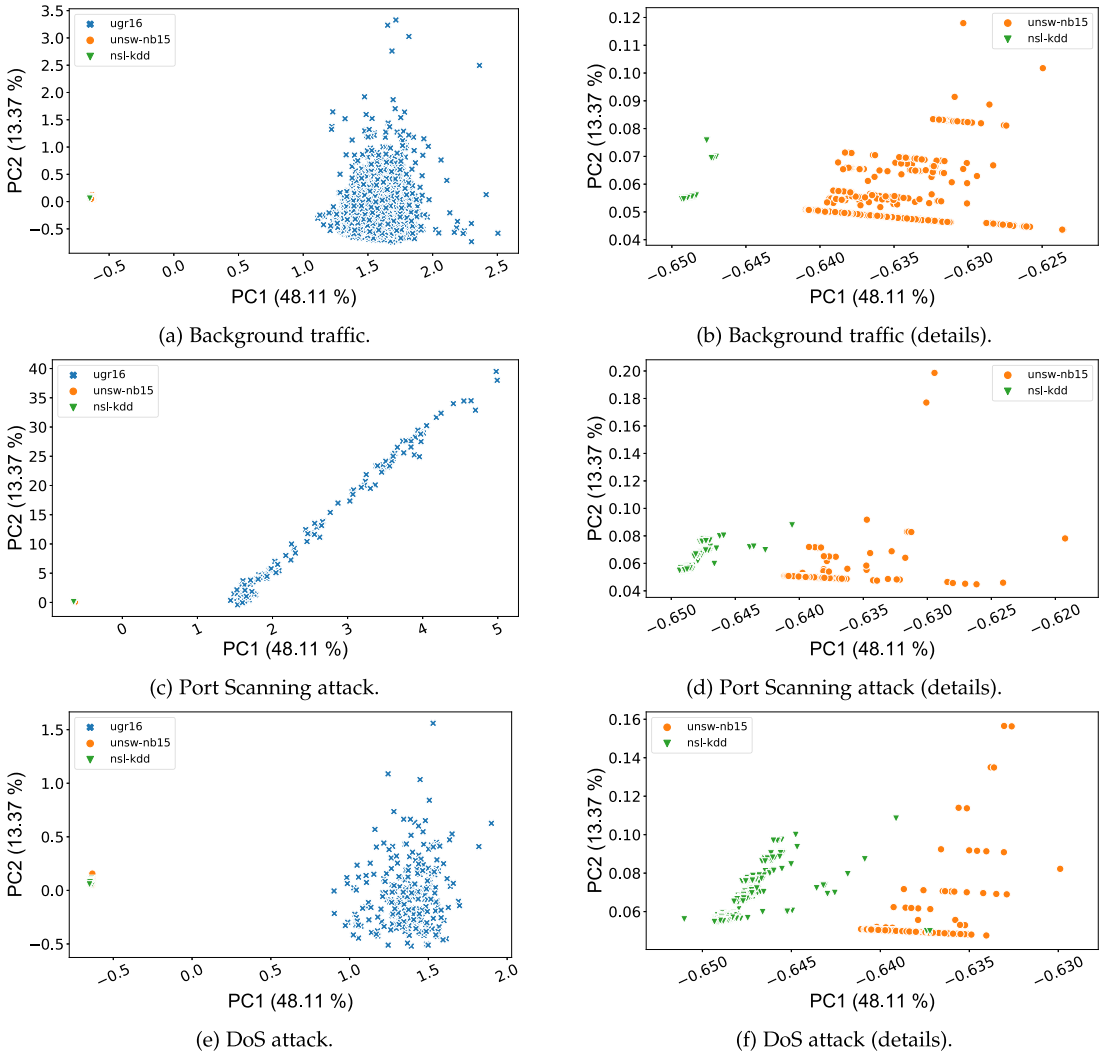
This section summarizes the results of the analysis and discusses the main findings. We first compare the performance of LR and RF classifiers when they are individually built on one of the above introduced network datasets: UGR'16, USNW-NB15, and NSL-KDD. The results are given in the cells with gray background color in Table 3 for the common attacks in the three datasets, and more detailed results are provided in Section S.II of the supplementary material, available online.

In order to verify the behaviour of these kind of ML-based solutions in generic network environments, another analysis across network datasets was performed. In this sense, Table 3 shows the AUC performance results when ML models (LR and RF) are fully trained using one of the previously mentioned datasets (train data column in the table) and fully tested on the other NIDS datasets (test data field in the table). This crossed analysis was carried out only for the common classes in the three datasets (i.e., *Background*, *DoS* and *Port Scanning*). Gray cells in this table correspond to the performance presented in Tables S4-S6 of the supplementary material, available online, —i.e., analysis on one single dataset, which in the R-NIDS framework would occur when $x = 1$ in Figure 1—. Cells without numeric values means that no confident results were obtained mainly due to the low number of samples available to train the model for that specific class.

These results clearly expose the issues of ML-based NIDS solutions when they are built exclusively on one single dataset expecting them to perform well in generic network environments (i.e., in observations belonging to other network datasets not used to train the models). In fact, the performance of ML-based solutions drastically drops in this more realistic scenario, since the aim of this kind of solutions is to put them in production to detect possible attacks in generic network environments. In other words, we can safely conclude that the classic approach employed to build ML-based NIDS using one single dataset tend to overfit to observations of the network environment used to create that specific dataset, but results will very rarely generalize to other network environments.

We computed some statistics on the results in Table 3 to support the previous conclusion. Particularly, the percentage of the mentioned accuracy loss of the models was calculated as an average for the three attacks. The highest loss was a 116.6% decay in the performance of RF when trained with UGR'16 and tested on NSL-KDD, for 20,000 observations. In average, the models offered a 67.7% accuracy loss when tested in other datasets different than the one used for training. The average accuracy of the models for all classes when trained and tested on the same dataset is 0.905, while it worsens to 0.533 when a different dataset is used for testing.

For a better understanding of these results, an additional combined Principal Component Analysis (PCA) study was carried out to unveil hidden behaviors of the involved classes and that, otherwise, it would be difficult to observe. The graphical analysis (score plots) performed on the derived datasets of 20,000 observations is shown in Figure 3. Since the equivalent analysis for the 10,000 observations version of the derived datasets is very similar, authors decided to omit it. Focusing on the three plots on the left of Figure 3, one can clearly distinguish two clusters independently of the class analyzed (*Background*, *DoS* or *Port Scanning*). This result points out that UGR'16 observations are behaving completely different from the ones in the other two datasets.



(a) Background traffic.

(b) Background traffic (details).

(c) Port Scanning attack.

(d) Port Scanning attack (details).

(e) DoS attack.

(f) DoS attack (details).

FIGURE 3. Results of the PCA analysis of the three NIDS datasets (UGR'16, UNSW-NB15, NSL-KDD), for each class.

In fact, this also stands out in the results included in Table 3, where the performance of the ML-based solutions trained on the UGR'16 dataset drastically drops when they are tested on any of the other two datasets.

Moreover, on the three plots on the right of Figure 3 we zoomed the graphical analysis of the corresponding score plots on the left side to show some details within the cluster. On these three plots, one can see two clusters of slightly separated observations corresponding to the USNW-NB15 and NSL-KDD datasets. Besides, trend differences are also visible, specially for the *Background* and *Port Scanning* classes. However, this is not the case for the *DoS* class where observations behave similarly in both datasets. Thus, one could expect a better performance in the *DoS* class than in the others when training or testing with USNW-NB15 and NSL-KDD datasets which is, in general, observed in Table 3.

One last analysis was carried out on the UNK22 dataset in order to measure the benefits of using the R-NIDS framework when combining several network datasets. In this sense, the results included in column "Overall" in Table 4 show how ML-based models perform when using the whole UNK22 for training and testing. As it can be seen in the table, models achieve, on average, an AUC $\approx$ 0.8 for the *Port Scanning* attack, AUC $\approx$ 0.95 for the *DoS* attack and AUC $\approx$ 0.995 for the *Background* traffic. In comparison with the values in Tables S4-S6, available online, the UNK22 slightly degrades the obtained performance for *DoS* and *Port Scanning* attacks in general. It can be motivated by the influence of the NSL-KDD dataset that pollutes, in some sense, the UNK22 dataset: almost the half part of the *DoS* and *Port Scanning* observations are added by the NLS-KDD (see Figure 2). This fact, together with the nature of the NSL-KDD dataset, makes the models to perform slightly worse. Nevertheless, the most interesting results arise when we focus on analysing the performance of these ML-based models on observations of only one of the network datasets, i.e., training them with the UNK22 dataset and testing only with samples corresponding to the specific network datasets used to build UNK22. These results can be seen in columns "Only UGR'16", "Only UNSW-NB15" and "Only NSL-

TABLE 4. Average AUC performance for UNK22 dataset using a 20 repetitions of 5-fold cross-validation evaluation strategy.

Dataset	Model	Class	Overall	Only UGR'16	Only UNSW-NB15	Only NSL-KDD
UNK22 ~20,000 obs.	LR	Background	0.997	0.974	0.822	**0.780**
		DoS	**0.962**	0.960	0.901	**0.576**
		Port Scanning	0.766	0.985	0.756	**0.575**
		Weighted avg.	*0.967*	*0.974*	*0.822*	**0.574**
	RF	Background	**0.998**	**0.983**	**0.892**	0.676
		DoS	0.960	**0.978**	**0.937**	0.542
		Port Scanning	0.765	**0.995**	**0.849**	0.539
		Weighted avg.	*0.967*	**0.984**	**0.891**	*0.539*
UNK22 ~40,000 obs.	LR	Background	0.992	0.973	0.742	0.809
		DoS	0.928	0.962	0.789	**0.606**
		Port Scanning	0.791	0.985	0.710	0.608
		Weighted avg.	*0.943*	*0.973*	*0.743*	**0.609**
	RF	Background	**0.994**	**0.987**	**0.831**	0.819
		DoS	0.928	**0.985**	**0.877**	0.600
		Port Scanning	**0.799**	**0.991**	**0.780**	0.607
		Weighted avg.	**0.945**	*0.987*	*0.830*	*0.604*

KDD" in Table 4. Overall, a high AUC of any of the ML models in these three columns would imply a better generalization, thus a better performance, on heterogeneous network environments. In concrete, the homogenization and aggregation of network datasets provided by the R-NIDS framework allowed to obtain reliable ML-based solutions that perform better in generic network environments than those built using the classic one-single-dataset approach. For instance, a RF model trained on the 20,000 observations version of the UNK22 dataset achieves an AUC of {0.983, 0.892, 0.676} for the *Background* class in the UGR'16, UNSW-NB15 and NSL-KDD datasets, respectively. However, in Table 3 the best results obtained for exactly the same setting were {0.508, 0.814, 0.712}, which is a significant improvement in two out of the three datasets. Similarly, a RF model trained on the 40,000 observations version of the UNK22 dataset achieves an AUC of {0.991, 0.780, 0.607} for the *Port Scanning* class in the UGR'16, UNSW-NB15 and NSL-KDD datasets, respectively, while in Table 3 the best results obtained for exactly the same setting were {0.398, 0.562, 0.562}, which in this case is a significant improvement in all the datasets. This pattern is generally observed in Table 4 regardless the ML model and network dataset in which one wants to focus on.

Finally, though far from the main aim of the work, we analyzed the most relevant features in the datasets, concluding that those related to network services (source and destination ports) play a relevant role in training the ML-models, followed by those associated to the protocol and the number of packets sent in the communications. All the details are given in Section S.III of the supplementary material, available online.

VII. CONCLUSION

This work has presented a novel methodology for the design of Machine Learning (ML) based solutions for trustworthy and reliable Network Intrusion Detection Systems (NIDS), called Reliable-NIDS (or R-NIDS for short). In contrast to the existing methods, the ML models designed following the R-NIDS methodology achieve more accurate predictions in different network scenarios. Thus, they arise as more suitable tools for being deployed in a real environment, compared to previously existing ones.

There is a large number of works in the literature addressing this problem with ML-based methods, and also several network datasets have been proposed. All these works use the same dataset for training and validating their models, and unsurprisingly highly accurate results are published in most of the cases. However, this approach leads to models which overfit data from the given network environment used to train them, thus not being able to generalize to new data that can be present in generic scenarios or under different conditions. Therefore, these existing methods are not suitable to be deployed in real environments.

Through an extensive experimentation, this work has demonstrated how two well-known ML models (selected as widely used linear and non-linear methods), commonly used for NIDS, suffer from the mentioned overfitting problem when trained on one network dataset and validated on a different one. Indeed, the obtained performance loss was up to 116.6% when validating the model on a different dataset. For this study, we chose three well accepted datasets, namely UGR'16, USNW-NB15, and NSL-KDD. Although these datasets have different features and making a joint analysis was a priory unfeasible, it turned out to be possible thanks to the use of the FaaC feature engineering method present in R-NIDS. This technique makes the homogenization of different network datasets possible, creating different derived datasets from them (all having the same variables), thus being a key driver to perform data aggregation over several heterogeneous datasets.

Additionally, this work has introduced a novel dataset, called UNK22. It was built from the samples contained in three of the most studied datasets in the literature (UGR'16,

USNW-NB15, and NSL-KDD), that could be integrated using the R-NIDS methodology proposed. Last but not least, by using the UNK22 dataset this paper has shown how the two ML models considered in this work were able to avoid overfitting, thus leading to more reliable ML-based NIDS solutions compared to those trained on just one single network dataset. The global performance of the models trained on UNK22 is on average 90.67% when testing on unseen samples of UNK22, and its accuracy decreases by only 9.05% when separately tested on each of the three datasets taken from the literature. This accuracy error rises up to 39.15% when the models are trained on one of the three datasets and tested on the others. This result proves how it is possible to generate more reliable models thanks to the integration of the datasets and the methodology proposed in this work.

Finally, this work opens interesting research lines in the field. In general, we are interested in extending the study with other state-of-the-art ML methods from the literature, as well as with other existing relevant datasets. Besides, and given the importance of integrating different datasets for building generic methods as it has been shown in this paper, the authors consider interesting to work towards the design and exploration of other generalization mechanisms based on different ways of integrating the main network datasets in the field with minimum information loss.

REFERENCES

[1] *Cisco Annual Internet Report (2018–2023)*, White Paper, 2020. Accessed: May 27, 2022. [Online]. Available: https://bit.ly/3jpAgNx

[2] *Enisa Threat Landscape Report*, 2020. Accessed: May 27, 2022. [Online]. Available: https://bit.ly/3mbH56U

[3] D. Chou and M. Jiang, "A survey on data-driven network intrusion detection," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 182:1–182:36, 2021.

[4] T. Zoppi and A. Ceccarelli, "Prepare for trouble and make it double! Supervised – Unsupervised stacking for anomaly-based intrusion detection," *J. Netw. Comput. Appl.*, vol. 189, pp. 103–106, 2021.

[5] 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," *Appl. Sci.-Basel*, vol. 10, no. 5, 2020, Art. no. 1775.

[6] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," *Comput. Netw.*, vol. 136, pp. 37–50, 2018.

[7] E. Kabir, J. Hu, H. Wang, and G. Zhuo, "A novel statistical technique for intrusion detection systems," *Future Gener. Comput. Syst.*, vol. 79, pp. 303–318, 2018.

[8] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. Cambridge, MA, USA: MIT Press, 2009.

[9] T. J. T. Heiser, M.-L. Allikivi, and M. Kull, "Shift happens: Adjusting classifiers," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2019, pp. 55–70.

[10] J. Camacho, G. Macía-Fernández, J. Díaz-Verdejo, and P. García-Teodoro, "Tackling the Big Data 4 vs for anomaly detection," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2014, pp. 500–505.

[11] J. Camacho, A. Pérez-Villegas, P. García-Teodoro, and G. Maciá-Fernández, "PCA-based multivariate statistical network monitoring for anomaly detection," *Comput. Secur.*, vol. 59, pp. 118–137, 2016.

[12] 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," *Comput. Secur.*, vol. 87, 2019, Art. no. 101603.

[13] J. Camacho, G. Maciá-Fernández, N. M. Fuentes-García, and E. Saccenti, "Semi-supervised multivariate statistical network monitoring for learning security threats," *IEEE Trans. Inf. Forensic Security*, vol. 14, no. 8, pp. 2179–2189, Aug. 2019.

[14] 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," *Comput. Secur.*, vol. 73, pp. 411–424, 2018.

[15] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf.*, 2015, pp. 1–6.

[16] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, 2009, pp. 1–6.

[17] 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 *Proc. 14th Int. Conf. Comput. Intell. Secur. Inf. Syst.*, 2022, pp. 116–125.

[18] R. Magán-Carrión, J. Camacho, G. Maciá-Fernández, and A. Ruíz-Zafra, "Multivariate statistical network monitoring–Sensor: An effective tool for real-time monitoring and anomaly detection in complex networks and systems," *Int. J. Distrib. Sensor Netw.*, vol. 16, no. 5, 2020, Art. no. 155014772092130.

[19] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[20] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2019, pp. 108–116.

[22] M. Siddiqi and W. Pak, "Efficient filter based feature selection flow for intrusion detection system," in *Proc. Int. Workshop Emerg. ICT*, 2020, pp. 1–3.

[23] 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," *Cluster Comput.*, vol. 23, pp. 1397–1418, 2020.

[24] Q. Tian, D. Han, K.-C. Li, X. Liu, L. Duan, and A. Castiglione, "An intrusion detection approach based on improved deep belief network," *Appl. Intell.*, vol. 50, pp. 3162–3178, 2020.

[25] A. Verma and V. Ranga, "Machine learning based intrusion detection systems for IoT applications," *Wireless Pers. Commun.*, vol. 111, no. 4, pp. 2287–2310, 2020.

[26] B. Riyaz and S. Ganapathy, "A deep learning approach for effective intrusion detection in wireless networks using CNN," *Soft Comput.*, vol. 24, pp. 17265–17278, 2020.

[27] A. M. Aleesa, M. Younis, A. A. Mohammed, and N. M. Sahar, "Deep-intrusion detection system with enhanced UNSW-NB15 dataset based on deep learning techniques," *J. Eng. Sci. Technol.*, vol. 16, no. 1, pp. 711–727, 2021.

[28] 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," *MDPI-Electron.*, vol. 10, no. 15, 2021, Art. no. 1854.

[29] T. S. Pooja and S. Purohit, "Evaluating neural networks using bi-directional LSTM for network IDS (intrusion detection systems) in cyber security," *Glob. Transitions Proc.*, vol. 2, pp. 448–454, 2021.

[30] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Netw. Appl.*, vol. 27, pp. 357–370, 2022.

[31] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami, "Multi-stage optimized machine learning framework for network intrusion detection," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1803–1816, Jun. 2021.

[32] K. Stapor, P. Ksieniewicz, S. García, and M. Woźniak, "How to design the fair experimental classifier evaluation," *Appl. Soft. Comput.*, vol. 104, 2021, Art. no. 107219.

[33] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Secur.*, vol. 86, pp. 147–167, 2019.

[34] A. Kenyon, L. Deka, and D. Elizondo, "Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets," *Comput. Secur.*, vol. 99, 2020, Art. no. 102022.

[35] *Enisa Threat Landscape Report*, 2017. Accessed: May 27, 2022. [Online]. Available: https://bit.ly/3KLBIV8

[36] *Enisa Threat Landscape Report*, 2018. Accessed: May 27, 2022. [Online]. Available: https://bit.ly/3q629gi

[37] R. Magán-Carrión, J. Camacho, and P. García-Teodoro, "Multivariate statistical approach for anomaly detection and lost data recovery in wireless sensor networks," *Int. J. Distrib. Sens. Netw.*, vol. 11, no. 6, 2015, Art. no. 672124.
[38] D. Urda, J. Montes-Torres, F. Moreno, L. Franco, and J. M. Jerez, "Deep learning to analyze RNA-seq gene expression data," in *Proc. Int. Work-Conf. Artif. Neural Netw.*, 2017, pp. 50–59.
[39] D. Urda *et al.*, "BLASSO: Integration of biological knowledge into a regularized linear model," *BMC Syst. Biol.*, vol. 12, 2018, Art. no. 94.
[40] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *J. Statist. Softw.*, vol. 33, no. 1, pp. 1–22, 2010.
[41] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives," in *Proc. IEEE 3rd Int. Conf. Comput. Commun. Secur.*, 2018, pp. 1–8.
[42] C. Chio and D. Freeman, *Machine Learning and Security: Protecting Systems With Data and Algorithms*. Sebastopol, CA, USA: O'Reilly Media, 2018.
[43] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.
[44] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2951–2959.
[45] S. Soufiane, R. Magán-Carrión, I. Medina-Bulo, and H. Bouden, "Preserving authentication and availability security services through multivariate statistical network monitoring," *J. Inf. Secur. Appl.*, vol. 58, 2021, Art. no. 102785.
[46] F. Pérez-Bueno, L. García, G. Maciá-Fernández, and R. Molina, "Leveraging a probabilistic PCA model to understand the multivariate statistical network monitoring framework for network security anomaly detection," *IEEE-ACM Trans. Netw.*, 2022, pp. 1–13, doi: 10.1109/TNET.2021.3138536.
[47] F. Salo, M. Injadat, A. B. Nassif, A. Shami, and A. Essex, "Data mining techniques in intrusion detection systems: A systematic literature review," *IEEE Access*, vol. 6, pp. 56 046–56 058, 2018.
[48] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric Statistical Methods*. Hoboken, NJ, USA: Wiley, 2013.

ROBERTO MAGÁN-CARRIÓN received the PhD degree in ICT from the University of Granada with a Cum Laude grade, in 2016. He is currently an assistant professor with the Department of Signal Theory, Telematics and Communications with the University of Granada, Spain, and member of "Network Engineering and Security Group (NESG)" research group. He also worked as a postdoctoral fellow with the University of Cádiz, as part of a research talent attraction international program promoted by this university. His research interests include security in heterogeneous communications networks and systems, specifically on anomaly detection and classification, response, and resilient solutions.

DANIEL URDA received the PhD degree in computer science from the University of Málaga, Spain, and is currently an assistant professor with the University of Burgos, Spain. His research mainly involves the development and application of machine learning methods for several domains such as bioinformatics, cybersecurity, air pollution, logistics and transportation and industry. He has a strong publication record in relevant journals and international conferences (more than 50 papers) and has been guest editor of several special issues in well-known journals related to *Computer Science*. He is responsible for teaching several courses of bachelor's and master's degree in computer science like artificial intelligence or neural computing, among others.

IGNACIO DIAZ-CANO is a predoctoral researcher with the Department of Automatic, Electronic, Computer Architecture and Communication Networks Engineering from the University of Cádiz, Spain. His research mainly involves two main areas of knowledge: cybersecurity and industrial robotics, with some publications and conferences in both areas.

BERNABÉ DORRONSORO received the PhD degree in computer science from the University of Málaga, Spain, in 2007. He worked for more than seven years in Luxembourg and Lille Universities, and he currently serves as an Associate Professor with the University of Cádiz, Spain, where he leads the GOAL research group. His main research interests include green computing, sustainable transportation, grid and cloud computing, smart cities, complex problems optimization, and machine learning. He has published more than 50 journal papers and six books. He is associate editor of the *International Journal of Metaheuristics*, and member of the editorial board of several journals: *Engineering Applications of Artificial Intelligence*, *Applied Sciences*, *International Journal of High Performance Systems Architecture*, *International Journal of Innovative Computing and Applications*, and *Progress in Artificial Intelligence*.