future internet

Article

A Local Feature Engineering Strategy to Improve Network Anomaly Detection †

Salvatore Carta ‡, Alessandro Sebastian Podda ‡, Diego Reforgiato Recupero ‡ and Roberto Saia *,‡

Department of Mathematics and Computer Science, University of Cagliari, 09124 Cagliari, Italy; salvatore@unica.it (S.C.); sebastianpodda@unica.it (A.S.P.); diego.reforgiato@unica.it (D.R.R.)

* Correspondence: roberto.saia@unica.it

† This is an extended version of our paper accepted at the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, 2020.

‡ These authors contributed equally to this work.

Received: 10 September 2020; Accepted: 18 October 2020; Published: 21 October 2020

Abstract: The dramatic increase in devices and services that has characterized modern societies in recent decades, boosted by the exponential growth of ever faster network connections and the predominant use of wireless connection technologies, has materialized a very crucial challenge in terms of security. The anomaly-based intrusion detection systems, which for a long time have represented some of the most efficient solutions to detect intrusion attempts on a network, have to face this new and more complicated scenario. Well-known problems, such as the difficulty of distinguishing legitimate activities from illegitimate ones due to their similar characteristics and their high degree of heterogeneity, today have become even more complex, considering the increase in the network activity. After providing an extensive overview of the scenario under consideration, this work proposes a Local Feature Engineering (LFE) strategy aimed to face such problems through the adoption of a data preprocessing strategy that reduces the number of possible network event patterns, increasing at the same time their characterization. Unlike the canonical feature engineering approaches, which take into account the entire dataset, it operates locally in the feature space of each single event. The experiments conducted on real-world data showed that this strategy, which is based on the introduction of new features and the discretization of their values, improves the performance of the canonical state-of-the-art solutions.

Keywords: intrusion detection; anomaly detection; networking; data preprocessing; machine learning

1. Introduction

The literature offers several definitions of the intrusion event, e.g., Sundaram [1] defined it as an attempt to bypass or compromise a specific objective, which can be a single machine or an entire network. The literature also indicates the CIA model (also called CIA triad or CIA triangle) as the security model to take into account in terms of goals to achieve [2]. It defines the three main requirements of security, namely *confidentiality*, *integrity*, and *availability*, as summarized in Figure 1. The *confidentiality* requirement expresses the need that a resource related to a machine/network must be accessed only by authorized users; the *integrity* requirement is related to the need that a resource can be modified only by authorized users, according to their permissions; and the *availability* requirement means that a resource must be available to authorized users in the times established for it, without any limitation or interruption.

In the context of network security, *Intrusion Detection Systems* (*IDSs*) [3] represent the main tool, as in their different configurations and modalities they are able to analyze the events that occur in a network, with the aim of detecting the illegitimate ones. Nowadays, this activity has become crucial due to the huge number of network services, public and private, which involve critical and important areas, such as health services [4], education [5], financial services [6], and so on.

One of the most crucial transversal security problems, that of privacy, was deeply investigated by Verykios et al. [7], who provided an overview on the area of privacy-preserving data mining, whereas Narwaria and Arya [8] provided a comparative analysis of some well-known privacy preservation techniques. However, the massive growth of network security risks is given by the contribution of different factors, such as the increase of wireless network connections [9], the advent of the Internet of Thing (IoT) [10], and the large development of new broadband infrastructures (optical fiber [11], 5G [12], etc.). Therefore, in the context of the network protection, the choice of using IDSs together with other canonical protection systems (e.g., firewalls [13]) mainly depend on the inability of these to operate extensively, without previously defined rules.

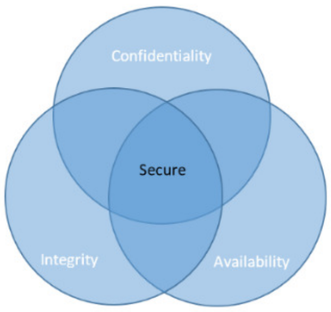

Figure 1. Security CIA model.

Given this background, the literature identifies different types of IDSs: although they can operate in different manner, their common objective is the analysis and classification of each network event, distinguishing the normal activities from the intrusion ones. For instance, the literature reports intrusion detection approaches based on machine learning criteria such as gradient boosting [14], adaptive boosting [15], and random forests [16]. Other proposals involve artificial neural networks [17], probabilistic criteria [18], or data transformation/representation [19–21], similarly to what is done, in terms of scenario and data balance, in closely related domains [22–30]. It should be noted that, besides sharing the same objectives, they also tackle analogous problems, such as the difficulty of classifying intrusion events that are very similar to normal ones in terms of characteristics or the difficulty to detect novel form of attacks (e.g., zero-days attacks [31]).

For these reasons, after having positively evaluated the effectiveness of data transformation in other fields [32–34], which show it to be able to achieve better characterization of the involved information and a clear reduction of the possible feature patterns, we experimented on this strategy in the IDS domain. In more detail, the proposed LFE strategy operates by introducing a series of new features calculated on the basis of the existing ones, along with a discretization of each feature value, according to a numeric range experimentally defined. The combination of these two data pre-processing steps offers a better characterization of each network event (introduction of new features) and helps to reduce the number of feature patterns by aggregating the similar ones (discretization process).

Scientific Contribution

This work represents an extension of a previous one [32], completely rewritten and extended, formally and substantially, by updating and expanding the background and related work under the light of the recent literature, as well as providing an extensive overview of the scenario under consideration. Specifically, each provided reference is verified and updated if necessary, and further aspects—very close to the research area taken into account—are added and discussed, such as the analysis of the common classes of network attacks, and the process of assessment of the network vulnerabilities.

A novel contextualization of this work in the feature engineering technique area is also performed, underlining advantages and differences with regard to the canonical literature exploitation of such a technique. In addition, the formalization of the algorithms related to the proposed strategy is detailed, and the related impact in terms of time complexity is analyzed and discussed. A detailed description of the involved state-of-the art algorithms of classification, in terms of literature references, operating principles, and used parameters, is also provided.

Detailed and extensive information about the process of evaluation in terms of the three adopted metrics is added, with the aim to offer an in-deep analysis of the experimental results in terms of both each single metrics and their average. An overview about the overall mean performance of the proposed LFE strategy, with regard to its application in the context of all the algorithms (aimed to estimate its behavior among the most common classification algorithms in the literature), and all the network events (aimed to estimate its behavior for the different types of intrusion events), is also provided. The main contributions related to this work can then be summarized as follows:

- a definition of the proposed LFE strategy in the intrusion detection context, with regard to the current literature, along with a preliminary analysis of the IDS solutions and the involved network vulnerabilities, in terms of common classes of network attacks;
- a formalization of the LFE strategy in the context of the feature engineering technique area, presenting its implementation, advantages, and differences, with regard to the canonical literature use;
- a definition of a classification algorithm that exploits the LFE strategy to perform the network events classification into two classes (i.e., normal or intrusion), together with the evaluation of its asymptotic time complexity;
- a detailed categorization of the events included in the widely adopted NSL-KDD dataset, through the definition of a novel taxonomy, and an independent experimental evaluation of our strategy carried out separately by intrusion typology.
- an evaluation of the proposed LFE strategy in the context of additional datasets, aimed to evaluate its capabilities to face real-world scenarios characterized by heterogeneous type of network attacks; and
- an extensive validation of our LFE strategy, made by comparing it to a series of largely used solutions based on different state-of-the-art algorithms, using real-world data and three different metrics of evaluation, whose results are evaluated both individually and in terms of their average.

This work is structured as follows. Section 2 provides related works in the domain, whereas Section 3 includes background information, metrics for evaluation, and IDS essential concepts. Section 4 presents the formal notation used in this work, together with the formulation of our objective. Section 5 describes the implementation of the proposed strategy, from the feature engineering approach to the algorithm formalization and complexity analysis. Section 6 provides information on the development environment, the adopted real-world dataset and evaluation metrics, and the validation modality. Section 7 analyzes and discusses the experimental results, with regard to the canonical solutions. Section 8 concludes this work by recapping the obtained results and introducing promising directions for future research.

2. Related Work

According to Section 1, where we implicitly underline the great heterogeneity that characterizes the intrusion detection domain, in terms of both approaches and strategies, here we discuss some significant solutions in the literature.

Considering that the proposed strategy is based on the data preprocessing of the original feature space, it should be noted how this represents a common strategy in a number of works in the literature, albeit with different criteria. Such data preprocessing can involve many techniques, from the feature selection [35] ones, aimed to select a subset of representative features to use during the evaluation model definition process, to the feature engineering [36] ones, where the data domain knowledge is exploited to define/redefine features in order to improve the performance. An example of feature selection technique used in the context of the network anomaly intrusion detection can be found in [37], where the authors proposed a framework aimed to improve the performance by exploiting a feature selection technique. An approach based on the mutual information used to perform the feature selection in the intrusion detection context is defined in [38], whereas an interesting approach that combines several feature selection algorithms is presented in [39]. Another interesting approaches of feature selection is presented in [40], where the authors formalize such a technique in terms of a pigeon-inspired optimizer algorithm, applying it in an intrusion detection scenario. In [41], the feature selection technique is considered in the context of the big data scenario, since the authors proposed a penalty-based wrapper objective function to evaluate the feature selection process, whereas, in [42], the same authors faced the feature selection problem in a high-dimensional data scenario.

Several feature engineering techniques can be found in the current literature; for example, Kasongo and Sun [43] proposed a deep learning method for the intrusion detection task, based on a feature engineering process and Wang et al. [44] applied a feature engineering criterion that learns the hierarchical spatial-temporal aspects of the involved features through deep neural networks, in order to improve the intrusion detection performance. Recently, an interesting approach is proposed in [45], where the authors defined a novel feature learning model for cybersecurity tasks. Kunang et al. [46] proposed an automatic feature extraction technique based on autoencoder (i.e., a type of artificial neural network aimed to learn efficient data encodings, by following an unsupervised strategy) and support vector machine for the intrusion detection task. Recently, Ieracitano et al. [47] combined the autoencoder technique with a statistical analysis, in order to define an intrusion detection approach.

Although they present some overlap, feature selection and feature engineering techniques have different objectives; however, the proposed strategy operates in a different manner with respect to both of them. This happens since, instead of performing a selection of the most relevant features or defining some new ones on the basis of the entire dataset via data mining techniques, we first introduce new features based on statistical metrics (minimum, maximum, average, and standard deviation), and then we discretize the value of each feature, reaching a better characterization of each event, without significant computational efforts.

Both components of the proposed strategy operate synergistically, because the introduction of new features (a process aimed to improve the event characterization) counteracts the loss of information related to the data discretization (a process aimed to reduce the potential event patterns). It should also be observed that the strategy we proposed is not in direct competition with the intrusion detection approaches in the literature, as its main objective is to improve their intrusion detection performance by preprocessing the involved data, regardless of the type of algorithm they adopted.

3. Background

To start, we can observe how the literature classifies several types of Intrusion Detection Systems (IDSs), along with the definition of the intrusion detection concept, which was first introduced by James Anderson in 1980 [48], and later formalized by Dorothy Denning in 1987 [49]. In the following years, many works and surveys have deepened and discussed this concept, up to the present day. In 2000, Axelsson [50] discussed the taxonomy of intrusion detection systems, presenting a survey and their classification. In 2008, Modi et al. [51] proposed a survey aimed to study how the different intrusions can affect the availability, confidentiality, and integrity of cloud resources and services. Another interesting work was presented in 2017 by Zarpel ao et al. [52], who extended the intrusion detection concept to the IoT scenario, where the authors tried to identify leading trends, open issues, and future research possibilities, classifying the IDSs formalized in the literature on the basis of several attributes.

3.1. Intrusion Detection Systems

The main task of an IDS is the analysis and classification of the network activity, which is aimed to discriminate the normal network events from the intrusion ones. The concept of intrusion can be related both to a software activity, such as malware [53,54] (e.g., spyware, virus, rootkit, etc.) and to a human activity [55] addressed to the illegitimate exploitation of network resources.

3.1.1. Operative Ways

Although there are many ways to classify the IDSs, one of the most used in the literature classifies them into four categories: anomaly-based network IDSs, signature-based network IDSs, specification-based network IDSs, and Hybrid-based network IDSs. They are summarized as follows:

- *Anomaly-based network IDSs*, also named behavior-based network IDSs, operate by analyzing and classifying each network event in two classes, normal or intrusion. Instead of performing a process of comparison between the new network events and a database that contains the signatures/patterns related to the known ones, the classification process exploits a heuristic or rules-based strategy aimed to detect the intrusion in terms of events not attributable to a normal network activity. It should be noted that, despite the huge research effort in this area, as well as the huge number of such systems formalized in the literature, problems to be solved continue to exist [56];
- *Signature-based network IDSs*, also named knowledge-based network IDSs, operate by comparing the pattern of each new network event to a database of known signatures/patterns [57]. In [58], the authors proposed to face the problem related to the significant overheads in terms of memory usage and execution time related to the signature/pattern matching operation by parallelizing the process on a multi-core CPU.
- *Specification-based network IDSs*, also named stateful protocol analysis network IDSs, operate by knowing and tracing the involved protocol states with the objective to detect unexpected sequences of commands [59]. The related disadvantages are the computational load that needs for the protocol state tracing and analysis and their inability to inspect attacks based on the same behavior of the normal network activities. An example of this category of IDSs can be found in [60], where the protocol packets are examined to detect malicious payloads.
- *Hybrid-based network IDSs* operate by mixing the aforementioned approaches with the goal to improve the detection performance, according to a specific objective/scenario [61]. An example is the work proposed in [62], where the authors formalized a hybrid of anomaly-based and specification-based IDS aimed to operate in the context of an IoT environment.

On the basis of the researchers' experience reported in the literature, the main pros and cons of the aforementioned approaches are summarized in Table 1.

Table 1. Pros and cons of IDS operative approaches.

Approach	**Pros**	**Cons**
Anomaly-based	Capability to detect zero-days attacks; effectiveness to detect misuse of privileges; not completely dependent from the operating system	Low accuracy given by the high dynamicity of the network events; latency that does not allow the system to operate in real-time
Signature-based	Able to detect known attacks; provide a detailed analysis pf the attacks	Inability to detect unknown attacks and variations of the known ones; inability to analyze the protocols states; difficulty to keep updated the signatures/patterns database; high computational cost

Table 1. *Cont.*

Approach	Pros	Cons
Specification-based	Able to investigate and trace on the protocol states; capability to detect abuses in the protocol usage	Inability to detect attacks similar to normal network activities; high cost in terms of resources consumed to inspect and trace the protocols state; it can not work by using devoted operating systems or devices
Hybrid-based	It depends on the combination and implementation of the approaches described above	It depends on the combination and implementation of the approaches described above

3.1.2. Operative Placement

A further classification of the IDSs has been made by taking into account their placement in the network to be protected. In this case, the literature classifies the IDSs into the following four categories:

- *Host-based* [63]: A Host-based Intrusion Detection System (HIDS) operates by exploiting several hosts (Figure 2) with the aim to capture the network activity. Each new network event is compared to the information stored in a database (signatures), and, when an intrusion is detected, the system responds with a series of countermeasures. The main advantage of such a configuration is related to the system scalability in terms of possibility to add other hosts to improve the detection performance. The latency between an intrusion and its detection represents the main disadvantages, as well as the high number of misidentifications of normal (false positive) and intrusions (false negative) events.

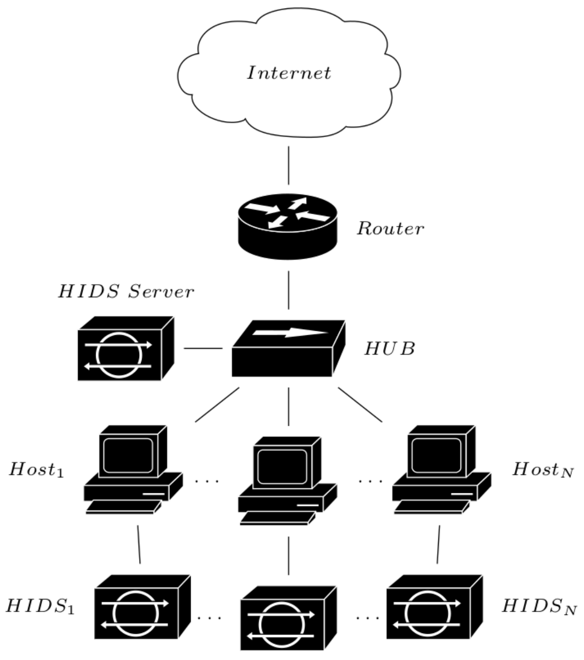

Figure 2. IDS host-based modality.

- *Network-based* [64]: A Network-based Intrusion Detection System (NIDS) operates by capturing and analyzing the whole network activities (Figure 3). Each activity is then compared to the signatures stored in a database and related to known events (normal and intrusion), and only the unknown signatures will be analyzed. This hybrid approach (signature-based and analysis-based) allows us to deal both with known and unknown intrusions (e.g., zero-days attacks). The main disadvantage in this approach is instead related to its inability to operate proactively, together with that to operate with encrypted data.

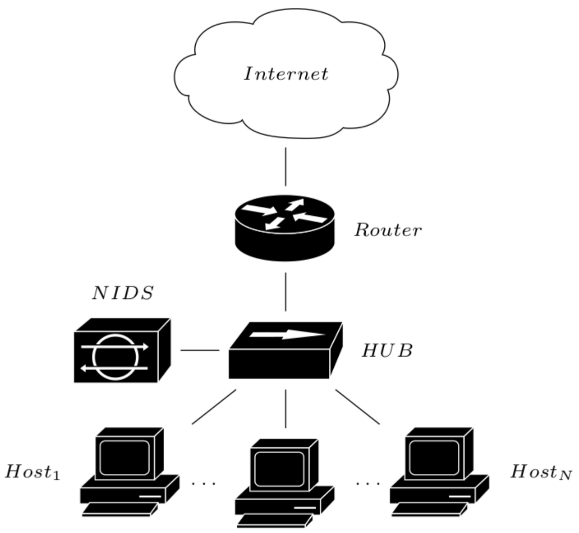

Figure 3. IDS network-based modality.

- *Network-node-based* [65]: A Network-Node-based Intrusion Detection System (NNIDS) operates by positioning the system in a strategic location of the network (Figure 4), and its operative way is a combination of the HIDS and NIDS strategies; thus, it shares with them the same advantages and disadvantages.

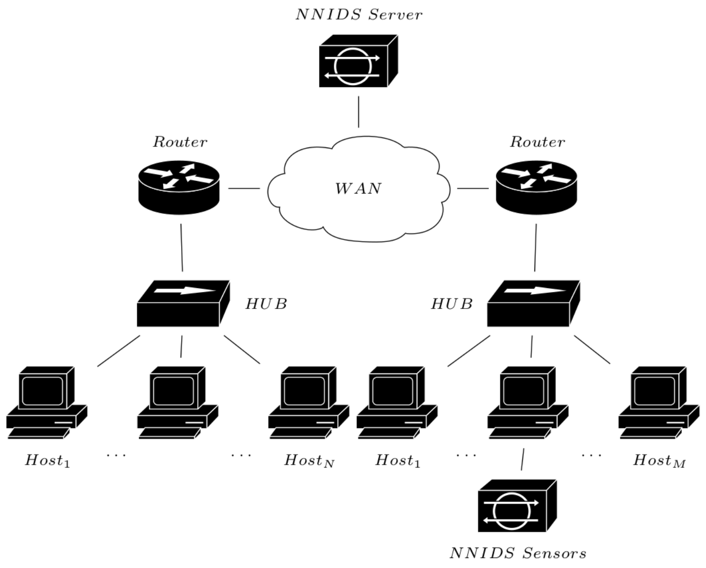

Figure 4. IDS network-node-based modality.

- *Distributed-based* [66]: The Distributed-based intrusion detection systems represent hybrid solutions aimed to combine the aforementioned strategies in order to improve the performance in a specific network context.

3.1.3. Network Attacks

To place the appropriate countermeasures against the potential network attacks, the researchers need detailed knowledge of techniques, strategies, and tools used by the attackers. Such information is easily available, as it is usually in the public domain. On the basis of their behavior and potential impact on the attacked resources, Table 2 groups the most common ones into several classes.

Table 2. Common classes of network attacks.

Attack	Description	References
Denial of Service	The Denial of Service (DoS) and Distributed Denial of Service(DDoS) represent a class of attacks aimed to reduce/block the legitimate use of a service by saturating its resources through a huge number of requests that exceed the response capacity of these	[67,68]
Packet Forging	They are tools focused on the exploitation of the network protocols by manipulating/forging them, for instance in order to spoof the IP source address or other crucial elements	[69,70]
Trojan Horses	Trojan horses are malicious software designed to jeopardize/break network resources, such as a single machine or an entire network. They are hidden inside a legitimate software and, once executed, perform their activity	[71,72]
Fingerprinting	The fingerprinting attacks are aimed to discover as many as possible information about a network protocol implementation and/or network resource by observing its behavior. Information such as parameters, vendor, operating system, and version is then used later to define targeted attacks	[73,74]
Buffer Overflow	This class of attacks exploits a type of bug defined buffer overflow (also defined buffer overrun), where a targeted program write more data to a buffer than it can hold, due to the absence of a buffer bounds checking. Such an operation allows the attacker to execute arbitrary code on the target system	[75,76]

The aforementioned classes of attacks are exploited in order to conduct dangerous well-known attacks such as SYN flooding, IP spoofing, session hijacking, man-in-the-middle, DHCP starvation, and many more [77]. It should be underlined that their intrinsic dangerousness arises from the fact that many of these attacks do not require any in-depth expertise, since they can be conducted using tools easily available on the Internet [78]. By exploiting such tools, people can perform a number of dangerous activities without particular efforts, from those related to the information discovering to those aimed to the exploitation/block of network resources. In addition, the possibility to operate these attacks both by using wired and wireless networks [79], makes it even more difficult to identify those responsible for the attacks.

It can also be observed that many common instruments/tools, which are not precisely classified as attacks, are able to perform crucial operations in an attack context. Some significant examples are simple commands such as ping, fping, arping, and traceroute and more sophisticated tools such as EtherApe [80], able to perform network traffic sniffing [69]. An interesting investigation about the deep-learning techniques used in order to perform automate security tasks such as the malware analysis was presented by Singla and Bertino [81].

3.2. Performance Evaluation

The performance evaluation in the context of the IDSs must take into account several aspects, mainly for the same reason that characterizes other domains, the presence on a high degree of data imbalance [82,83]. Such a problem is related to the fact that most of the involved network events belong to one majority class (normal activities). Although the literature reports numerous performance evaluation metrics in this domain [84], the common choice of most researchers is to combine more than one of these metrics [85], in order to obtain an evaluation as correct/reliable as possible. Considering that the IDS task is usually treated as a classification problem, the evaluation metrics taken into account have to assess the performance in terms of a binary classification in two classes, normal and intrusion [86]. Some categories of metrics largely used in the IDS context are introduced in the following:

- *Confusion-matrix based*: The IDS evaluation is performed by using the confusion matrix values. This is a 2 × 2 matrix that contains True Negatives (TN), False Negatives (FN), True Positives (TP), and False Positive (FP), as shown in Table 3, where such a matrix is contextualized in the intrusion detection context. Some of the metrics (largely used in the intrusion detection domain) derived from that matrix are the Matthews correlation coefficient, F-score, sensitivity, and specificity [84].

Table 3. Confusion matrix.

		Predicted	
		Normal	*Intrusion*
Actual	*Normal*	**TP**	**FN**
	Intrusion	**FP**	**TN**

- *ROC-Curve based*: The IDS evaluation is performed on the basis of the Receiver Operating Characteristic (ROC) curve, where one of the most used derived metrics is the Area Under the Receiver Operating Characteristic (AUC), which is able to assess the capability of an IDS to discriminate the normal events form the intrusion ones, regardless the level of data imbalance, correctly [87].
- *Additional Metrics*: Other metrics are instead adopted when the problem is expressed in terms of regression instead of classification, such as the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) [88]. Different metrics are also aimed to evaluate secondary aspect of an IDS, taking into account, for instance, its performance in terms of computational load, memory usage, etc. [89,90].

4. Formal Notation and Problem Formulation

We denote as E the set of classified network events, which is composed by the subset $E^+ \subseteq E$ of the events classified as *normal* and the subset $E^- \subseteq I$ of those classified as *intrusion*, while we define as $\hat{E}$ the set that contains the network events to classify. Each event is characterized by a series F of features, and it can belong to only one of the two classes in set C. Such an aforementioned formalization is recapped and detailed in Table 4.

Table 4. Formal notation.

Notation	**Description**	**Note**
$E = \{e_1, e_2, \ldots, e_X\}$	*Set of classified events*	
$E^+ = \{e_1^+, e_2^+, \ldots, e_Y^+\}$	*Subset of normal events*	$E^+ \subseteq E$
$E^- = \{e_1^-, e_2^-, \ldots, e_W^-\}$	*Subset of intrusion events*	$E^- \subseteq E$
$\hat{E} = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_Z\}$	*Set of unclassified events*	
$F = \{f_1, f_2, \ldots, f_N\}$	*Set of event features*	
$C = \{normal, intrusion\}$	*Set of event classifications*	

Problem Formulation

On the basis of the previous formal notation, by denoting as λ the classification process performed by the proposed LFE strategy and as $Evaluate(\hat{e}, \lambda)$ an evaluation function that returns 1 in case of correct classification or 0 otherwise, we can formalize our goal in terms of maximization of the sum Ω of its results, as shown in Equation (1):

$$\max_{0 \leq \Omega \leq |\hat{E}|} \Omega = \sum_{z=1}^{|\hat{E}|} \text{Evaluate}(\hat{e}_z, \lambda) \tag{1}$$

In agreement with most of the literature in this area, we consider the intrusion detection in terms of a binary classification task, where the two mutually exclusive classes of destination for each analyzed event are *normal* or *intrusion*, according to the set of event classifications previously formalized in Table 4.

5. Proposed Strategy

The Local Feature Engineering (LFE) strategy proposed in this work introduces a series of new features in the set of features F that characterizes each network event. The addition of new features is then followed by a discretization process [91] applied on them, which is aimed to reduce the number of potential event patterns, since the result is the aggregation of the similar ones.

Since the computational complexity related to the definition of a classification model in the intrusion detection domain [92] is proportional both to its feature space (in terms of number of features) and its data (in terms of number of events), such a pattern reduction has a positive effect to the computational load.

The high-level architecture of the proposed LFE strategy is shown in Figure 5. It indicates how such a strategy is based on two different data paths: the first one is related to the preprocessing of set E (classified events) and set $\hat{E}$ (unclassified events) through the LFE strategy (i.e., feature addition and values discretization), whereas the second one is aimed to detect the optimal δ value to use for the previously mentioned discretization process. This process is denoted as $\circlearrowright$, and it is performed by using only E. As last step, the evaluation model is trained by using set E, previously preprocessed according to the δ value we defined. The unevaluated events in set $\hat{E}$, whose features have also been preprocessed by using the same δ value, are then classified on the basis of the defined evaluation model. All the implementation details are reported in Sections 5.1 and 5.2.

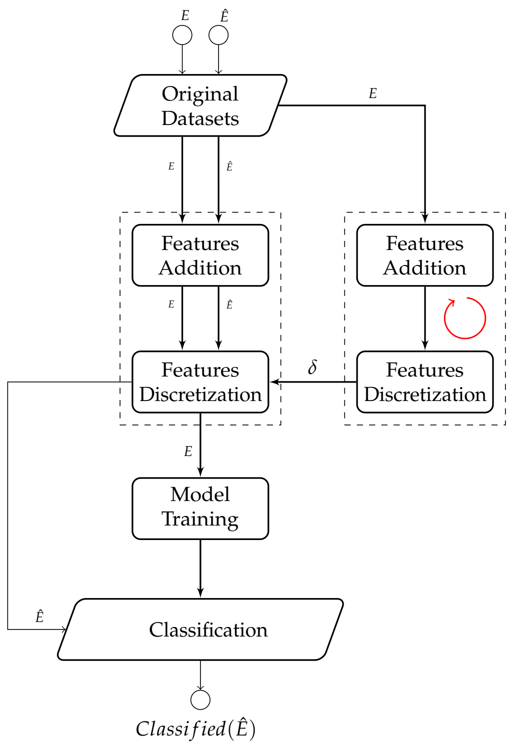

Figure 5. LFE high-level architecture.

5.1. Feature Engineering

The addition of new features that characterizes the proposed LFE strategy falls into the literature background as a *feature engineering process*. Feature engineering is defined as a process aimed to improve the performance of a predictive model by exploiting the transformation of the original feature space [36]. An example can be found in [93], where the authors proposed a technique for automating feature engineering in classification task.

The process adopted in our approach is very close to this operative way, but, instead of transforming raw data into features, with the goal to better represent the data in the definition of a predictive model, it exploits the original features to defines the additional ones. The idea behind this choice is that there are normal and intrusion patterns that are very similar, and the addition of discriminant elements (i.e., the new features) is able to better differentiate them, allowing the prediction model to perform more reliable classifications. The new features are calculated on the basis of the existing ones and their aim is a better characterization of the event in the two considered classes (i.e., normal and intrusion).

The new features are calculated on the basis of some metrics (that are described below), with the result formalized in Equation (2), where $\{e_1, e_2, \dots, e_X\}$ denotes the set of original event features, $\{m_1, m_2, \dots, m_Y\}$ denotes the set of the new ones, and the new feature vector $\hat{F}$ is given by the concatenation of these two sets.

$$\hat{F} = \boxed{f_1}\ \boxed{f_2}\ \cdots\ \boxed{f_N} \frown \boxed{m_{N+1}}\ \boxed{m_{N+2}}\ \cdots\ \boxed{m_{N+O}} \tag{2}$$

This process is applied on each instance e and $\hat{e}$, which are characterized by the features in set F, respectively, related to the set of the evaluated instances in set E and the unevaluated ones in set $\hat{E}$. In more detail, denoting as $\mu = \{m_1, m_2, m_3, m_4\}$ the new features to add to set F, they are defined using the following calculation criteria: m_1 = *Minimum*, m_2 = *Maximum*, m_3 = *Average*, m_4 = *Standard Deviation*, as formalized in Equation (3).

$$\mu = \begin{cases} m_1 = min(f_1, f_2, \dots, f_N) \\ m_2 = max(f_1, f_2, \dots, f_N) \\ m_3 = \dfrac{1}{N} \sum_{n=1}^{N} (f_n) \\ m_4 = \sqrt{\frac{1}{N-1} \sum_{n=1}^{N} (f_n - \bar{f})^2} \end{cases} \tag{3}$$

5.2. Feature Discretization

The feature discretization process is instead a process that the literature describes as the conversion of the feature values into a categorical one [94], for instance in order to allow us the use of classification algorithms unable to operate with continuous feature values [95], as well as to reach a better data understandability. The loss of information, which represents a disadvantage of the discretization process, is balanced by the proposed strategy through the addition of new features previously described.

As previously emphasized in Section 2, the feature discretization process should be considered as a synergistic component of the proposed LFE strategy, since it reduces the potential event patterns, while the introduction of new features (the other LFE component) counteracts the loss of information, which represents a side effect of a discretization process.

In more detail, this process divides the value of each feature into a discrete number of non-overlapped intervals, mapping the original numerical value (continuous or discrete) into one of the intervals of a defined range. Such a process is shown in Figure 6, which exemplifies it by considering a conversion of feature values from its original range $[0, 100]$ into a discrete range $\{0, 1, \dots, 60\}$. The resulted values $\{12, 30, 7, 54, 42, 60\}$ then represent the discretization of the source values $\{10.60, 28.78, 5.20, 52.11, 41.00, 92.11\}$.

The feature engineering described above extends the original feature space F into the new feature space $\hat{F}$ (by adding the four calculated values in set μ). Then, this feature space (i.e., $\{f_1, f_2, \dots, f_N, m_1, m_2, m_3, m_4\}$) is further processed by discretizing all the feature values (continuous or discrete) according to a defined range $\{d_1, d_2, \dots, d_\delta\}$, experimentally defined.

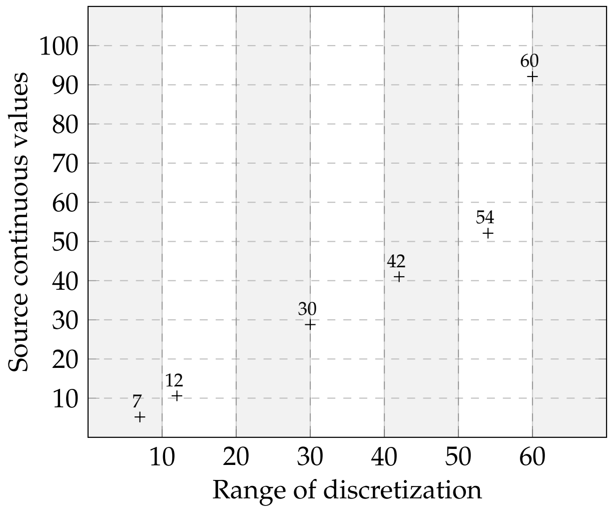

Figure 6. Data discretization process.

Denoting as $f \overset{\delta}{\rightarrow} d$ the function of discretization, this process is formalized in Equation (4), where for simplicity we denote (from now on) the resulted set again as F, according to the formal notation provided in Section 4. It should be noted that this process is performed for each event $e \in E$ and for each event $\hat{e} \in \hat{E}$, respectively, the training set and the set to evaluate.

$$\begin{aligned} &\hat{F} = \{f_1, f_2, \ldots, f_N, f_{N+1}, f_{N+2}, f_{N+3}, f_{N+4}\} \\ &\downarrow \delta \\ &F = \{d_1, d_2, \ldots, d_N, d_{N+1}, d_{N+2}, d_{N+3}, d_{N+4}\} \end{aligned} \tag{4}$$

5.3. Data Model

The feature engineering and discretization processes generate a new data model, which is used during both training and classification operations. Its formalization is shown in Equation (5), where for simplification reasons we refer only to set E. (i.e., the set that contains the evaluated events used for the evaluation model training). This means that the same formalization is also valid for set $\hat{E}$ (i.e., the set that contains the unevaluated events to classify).

$$E = \begin{array}{|c|c|c|c|c|c|c|c|} \hline d_{1,1} & d_{1,2} & \cdots & d_{1,N} & m_{1,N+1} & m_{1,N+2} & m_{1,N+3} & m_{1,N+4} \\ \hline d_{2,1} & d_{2,2} & \cdots & d_{2,N} & m_{2,N+1} & m_{2,N+2} & m_{2,N+3} & m_{2,N+4} \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline d_{X,1} & d_{X,2} & \cdots & d_{X,N} & m_{X,N+1} & m_{X,N+2} & m_{X,N+3} & m_{X,N+4} \\ \hline \end{array} \tag{5}$$

5.4. Event Classification

Algorithm 1 here formalized has been designed to exploit the new data model, defined on the basis of the proposed LFE strategy.

In more detail, the algorithmic formalization of Steps 2 and 3 of Algorithm 1, which refers to the application of the LFE strategy, is shown in Algorithm 2.

Algorithm 1 takes as input a classifier clf, the set of classified events E, the set of events to classify $\hat{E}$, and a value δ that defines the range of discretization, which is defined experimentally. It returns the classification of all events in set $\hat{E}$. The data transformation is performed at Steps 2 and 3, as detailed in Algorithm 2, where the addition of the four new features (Step 4), calculated as described in Section 5.1, is followed by the discretization of the feature values (Step 5), and the new data model is returned at Step 8.

Algorithm 1 LFE events classification

```
Input: $clf$=Classifier, $E$=Classified events, $\hat{E}$=Events to classify, $\delta$=Range of discretization
Output: $out$=Classification of events in $\hat{E}$
 1: procedure EventsClassification($clf$, $E$, $\hat{E}$)
 2:     $E'' \leftarrow DataTransformation(E, \delta)$             ▷ LFE data transformation
 3:     $\hat{E}'' \leftarrow DataTransformation(\hat{E}, \delta)$     ▷ LFE data transformation
 4:     $model \leftarrow ClassifierTraining(clf, E'')$            ▷ Evaluation model training
 5:     for each $\hat{e}'' \in \hat{E}''$ do                       ▷ Processes all the events in $\hat{E}$
 6:         $c \leftarrow getEventClass(model, \hat{e}'')$          ▷ Event classification
 7:         $out.add(c)$                                          ▷ Stores classified events
 8:     end for
 9:     return $out$                                              ▷ Returns classification of event in $\hat{E}$
10: end procedure
```

Algorithm 2 LFE data transformation

```
Input: $DB1t$=Set of data to transform, $\delta$=Range of discretization
Output: $DB2$=Transformed set of data in agreement with the LFE strategy
 1: procedure DataTransformation($DB1$, $\delta$)
 2:     for each $event \in DB1$ do
 3:         $F \leftarrow getFeatures(event)$                          ▷ Get event features
 4:         $F.add(Features(m_1, m_2, m_3, m_4)$                       ▷ Add new features
 5:         F = $discretizeData(F)$                                    ▷ Discretize feature values
 6:         $DB2.add(F)$                                               ▷ Add processed event features
 7:     end for
 8:     return $DB2$                                                   ▷ Returns transformed dataset
 9: end procedure
```

The classifier clf model training is then performed at Step 4 of Algorithm 1 by using the new data model, whereas the classification of each event in set $\hat{E}$ is performed at Steps 5–8, and the results returned at Step 9 of the same algorithm.

Computational Complexity

Here, we evaluate the computational complexity related to the proposed LFE strategy, since this provides important information in the context of real-world applications. Such an evaluation was made by analyzing the theoretical complexity of the events classification (Algorithm 1) and the events features transformation (Algorithm 2), as previously formalized. The complexity is expressed according to the *Big O* notation [96], which is aimed to define the upper bound of an algorithm, since it bounds a function only from above.

As a premise, considering that our experiments involve different classification algorithms, and that the proposed strategy has been designed to operate regardless of the adopted algorithm, we do not take into account in this evaluation the classification algorithm complexity [97].

Generalizing as n the total number of events to process, Table 5 reports the significant steps of Algorithm 1, along with the involved single components complexity, and the final *Big O* complexity that derives from them.

Table 5. LFE computational complexity.

Step	Components Complexity	Global Complexity
Step 2	$O(n) \times O(1) \times O(1) \times O(1)$	$O(n)$
Step 3	$O(n) \times O(1) \times O(1) \times O(1)$	$O(n)$
Steps 5−8	$O(n) \times O(1)$	$O(n)$

By carrying out a more detailed analysis of the operations involved in Algorithm 1 components, according to the complexity analysis presented in Table 5, we can observe the following.

- Steps 2 and 3 of Algorithm 1 are related to the operation performed by Algorithm 2, where, in the main loop, the original feature F are extracted from the processed event in a constant time $O(1)$, as well as the addition of the four new features m_1, m_2, m_3, m_4, and the storing of the extended feature space. The dominant complexity $O(n)$ related to the main loop represents the the global complexity of all the involved operations.
- Steps 5–8 of Algorithm 1 are related to the loop where the event classification is performed by exploiting the evaluation model trained in Step 4. The dominant complexity of this loop is $O(n)$, since the storing of the classifications take a constant time $O(1)$.
- The complexity related to the adopted algorithm of classification, training and classification time, is not taken into consideration for the reasons discussed above. This trivially means that the final complexity of the proposed LFE strategy is defined by considering the upper bound $O(n)$ complexity that characterizes the proposed strategy and the complexity of the adopted algorithm.
- In any case, it should be observed that the complexity can be further reduced by parallelizing the process on several machines, for instance using large-scale distributed computing models, such as *MapReduce* [98–100].

6. Strategy Validation

The environment used to validate the proposed LFE strategy was based on the following hardware and software: a single machine quad-core Intel i7-4510U, 2.00 GHz CPUs, 12 GB of RAM; a Linux-based operating system (kernel 3.16.0-amd64); and Python programming language with the `Scikit-learn` (http://scikit-learn.org) library.

Note that, to grant the experiments reproducibility, the seed of the pseudo-random number generator in `Scikit-learn` was fixed to 1.

6.1. Dataset

The main real-world dataset we used in the validation process is the NSL-KDD dataset (https://github.com/defcom17/NSL_KDD). It essentially represents an improved version of the KDD-CUP99 dataset widely used in the past, which suffered from some issues [101]. In more detail, the main improvements of the NSL-KDD dataset, with regard to the original KDD version, are the following:

- The redundant records in the training set have been removed to avoid that a classifier can be biased by the frequency of them.
- The duplicate records in the test sets have been removed to avoid that a classifier performance can be biased when the better detection rate of it is related to these records.
- The new number of records in the training and test sets allows a classifier to perform the validation process without the need to select small portions of them, randomly.

The NSL-KDD dataset was selected since it is considered as a benchmark in the intrusion detection research area, allowing the researchers to compare their solutions performance with an ever-increasing number of works in this field, unlike other recent datasets. Moreover, we chose this dataset for the network scenario it provides, as we aimed to assess our performance in the context of canonical networks. The NSL-KDD dataset involves events related to the UDP, TCP, and ICMP protocols, allowing to validate approaches and strategies in a real-world network scenario, in terms of both number of events and involved protocols and attacks.

We adopted this dataset to perform an independent experimental evaluation of the proposed strategy by intrusion typology, separately. On the other hand, in Section 7.1, we take into account two additional datasets, with the aim to evaluate such a strategy in heterogeneous scenarios (i.e., which include all types of attacks).

The numerical relevance of the NSL-KDD dataset in terms of normal and intrusion events (respectively, E_+ and E_-) is reported in Table 6. Note that the original dataset consists of two different files, i.e. `KDDTrain+` and `KDDTest+`, respectively, related to the original training and test sets, as defined by the dataset authors. However, since some normal/intrusion unique events can be found in one file only, we discarded this original separation, and joined the two files as a unique full dataset for the rest of this section, thus only considering the global values IN Table 6 (here, the original distinction between `KDDTrain+` and `KDDTest+` is reported for information purposes).

Table 6. Numerical relevance of the original dataset events.

Dataset	Total Events $\lvert E\rvert + \lvert\hat{E}\rvert$	Normal $\lvert E_+\rvert$	Intrusion $\lvert E_-\rvert$	Features $\lvert F\rvert$	Unique Events
KDDTrain+	125,973	67,343	58,630	41	23
KDDTest+	22,543	9,710	12,833	41	38
Global	148,516	77,053	71,463		

The distribution of the network events is provided in Table 7, while their classification in the five classes (i.e., PEA, DSA, RSA, RAA, and NNA) is detailed in Table 8. Finally, Table 9 presents a numerical overview of the aforementioned classes of dataset events.

Table 7. Distribution of dataset events.

	Network Event	Occurrences	Class		Network Event	Occurrences	Class
01	*apache2*	737	*DSA*	21	*processtable*	685	*DSA*
02	*back*	1315	*DSA*	22	*ps*	15	*PEA*
03	*buffer_overflow*	50	*PEA*	23	*rootkit*	23	*PEA*
04	*ftp_write*	11	*RAA*	24	*saint*	319	*RSA*
05	*guess_passwd*	1283	*RAA*	25	*satan*	4368	*RSA*
06	*httptunnel*	133	*RAA*	26	*sendmail*	14	*RAA*
07	*imap*	12	*RAA*	27	*smurf*	3311	*DSA*
08	*ipsweep*	3740	*RSA*	28	*snmpgetattack*	178	*RAA*
09	*land*	25	*DSA*	29	*snmpguess*	331	*RAA*
10	*loadmodule*	11	*PEA*	30	*sqlattack*	2	*PEA*
11	*mailbomb*	293	*DSA*	31	*spy*	2	*RAA*
12	*mscan*	996	*RSA*	32	*teardrop*	904	*DSA*
13	*multihop*	25	*RAA*	33	*udpstorm*	2	*DSA*
14	*named*	17	*RAA*	34	*warezclient*	890	*RAA*
15	*neptune*	45,871	*DSA*	35	*warezmaster*	964	*RAA*
16	*nmap*	1566	*RSA*	36	*worm*	2	*DSA*
17	*perl*	5	*PEA*	37	*xlock*	9	*RAA*
18	*phf*	6	*RAA*	38	*xsnoop*	4	*RAA*
19	*pod*	242	*DSA*	39	*xterm*	13	*PEA*
20	*portsweep*	3088	*RSA*	40	*normal*	77,053	*NNA*

Table 8. Classes of dataset events.

Class	Description	Target
PEA	*Privilege Escalation Attack*	Gains a privileged access as unprivileged user (e.g., Buffer overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, and Ps).
DSA	*Denial of Service Attack*	Puts out of service a service/system through a number of normal activities (e.g., Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb, Processtable, Udpstorm, Apache2, and Worm).
RSA	*Remote Scanning Attack*	Obtains information on services/systems by using invasive and non-invasive techniques (e.g., Satan, IPsweep, Nmap, Portsweep, Mscan, and Saint).

Table 8. *Cont.*

Class	Description	Target
RAA	*Remote Access Attack*	Gets an access to a remote system by using several techniques (e.g., Guess_password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, and Named).
NNA	*Normal Network Activity*	This is the class we used in order to identify the normal network events).

Table 9. Event categories numerical overview.

	PEA Events	**DSA Events**	**RSA Events**	**RAA Events**	**NNA Events**
Total	119	53,387	14,077	3879	77,053
%	0.08	35.95	9.48	2.61	51.88

6.2. Metrics

We adopted three different metrics, in the validation process, to evaluate the performance of the proposed LFE strategy: (i) the *specificity* [102]; (ii) the Matthews Correlation Coefficient (MCC) [103,104]; and (iii) the Area Under the Receiver Operating Characteristic curve (AUC) [105].

The first two are based on the results related to the confusion matrix described in Section 3.2, whereas the last one is related to the Receiver Operating Characteristic (ROC) curve. The combination of these metrics provides us a reliable evaluation of the strategy effectiveness, regardless to data imbalance (i.e., the numerical difference between normal and intrusion events), according to the literature in this research area.

- The formalization of the specificity metric, which is aimed to assess the performance in terms of the system capability to detect intrusions (i.e., the true negative rate), is shown in Equation (6), according to the formal notation provided in Section 4. Denoting as $\hat{E}$ the unclassified instances, TN reports the number of events classified as intrusion, correctly, and FP reports the number of intrusion events classified as normal, erroneously.

$$Specificity(\hat{E}) = \frac{TN}{(TN + FP)} \tag{6}$$

- The formalization of the MCC metric, which respect to other confusion-matrix metrics is able to operate even with unbalanced data, is shown in Equation (7). It provides a result in the $[-1, +1]$ range, where the correctness of all the performed classification leads to $+1$, whereas the incorrectness of all of them leads to -1, and 0 indicates the typical performance reached by a random classifier.

$$MCC(\hat{E}) = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{7}$$

- The formalization of the AUC metric, which is able to evaluate the capability of a classifier to differentiate the normal from the intrusion events, regardless their numerical imbalance, is shown in Equation (8). Given set E, denoting as κ all the possible score comparisons related to each event e, the final result (in the $[0, 1]$ range) is given by the average of them, where 1 indicates the best classification performance.

$$AUC = \frac{1}{I_+ \cdot I_-} \sum_{1}^{|I_+|} \sum_{1}^{|I_-|} \kappa(i_+, i_-) \;\; \textbf{with} \;\; \kappa(i_+, i_-) = \begin{cases} 1, & if\ i_+ > i_- \\ 0.5, & if\ i_+ = i_- \\ 0, & if\ i_+ < i_- \end{cases} \tag{8}$$

6.3. Algorithms

To validate the performance of the proposed LFE strategy, its effectiveness was evaluated in the context of a series of largely used state-of-the-art classification algorithms, whose performance has been measured before and after the application of the proposed strategy in the datasets. They are Gradient Boosting (GB), Adaptive Boosting (AB), Random Forests (RF), Multilayer Perceptron (MP), and Decision Tree (DT). A brief description of how each algorithm works is shown below. Let us observe that, actually, a fine-tuning of the algorithms is not a crucial constraint, since the main objective of the proposed LFE strategy is the improvement of an algorithm performance, regardless of its optimal configuration. For these reasons, we selected default parameters for the considered algorithms, identical for each single run of the experiment, hence without the necessity of using a validation set. Such a default parameter configuration is reported, for each algorithm, in Table 10.

- *Gradient Boosting*: It is used to face regression and classification tasks. Its evaluation model is based on an ensemble of weak prediction models such as decision trees, which is exploited to produce the final predictions. Its way to operate is based on continuous exploitation of a weak learning method in order to obtain a sequence of hypotheses, which are reused on the difficult cases, with the aim to improve the classification performance. Some significant works in the intrusion detection area that exploit this algorithm can be found in [14,106].
- *Adaptive Boosting*: Similar to gradient boosting, it works by combining multiple weak classifiers in order to obtain a strong one. Its way to operate relies on the consideration that the single classifiers may not reach to perform the classification, accurately, and thus they are grouped, and each of them learns from the other ones, progressively. Some significant works in the intrusion detection area that exploit this algorithm can be found in [15,107].
- *Random Forests*: It is based on many decisions trees. Bagging and feature randomness when building each individual tree are used in order to define an uncorrelated forest of trees, which leads to best performance of classification, with respect to a single tree. Some significant works in the intrusion detection area that exploit this algorithm can be found in [108,109].
- *Multilayer Perceptron*: It belongs to the class of feed-forward Artificial Neural Network (ANN), working using at least three layers (input, hidden, and output), It uses a supervised learning back-propagation technique, exploiting a non-linear activation function (except for the input nodes). Some significant works in the intrusion detection area that exploit this algorithm can be found in [17,110].
- *Decision Tree*: It represents a supervised machine learning approach aimed to induct a decision tree by starting from the training data. Such a decision tree is a model that relates the information about an item to a target classification. Some significant works in the intrusion detection area that exploit this algorithm can be found in [17,111].

Table 10. Algorithms parameters.

Classification Algorithm	**Parameters Values**
Gradient Boosting	criterion='friedman_mse', init=None, learning_rate=0.1, loss='deviance', max_depth=3, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, presort='auto', random_state=1, subsample=1.0, verbose=0, warm_start=False
Adaptive Boosting	algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=50, random_state=1
Random Forests	bootstrap=True, class_weight=None, criterion='gini', max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1, oob_score=False, random_state=1, verbose=0, warm_start=False
Multilayer Perceptron	activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(100,), learning_rate='constant', learning_rate_init=0.001, max_iter=200, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False
Decision Tree	class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=1, splitter='best'

6.4. Modality

Following the LFE strategy high-level architecture presented in Figure 5, we first transformed the classification of the events of the full dataset from categorical into numeric ones (i.e., by using 0 to classify the normal events and 1 to classify the intrusion ones).

Then, since the proposed strategy needs the experimental definition of the δ value to use in the context of the discretization process, we split the dataset into two subsets of same dimension, to avoid data dependency during the experiments:

- The first one, called *in-sample* subset, was used to perform a *k-fold* cross-validation (with $k = 5$), to find the optimal δ value for each category (recall that we used default algorithm parameters; however, a fine-tuning of such parameters can be eventually done at this stage). The optimal values of δ are shown in Table 11.
- The second one, called *out-of-sample* subset, was used to perform multiple runs of the pre-trained algorithms (with the previously chosen δ values), on *unseen* test data, to get the experimental results.

To find the optimal δ for each algorithm–event category pair, we iterated δ over the range $(1, MAX]$, with MAX empirically set to 300. In addition, note that, according to the problem formulation made in Section 4, the intrusion detection task is performed as a binary classification problem, since each event can be classified only as normal or intrusion, as already anticipated in the formal notation provided in Section 4. This follows what has been done by most of the works in the intrusion detection literature. However, the assessment of the performance related to each single class of events, instead of considering them as a whole, allows us to get information about the behavior of the LFE strategy in each attack scenario, separately.

Table 11. NSL-KDD dataset—optimal δ value detection.

Dataset	Algorithm	δ Value	Dataset	Algorithm	δ Value
	AB	12		AB	71
	DT	120		DT	73
DSA	GB	27	RAA	GB	250
	MP	7		MP	247
	RF	6		RF	89
	AB	125		AB	148
	DT	158		DT	187
NNA	GB	135	RSA	GB	221
	MP	112		MP	138
	RF	77		RF	118
	AB	171			
	DT	157			
PEA	GB	47			
	MP	70			
	RF	123			

7. NSL-KDD Results

The experimental results related to the process of comparison between the proposed strategy and the canonical use (baseline) of the state-of-the-art classification algorithms (i.e., without the LFE data preprocessing) in the context of the NSL-KDD dataset, are reported in Table 12. In this first view of the results, the performance is expressed in terms of the average value between all the adopted three metrics of evaluation (i.e., specificity, MCC, and AUC), in a similar way to what has been done in other studies [112], and the better average performance of the LFE strategy is denoted with $+$.

Table 12. NSL-KDD dataset—-LFE strategy performance comparison.

Dataset Category	Classification Algorithm	Baseline Performance	LFE Performance	Strategies Comparison	p-Value
	AB	0.9759	0.9774	+	<0.01
	DT	0.9762	0.9811	+	<0.01
DSA	GB	0.9787	0.9796	+	<0.01
	MP	0.9274	0.9719	+	<0.01
	RF	0.9767	0.9762	-	0.04
	AB	0.9225	0.9226	+	<0.01
	DT	0.9542	0.9535	-	<0.01
NNA	GB	0.9424	0.9453	+	<0.01
	MP	0.8326	0.9317	+	<0.01
	RF	0.9580	0.9523	-	<0.01
	AB	0.6314	0.6412	+	<0.01
	DT	0.5385	0.6369	+	<0.01
PEA	GB	0.6317	0.6512	+	<0.01
	MP	0.2703	0.5232	+	<0.01
	RF	0.5399	0.6222	+	0.07
	AB	0.8615	0.8758	+	<0.01
	DT	0.8841	0.8655	-	0.18
RAA	GB	0.8820	0.8855	+	<0.01
	MP	0.6005	0.8534	+	<0.01
	RF	0.8802	0.8681	-	1.00
	AB	0.9515	0.9324	-	<0.01
	DT	0.9303	0.9436	+	<0.01
RSA	GB	0.9404	0.9444	+	<0.01
	MP	0.8498	0.9219	+	<0.01
	RF	0.9349	0.9425	+	0.20

The statistical significance of the results was assessed through a *five by two-fold cross-validation paired t-test* [113]. Thus, we calculated the *t-values* for each dataset-algorithm pair, by considering a two-tailed t distribution with five degrees of freedom. We then considered as significant, with a 0.05 significance level, each strategy comparison (between our LFE approach and the baseline one) where $t > 2.57$. We then computed and reported the resulting *p-values*, as shown in Table 12.

These results led us to reject the null hypothesis for almost every considered pair, with the exception of seven cases out of 35, four out of 25 in the NSL-KDD dataset (Table 12) and three out of 10 in the CIC and UNS datasets, which are used and described in Section 7.1, as shown in Table 13, where the difference between us and the baseline is not statistically relevant.

Table 13. LFE strategy performance comparison—additional datasets.

Dataset Category	Classification Algorithm	Baseline Performance	LFE Performance	Strategies Comparison	*p*-Value
CIC	AB	0.9741	0.9539	−	<0.01
	DT	0.9631	0.9636	+	0.06
	GB	0.9799	0.9736	−	<0.01
	MP	0.7653	0.9153	+	<0.01
	RF	0.9783	0.9788	+	0.03
UNS	AB	0.9354	0.9317	−	<0.01
	DT	0.9173	0.9206	+	0.56
	GB	0.9333	0.9225	−	<0.01
	MP	0.5504	0.7558	+	<0.01
	RF	0.9214	0.9310	+	0.15

Furthermore, the table also reports the mean performance related to each of the evaluation metrics, measured in the out-of-sample subset with regard to each classification algorithm, whereas the results related to each metric are reported in Figures 7–9.

Finally, Figures 10 and 11 show the overall mean performance in relation to all the classification algorithms, and in related to all the involved classes of events are shown, respectively.

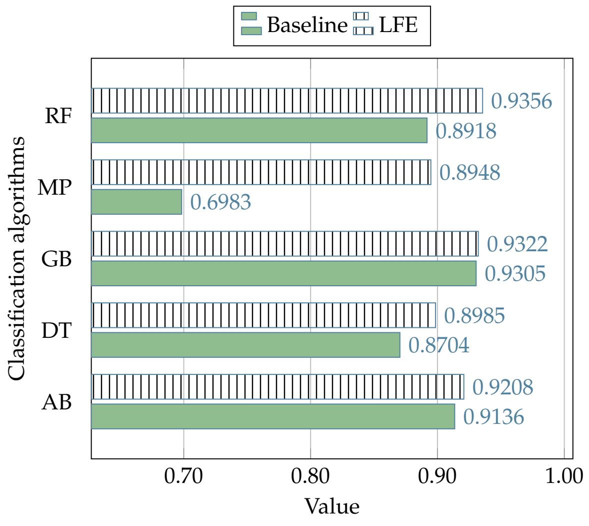

Figure 7. NSL-KDD dataset—specificity mean performance.

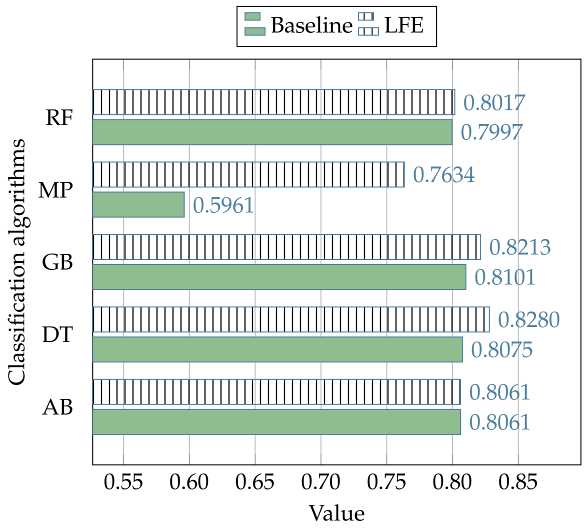

Figure 8. NSL-KDD dataset—MCC mean performance.

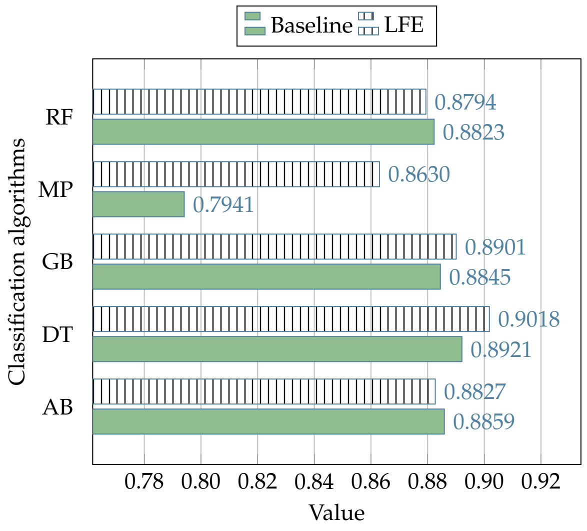

Figure 9. NSL-KDD dataset—AUC mean performance.

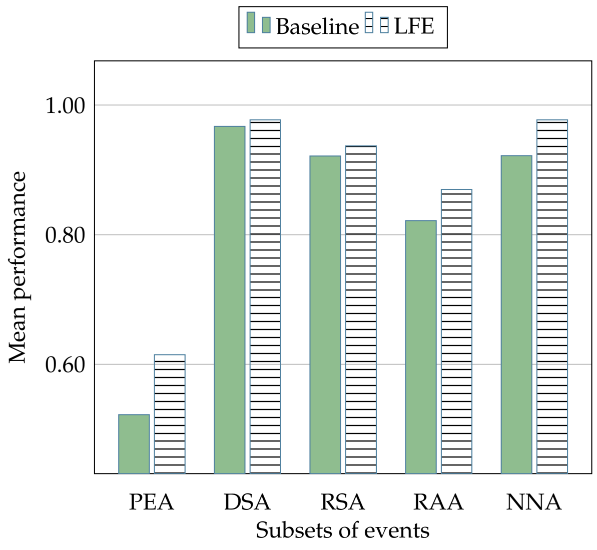

Figure 10. NSL-KDD dataset—overall mean performance in relation to all algorithms.

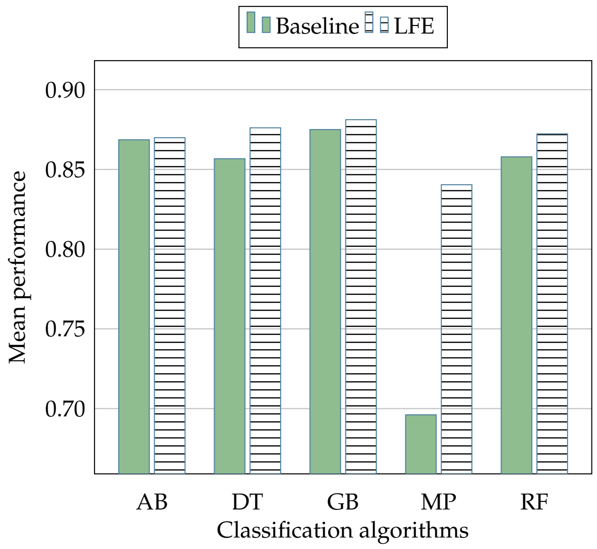

Figure 11. NSL-KDD dataset—overall mean performance in relation to all events.

7.1. Heterogeneous Network Scenarios

The previous validation process of the proposed LFE strategy, performed in relation to each category of events in the NSL-KDD dataset, is here extended with further experiments, which were carried out on other two datasets that involve new classes of devices and events. Differently to what we have done for the NSL-KDD dataset, where the analysis was focused on the goal to perform an in-depth evaluation of the LFE strategy behavior with regard to specific network events, here we consider these datasets in their entirety, with the aim to simulate a real-world exploitation of the LFE strategy, in order to offer an overview in terms of average performance, with respect to heterogeneous network scenarios.

The additional datasets that we used to this purpose are publicly available, and are the following:

- **CICIDS2017** [114]: This dataset (hereafter, CIC) involves network activity based on the HTTP, HTTPS, FTP, SSH, and email protocols, related to normal events and several intrusion attacks such as Brute Force SSH, Brute Force FTP, Denial of Service, Infiltration, Heartbleed, Web Attack, Botnet, and Distributed Denial of Service. It is freely downloadable from the Internet (https://www.unb.ca/cic/datasets/ids-2017.html).
- **UNSW-NB15** [115]: This dataset (hereafter, UNS) was created in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) (https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/). It contains nine different types of attacks: Fuzzers, Analysis, Backdoors, Denial of Service, Exploits, Reconnaissance, Shellcode, Worms, and Generic.

The optimal values of δ, shown in Table 14, have been detected by following the same criteria reported in Section 6.4, as well as the adopted metrics and algorithms, which are the same as those previously described, respectively, in Sections 6.2 and 6.3.

The experimental results related to these datasets are shown in Table 13.

Table 14. Optimal δ value detection—additional datasets.

Dataset	**Algorithm**	**δ Value**	**Dataset**	**Algorithm**	**δ Value**
	AB	256		AB	259
	DT	100		DT	239
CIC	GB	150	UNS	GB	246
	MP	217		MP	12
	RF	273		RF	239

7.2. Discussion

The experimental results reported in Tables 12 and 13, together with those shown in Figures 7–11, lead to the following considerations:

- The average performance between all the adopted metrics shows how the proposed LFE strategy is able to outperform the canonical use of the classification algorithms, which do not involve the LFE data preprocessing, as demonstrated by the results presented in Tables 12 and 13, where, respectively, the LFE strategy outperforms the canonical solutions in 19 cases out of 25, and 6 cases out of 10.
- Such improvements are also evident by observing Figures 7–9, where the performances related to all the subsets of events in the NSL-KDD dataset are evaluated with regard to each algorithm of classification.
- Another important aspect of the experimental results is related to the LFE capability to better operate in the context of different scenarios, since the out-of-sample subset of the NSL-KDD dataset involves both different type of events and different data configurations, according to the dataset characteristics reported in Table 9.
- Such a capability also emerges from the results of the experiments presented in Section 7.1, where the performance of the proposed LFE strategy was evaluated on additional datasets characterized by heterogeneous events.
- It should be noted that all the used metrics of evaluation (i.e., specificity, MCC, and AUC) are focused on the assessment of the LFE capability to detect, correctly, the intrusion events, and that their combination, together with the adoption of a k-fold cross-validation criterion, offers a reliable evaluation not significantly biased by problems such as data unbalance (MCC and AUC evaluations) or overfitting (cross-validation).
- The aforementioned considerations are supported by the fact that, in addition to the specificity, we evaluated the results in terms of MCC and AUC metrics, which are able to assess the ability to discriminate the two possible final classifications of the events, normal and intrusion.
- Another interesting aspect of the experimental results is related to the capability of the LFE strategy to improve the canonical classification performance, regardless the implemented algorithm, also by considering that such a strategy has been adopted, by using the same algorithm parameters used by the canonical classification solutions;
- As far as the cases where the increase in performance appears less significant are concerned, it should be considered that a typical activity of an IDS involves a huge number of classification; thus, in this case as well, the improvement can be considered a very important milestone.
- The LFE strategy has proven capable of improving the performance of the canonical algorithms largely used in this domain, showing its effectiveness in a number of different real-world scenarios, since it has been evaluated with a different number of events, as well as with different type of events, and different distribution of normal and intrusion events.
- The information shown in Figure 10, which reports the overall mean performance in relation to all the algorithms, offers us the measure of how much the proposed strategy is able to improve the average performance in the context of the most used algorithms in the intrusion detection literature, showing the benefits of applying the LFE strategy.
- In a similar manner, as shown in Figure 11, it provides information about the overall mean performance related to each algorithm used in all the different subset of events (i.e., PEA, DSA, RSA, RAA, and NNA), indicating a general improvement that in some cases (e.g., the MP algorithm) is really significant.
- The experiments we performed in detail for each category of events (NSL-KDD dataset), along with the additional ones we performed on several datasets characterized by heterogeneous network events (CIC and UNS datasets), proved again the effectiveness of the proposed LFE

strategy, as in most cases it has been able to improve the performance of the state-of-the-art algorithms taken into consideration.

- In light of the previous considerations based on the experimental results, we can conclude that the validation process demonstrates that the proposed LFE strategy can be profitably used in order to improve the classification performance of the canonical algorithms largely used in the intrusion detection literature, suggesting a possible exploitation also in those solutions that involve more than a single algorithm, e.g. those based on hybrid approaches or ensemble ones.

8. Conclusions and Future Directions

Today's scenario of modern societies, heavily dependent on the use of an ever-increasing number of network-based services, private and public, generates a huge number of network events related to these activities, which security systems, such as the IDSs, should process as quickly as possible. Unfortunately, the requested short response time in terms of events classification should go hand in hand with its reliability, since a wrong events classification leads toward a series of problems, both case of both false positives and false negatives (i.e., intrusions identified as normal events and vice versa).

For these reasons, the research activity in this area involves a large number of people and capital, with the aim to develop effective techniques and strategies able to face the problem. The challenges are mainly related to some open problems that affect this field, such as a data domain characterized by a high degree of data imbalance and the heterogeneity/dynamism of the involved network events.

The idea behind the LFE strategy proposed in this work revolves around the observation that an improvement in terms of event characterization given by the introduction of several new features, locally calculated for each network event on the basis of the original ones, combined with a process of discretization of all feature values, can lead to an improvement of the performance related to the canonical algorithm of classification largely adopted in this field. This kind of feature engineering process was validated by using real-world data, and the experimental results indicate its effectiveness, regardless of the involved algorithms and the type of events taken into account.

On the basis of the achieved improvements, the present work inspires some future research directions, in order to define additional intrusion detection approaches based on more sophisticated solutions, such as those based on hybrid-based or ensemble-based approaches, which involves more than a single algorithm of classification.

As a final note, we would like to point out that we did not consider sophisticated data pre-processing techniques, since our objective is to demonstrate the LFE strategy capability to improve the baseline algorithms performance without computational efforts, as this effect suggests an ability to also improve more complex approaches. Part of our future research will be oriented in this direction, together with the adoption of other datasets that involve different network scenarios such as Internet of Things (IoT), Industrial Internet of Things (IIoT), Internet of Health Things (IoHT), and so on.

Author Contributions: Conceptualization, S.C., A.S.P., D.R.R. and R.S.; data curation, S.C., A.S.P., D.R.R. and R.S.; formal analysis, S.C., A.S.P., D.R.R. and R.S.; methodology, S.C., A.S.P., D.R.R. and R.S.; resources, S.C., A.S.P., D.R.R. and R.S.; supervision, S.C.; validation, S.C., A.S.P., D.R.R. and R.S.; writing—original draft, S.C., A.S.P., D.R.R. and R.S.; and writing—review and editing, S.C., A.S.P., D.R.R. and R.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded and supported by "Bando 'Aiuti per progetti di Ricerca e Sviluppo'—POR FESR 2014-2020—Asse 1, Azione 1.1.3. Project IntelliCredit: AI-powered digital lending platform" and Fondazione di Sardegna 2017 research project "Algorithms for Approximation with Applications [Acube]".

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sundaram, A. An introduction to intrusion detection. *Crossroads* **1996**, *2*, 3–7. [CrossRef]
2. Pfleeger, C.P.; Pfleeger, S.L. *Security in Computing*, 4th ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2012.

3. Tidjon, L.N.; Frappier, M.; Mammar, A. Intrusion detection systems: A cross-domain overview. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3639–3681. [CrossRef]
4. Yüksel, B.; Küpçü, A.; Özkasap, Ö. Research issues for privacy and security of electronic health services. *Future Gener. Comput. Syst.* **2017**, *68*, 1–13. [CrossRef]
5. Luker, M.A.; Petersen, R.J. *Computer and Network Security in Higher Education*; Jossey-Bass: San Francisco, CA, USA, 2003.
6. Wazid, M.; Zeadally, S.; Das, A.K. Mobile banking: Evolution and threats: Malware threats and security solutions. *IEEE Consum. Electron. Mag.* **2019**, *8*, 56–60. [CrossRef]
7. Verykios, V.S.; Bertino, E.; Fovino, I.N.; Provenza, L.P.; Saygin, Y.; Theodoridis, Y. State-of-the-art in privacy preserving data mining. *ACM Sigmod Record* **2004**, *33*, 50–57. [CrossRef]
8. Narwaria, M.; Arya, S. Privacy preserving data mining—'A state of the art'. In Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 16–18 March 2016; pp. 2108–2112.
9. Pahlavan, K.; Krishnamurthy, P. *Principles of Wireless Networks: A Unified Approach*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2011.
10. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
11. Willner, A. *Optical Fiber Telecommunications*; Academic Press: Cambridge, MA, USA, 2019; Volume 11,
12. Gupta, A.; Jha, R.K. A survey of 5G network: Architecture and emerging technologies. *IEEE Access* **2015**, *3*, 1206–1232. [CrossRef]
13. Neupane, K.; Haddad, R.; Chen, L. Next generation firewall for network security: A survey. In Proceedings of the SoutheastCon 2018, St. Petersburg, FL, USA, 19–22 April 2018; pp. 1–6.
14. Verma, P.; Anwar, S.; Khan, S.; Mane, S.B. Network intrusion detection using clustering and gradient boosting. In Proceedings of the 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bangalore, India, 10–12 July 2018; pp. 1–7.
15. Perwira, R.I.; Fauziah, Y.; Mahendra, I.P.R.; Prasetyo, D.B.; Simanjuntak, O.S. Anomaly-based Intrusion Detection and Prevention Using Adaptive Boosting in Software-defined Network. In Proceedings of the 2019 5th International Conference on Science in Information Technology (ICSITech), Yogyakarta, Indonesia, 23–24 October 2019; pp. 188–192.
16. Miah, M.O.; Khan, S.S.; Shatabda, S.; Farid, D.M. Improving Detection Accuracy for Imbalanced Network Intrusion Classification using Cluster-based Under-sampling with Random Forests. In Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 3–5 May 2019; pp. 1–5.
17. Rai, K.; Devi, M.S.; Guleria, A. Decision tree based algorithm for intrusion detection. *Int. J. Adv. Netw. Appl.* **2016**, *7*, 2828.
18. Saia, R.; Carta, S.; Recupero, D.R. A Probabilistic-driven Ensemble Approach to Perform Event Classification in Intrusion Detection System. In *KDIR*; ScitePress: Setúbal, Portugal, 2018; pp. 139–146.
19. Hamid, Y.; Shah, F.A.; Sugumaran, M. Wavelet neural network model for network intrusion detection system. *Int. J. Inf. Technol.* **2019**, *11*, 251–263. [CrossRef]
20. Liu, W.; Liu, X.; Di, X.; Qi, H. A novel network intrusion detection algorithm based on Fast Fourier Transformation. In Proceedings of the 2019 1st International Conference on Industrial Artificial Intelligence (IAI), Shenyang, China, 23–27 July 2019; pp. 1–6.
21. Park, S.; Seo, S.; Jeong, C.; Kim, J. Network intrusion detection through online transformation of eigenvector reflecting concept drift. In Proceedings of the First International Conference on Data Science, E-learning and Information Systems, Madrid, Spain, 1–2 October 2018; pp. 1–4.
22. Saia, R.; Carta, S. A Frequency-domain-based Pattern Mining for Credit Card Fraud Detection. In Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security, IoTBDS 2017, Porto, Portugal, 24–26 April 2017.
23. Carta, S.; Corriga, A.; Ferreira, A.; Podda, A.S.; Recupero, D.R. A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. *Appl. Intell.* **2020**, 1–17.
24. Saia, R.; Carta, S. Evaluating Credit Card Transactions in the Frequency Domain for a Proactive Fraud Detection Approach. In Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017)—Volume 4: SECRYPT, Madrid, Spain, 24–26 July 2017.

25. Carta, S.; Ferreira, A.; Podda, A.S.; Recupero, D.R.; Sanna, A. Multi-DQN: An Ensemble of Deep Q-Learning Agents for Stock Market Forecasting. *Expert Syst. Appl.* **2020**, *300*, 113820. [CrossRef]
26. Saia, R. A discrete wavelet transform approach to fraud detection. In *International Conference on Network and System Security*; Springer: New York, NY, USA, 2017; pp. 464–474.
27. Saia, R.; Carta, S. A Linear-dependence-based Approach to Design Proactive Credit Scoring Models. In Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016)—Volume 1: KDIR, Portugal, 9–11 November, 2016.
28. Saia, R.; Carta, S.; Fenu, G. A Wavelet-based Data Analysis to Credit Scoring. In Proceedings of the 2nd International Conference on Digital Signal Processing, ICDSP 2018, Tokyo, Japan, 25–27 February 2018.
29. Saia, R.; Carta, S. A fourier spectral pattern analysis to design credit scoring models. In Proceedings of the 1st International Conference on Internet of Things and Machine Learning, IML 2017, Liverpool, UK, 17–18 October 2017.
30. Barra, S.; Carta, S.M.; Corriga, A.; Podda, A.S.; Reforgiato Recupero, D. Deep Learning and Time Series-to-Image Encoding for Financial Forecasting. *IEEE/CAA J. Autom. Sinica* **2020**, *7*, 683. [CrossRef]
31. Radhakrishnan, K.; Menon, R.R.; Nath, H.V. A survey of zero-day malware attacks and its detection methodology. In Proceedings of the TENCON 2019-2019 IEEE Region 10 Conference (TENCON), Kochi, India, 17–20 October 2019; pp. 533–539.
32. Saia, R.; Carta, S.; Recupero, D.R.; Fenu, G. A Feature Space Transformation to Intrusion Detection Systems. In *KDIR*; ScitePress: Setúbal, Portugal, 2020.
33. Saia, R.; Carta, S.; Recupero, D.R.; Fenu, G.; Stanciu, M. A Discretized Extended Feature Space (DEFS) Model to Improve the Anomaly Detection Performance in Network Intrusion Detection Systems. In *KDIR*; ScitePress: Setúbal, Portugal, 2019; pp. 322–329.
34. Saia, R.; Carta, S.; Recupero, D.R.; Fenu, G.; Saia, M. A Discretized Enriched Technique to Enhance Machine Learning Performance in Credit Scoring. In *KDIR*; ScitePress: Setúbal, Portugal, 2019; pp. 202–213.
35. Cai, J.; Luo, J.; Wang, S.; Yang, S. Feature selection in machine learning: A new perspective. *Neurocomputing* **2018**, *300*, 70–79. [CrossRef]
36. Zheng, A.; Casari, A. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2018.
37. Anwer, H.M.; Farouk, M.; Abdel-Hamid, A. A framework for efficient network anomaly intrusion detection with features selection. In Proceedings of the 2018 9th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 3–5 April 2018; pp. 157–162.
38. Zhao, F.; Zhao, J.; Niu, X.; Luo, S.; Xin, Y. A filter feature selection algorithm based on mutual information for intrusion detection. *Appl. Sci.* **2018**, *8*, 1535. [CrossRef]
39. Mohammadi, S.; Mirvaziri, H.; Ghazizadeh-Ahsaee, M.; Karimipour, H. Cyber intrusion detection by combined feature selection algorithm. *J. Inf. Secur. Appl.* **2019**, *44*, 80–88. [CrossRef]
40. Alazzam, H.; Sharieh, A.; Sabri, K.E. A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer. *Expert Syst. Appl.* **2020**, *148*, 113249. [CrossRef]
41. Rashid, A.B.; Ahmed, M.; Sikos, L.F.; Haskell-Dowland, P. A Novel Penalty-Based Wrapper Objective Function for Feature Selection in Big Data Using Cooperative Co-Evolution. *IEEE Access* **2020**, *8*, 150113–150129. [CrossRef]
42. Rashid, A.B.; Choudhury, T. Knowledge management overview of feature selection problem in high-dimensional financial data: Cooperative co-evolution and Map Reduce perspectives. *Probl. Perspect. Manag.* **2019**, *17*, 340.
43. Kasongo, S.M.; Sun, Y. A deep learning method with filter based feature engineering for wireless intrusion detection system. *IEEE Access* **2019**, *7*, 38597–38607. [CrossRef]
44. Wang, W.; Sheng, Y.; Wang, J.; Zeng, X.; Ye, X.; Huang, Y.; Zhu, M. HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access* **2017**, *6*, 1792–1806. [CrossRef]
45. Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U. Autoencoder-based feature learning for cyber security applications. In Proceedings of the 2017 International joint conference on neural networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3854–3861.

46. Kunang, Y.N.; Nurmaini, S.; Stiawan, D.; Zarkasi, A.; Jasmir, F. Automatic Features Extraction Using Autoencoder in Intrusion Detection System. In Proceedings of the 2018 International Conference on Electrical Engineering and Computer Science (ICECOS), Pangkal Pinang, Indonesia, 2–4 October 2018; pp. 219–224.
47. Ieracitano, C.; Adeel, A.; Morabito, F.C.; Hussain, A. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. *Neurocomputing* **2020**, *387*, 51–62. [CrossRef]
48. Anderson, J.P. *Computer Security Threat Monitoring and Surveillance*; Technical Report; James P. Anderson Company: Fort Washington, MD, USA, 1980. Available online: http://ranger.uta.edu/~dliu/courses/cse6392-ids-spring2007/papers/Anderson80-IDS-TechReport.pdf(accessed on 21 October 2020).
49. Denning, D.E. An intrusion-detection model. *IEEE Trans. Software Eng.* **1987**, *13*, 222–232. [CrossRef]
50. Axelsson, S. Intrusion Detection Systems: A Survey and Taxonomy. Technical Report. 2000; Volume 99. Available online: http://www.cse.msu.edu/~cse960/Papers/security/axelsson00intrusion.pdf (accessed on 20 October 2020).
51. Modi, C.; Patel, D.; Borisaniya, B.; Patel, H.; Patel, A.; Rajarajan, M. A survey of intrusion detection techniques in cloud. *J. Netw. Comput. Appl.* **2013**, *36*, 42–57. [CrossRef]
52. Zarpel ao, B.B.; Miani, R.S.; Kawakani, C.T.; de Alvarenga, S.C. A survey of intrusion detection in Internet of Things. *J. Netw. Comput. Appl.* **2017**, *84*, 25–37. [CrossRef]
53. Rehman, R.; Hazarika, G.; Chetia, G. Malware threats and mitigation strategies: A survey. *J. Theor. Appl. Inf. Technol.* **2011**, *29*, 69–73.
54. Campbell, T. Protection of Systems. In *Practical Information Security Management*; Springer: New York, NY, USA, 2016; pp. 155–177.
55. Latha, S.; Prakash, S.J. A survey on network attacks and Intrusion detection systems. In Proceedings of the 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 6–7 January 2017; pp. 1–7.
56. Samrin, R.; Vasumathi, D. Review on anomaly based network intrusion detection system. In Proceedings of the 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), Mysuru, India, 15–16 December 2017; pp. 141–147.
57. Bronte, R.; Shahriar, H.; Haddad, H.M. A signature-based intrusion detection system for web applications based on genetic algorithm. In Proceedings of the 9th International Conference on Security of Information and Networks, Newark, NJ, USA, 20–22 July 2016; pp. 32–39.
58. Aldwairi, M.; Abu-Dalo, A.M.; Jarrah, M. Pattern matching of signature-based IDS using Myers algorithm under MapReduce framework. *EURASIP J. Inf. Secur.* **2017**, 1–11. [CrossRef]
59. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
60. Werth, A.; Morris, T.H. A Specification-Based Intrusion Prevention System for Malicious Payloads. In *National Cyber Summit*; Springer: New York, NY, USA, 2019; pp. 153–168.
61. Li, Z.; Das, A.; Zhou, J. Usaid: Unifying signature-based and anomaly-based intrusion detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*; Springer: New York, NY, USA, 2005; pp. 702–712.
62. Bostani, H.; Sheikhan, M. Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach. *Comput. Commun.* **2017**, *98*, 52–71. [CrossRef]
63. Jose, S.; Malathi, D.; Reddy, B.; Jayaseeli, D. *A Survey on Anomaly Based Host Intrusion Detection System*; Journal of Physics: Conference Series; IOP Publishing: Bristol, UK, 2018; Volume 1000, p. 012049.
64. Mazini, M.; Shirazi, B.; Mahdavi, I. Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms. *J. King Saud. Univ.-Comput. Inf. Sci.* **2019**, *31*, 541–553. [CrossRef]
65. Potluri, S.; Diedrich, C. High Performance Intrusion Detection and Prevention Systems: A Survey. In Proceedings of the 15th European Conference on Cyber Warfare and Security (ECCWS 2016), Munich, Germany, 7–8 July 2016; p. 260.
66. Amrita, K.K.R. A hybrid intrusion detection system: Integrating hybrid feature selection approach with heterogeneous ensemble of intelligent classifiers. *Int. J. Netw. Secur.* **2018**, *20*, 41–55.
67. Mahjabin, T.; Xiao, Y.; Sun, G.; Jiang, W. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *Int. J. Distrib. Sen. Netw.* **2017**, *13*, 1550147717741463. [CrossRef]

68. Rajkumar, M.N. A survey on latest DoS attacks: Classification and defense mechanisms. *Int. J. Innov. Res. Comput. Commun. Eng.* **2013**, *1*, 1847–1860.
69. Hoque, N.; Bhuyan, M.H.; Baishya, R.C.; Bhattacharyya, D.K.; Kalita, J.K. Network attacks: Taxonomy, tools and systems. *J. Netw. Comput. Appl.* **2014**, *40*, 307–324. [CrossRef]
70. Mopari, I.; Pukale, S.; Dhore, M. Detection and defense against DDoS attack with IP spoofing. In Proceedings of the 2008 International Conference on Computing, Communication and Networking, St. Thomas, VI, USA, 18–20 December 2008; pp. 1–5.
71. Yu, W.; Yalin, Y.; Haodan, R. Research on the Technology of Trojan Horse Detection. In Proceedings of the 2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA), Xiangtan, China, 26–27 October 2019; pp. 117–119.
72. Zhang, X.y.; Qing, S.h.; Ma, H.t.; Zhang, N.; Sun, S.h.; Jiang, J.c. Research on the concealing technology of Trojan horses. *J.-China Institute Commun.* **2004**, *25*, 153–159.
73. Zhuo, Z.; Zhang, Y.; Zhang, Z.L.; Zhang, X.; Zhang, J. Website fingerprinting attack on anonymity networks based on profile hidden markov model. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 1081–1095. [CrossRef]
74. Jahani, H.; Jalili, S. Effective defense against fingerprinting attack based on autocorrelation property minimization approach. *J. Intell. Inf. Syst.* **2020**, *54*, 341–362. [CrossRef]
75. Bhardwaj, M.; Bawa, S. Fuzz testing in stack-based buffer overflow. In *Advances in Computer Communication and Computational Sciences*; Springer: New York, NY, USA, 2019; pp. 23–36.
76. Deckard, J. *Buffer Overflow Attacks: Detect, Exploit, Prevent*; Elsevier: Amsterdam, The Netherlands, 2005.
77. Daş, R.; Karabade, A.; Tuna, G. Common network attack types and defense mechanisms. In Proceedings of the 2015 23nd Signal Processing and Communications Applications Conference (SIU), Malatya, Turkey, 16–19 May 2015; pp. 2658–2661.
78. Sabillon, R.; Cano, J.; Cavaller Reyes, V.; Serra Ruiz, J. Cybercrime and cybercriminals: A comprehensive study. *Int. J. Comput. Netw. Commun. Secur.* **2016**, *4*, 165–176.
79. Aung, M.A.C.; Thant, K.P. Detection and mitigation of wireless link layer attacks. In Proceedings of the 15th IEEE International Conference on Software Engineering Research, Management and Applications (SERA), London, UK, 7–9 June 2017; pp. 173–178.
80. Sikos, L.F. Packet analysis for network forensics: A comprehensive survey. *Forensic Sci. Int. Dig. Investig.* **2020**, *32*, 200892. [CrossRef]
81. Singla, A.; Bertino, E. How deep learning is making information security more intelligent. *IEEE Secur. Priv.* **2019**, *17*, 56–65. [CrossRef]
82. Rodda, S.; Erothi, U.S.R. Class imbalance problem in the network intrusion detection systems. In Proceedings of the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Palanchur, Nazarethpet, Chennai, Tamil Nadu, India, 3–5 March 2016; pp. 2685–2688.
83. Cieslak, D.A.; Chawla, N.V.; Striegel, A. *Combating Imbalance in Network Intrusion Datasets*; In Proceedings of the IEEE International Conference on Granular Computing (GrC 2006), Atlanta, GA, USA, 10–12 May 2006; pp. 732–737.
84. Kumar, G. Evaluation metrics for intrusion detection systems—A study. *Evaluation* **2014**, *2*, 11–7.
85. Munaiah, N.; Meneely, A.; Wilson, R.; Short, B. Are Intrusion Detection Studies Evaluated Consistently? A Systematic Literature Review. Technical Report. 2016. Available online: https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=2837&context=article (accessed on 20 October 2020).
86. Chuang, H.M.; Huang, H.Y.; Liu, F.; Tsai, C.H. Classification of Intrusion Detection System Based on Machine Learning. In *International Cognitive Cities Conference*; Springer: New York, NY, USA, 2019; pp. 492–498.
87. Huang, J.; Ling, C.X. Using AUC and accuracy in evaluating learning algorithms. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 299–310. [CrossRef]
88. Chai, T.; Draxler, R.R. Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE in the literature. *Geosci. Model Dev.* **2014**, *7*, 1247–1250. [CrossRef]
89. Vijayanand, R.; Devaraj, D.; Kannapiran, B. Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. *Comput. Secur.* **2018**, *77*, 304–314. [CrossRef]
90. Chatfield, B.; Haddad, R.J.; Chen, L. Low-computational complexity intrusion detection system for jamming attacks in smart grids. In Proceedings of the International Conference on Computing, Networking and Communications (ICNC 2018), Maui, HI, USA, 5–8 March 2018; pp. 367–371.

91. Liu, H.; Hussain, F.; Tan, C.L.; Dash, M. Discretization: An enabling technique. *Data Mining Knowl. Discov.* **2002**, *6*, 393–423. [CrossRef]
92. Varma, P.R.K.; Kumari, V.V.; Kumar, S.S. A survey of feature selection techniques in intrusion detection system: A soft computing perspective. In *Progress in Computing, Analytics and Networking*; Springer: New York, NY, USA, 2018; pp. 785–793.
93. Nargesian, F.; Samulowitz, H.; Khurana, U.; Khalil, E.B.; Turaga, D.S. Learning Feature Engineering for Classification. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2017), Melbourne, Australia, 19–25 August 2017; pp. 2529–2535.
94. García, S.; Ramírez-Gallego, S.; Luengo, J.; Benítez, J.M.; Herrera, F. Big data preprocessing: Methods and prospects. *Big Data Anal.* **2016**, *1*, 9. [CrossRef]
95. Wu, X.; Kumar, V. *The Top Ten Algorithms in Data Mining*; CRC Press: Boca Raton, FL, USA, 2009.
96. Chivers, I.; Sleightholme, J. An introduction to Algorithms and the Big O Notation. In *Introduction to Programming with Fortran*; Springer: New York, NY, USA, 2015; pp. 359–364.
97. Computational Complexity of Machine Learning Algorithms. 2018. Available online: https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/ (accessed on 31 August 2020).
98. Hashem, I.A.T.; Anuar, N.B.; Gani, A.; Yaqoob, I.; Xia, F.; Khan, S.U. MapReduce: Review and open challenges. *Scientometrics* **2016**, *109*, 389–422. [CrossRef]
99. He, Q.; Zhuang, F.; Li, J.; Shi, Z. Parallel implementation of classification algorithms based on MapReduce. In *International Conference on Rough Sets and Knowledge Technology*; Springer: New York, NY, USA, 2010; pp. 655–662.
100. Dean, J.; Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Commun. ACM* **2008**, *51*, 107–113. [CrossRef]
101. Wang, Y.; Yang, K.; Jing, X.; Jin, H.L. Problems of KDD Cup 99 Dataset Existed and Data Preprocessing. *Appl. Mech. Mater. Trans. Tech. Publ.* **2014**, *667*, 218–225. [CrossRef]
102. De la Hoz, E.; Ortiz García, A.; Ortega Lopera, J.; De La Hoz Correa, E.M.; Mendoza Palechor, F.E. Implementation of an Intrusion Detection System Based on Self Organizing Map. *J. Theor. Appl. Inf. Technol.* **2015**. Available online: https://repositorio.cuc.edu.co/handle/11323/3253 (accessed on 21 October 2020).
103. Luque, A.; Carrasco, A.; Martín, A.; de las Heras, A. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognit.* **2019**, *91*, 216–231. [CrossRef]
104. Boughorbel, S.; Jarray, F.; El-Anbari, M. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PLoS ONE* **2017**, *12*, e0177678. [CrossRef] [PubMed]
105. Powers, D.M. *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation*; Bioinfo Publications: Malwadi, India, 2011. Available online: https://csem.flinders.edu.au/research/techreps/SIE07001.pdf (accessed on 20 October 2020).
106. Bansal, A.; Kaur, S. Extreme gradient boosting based tuning for classification in intrusion detection systems. In *International Conference on Advances in Computing and Data Sciences*; Springer: New York, NY, USA, 2018; pp. 372–380.
107. Zhou, Y.; Mazzuchi, T.A.; Sarkani, S. M-AdaBoost—A Based Ensemble System for Network Intrusion Detection. *Exp. Syst. Appl.* **2020**, *162*, 113864. [CrossRef]
108. Negandhi, P.; Trivedi, Y.; Mangrulkar, R. Intrusion Detection System Using Random Forest on the NSL-KDD Dataset. In *Emerging Research in Computing, Information, Communication and Applications*; Springer: New York, NY, USA, 2019; pp. 519–531.
109. Farnaaz, N.; Jabbar, M. Random forest modeling for network intrusion detection system. *Procedia Comput. Sci.* **2016**, *89*, 213–217. [CrossRef]
110. Chisholm, K.; Yakopcic, C.; Alam, M.S.; Taha, T.M. Multilayer Perceptron Algorithms for Network Intrusion Detection on Portable Low Power Hardware. In Proceedings of the 10th Annual Computing and Communication Workshop and Conference, CCWC 2020, Las Vegas, NV, USA, 6–8 January 2020; pp. 901–906.
111. Ingre, B.; Yadav, A.; Soni, A.K. Decision tree based intrusion detection system for NSL-KDD dataset. In *International Conference on Information and Communication Technology for Intelligent Systems*; Springer: New York, NY, USA, 2017; pp. 207–218.
112. Zhao, H.; Li, M.; Wu, T.; Yang, F. Evaluation of Supervised Machine Learning Techniques for Dynamic Malware Detection. *Int. J. Comput. Intell. Syst.* **2018**, *11*, 1153–1169. [CrossRef]

113. Dietterich, T.G. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.* **1998**, *10*, 1895–1923. [CrossRef]
114. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Portugal, 22–24 January 2018; pp. 108–116.
115. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6.