VOL. 39 Nº2 2022
https://doi.org/10.22491/IMECTA.10831.en

Detection of Cyber Attacks using Machine Learning: A Review

Ricardo da Silveira Lopes*, Julio Cesar Duarte and Ronaldo Ribeiro Goldschmidt
Military Institute of Engineering (IME)
Praça Gen. Tibúrcio, 80 – Urca, Rio de Janeiro – RJ, 22290-270
***ricardo.lopes@ime.eb.br**

ABSTRACT: *With the public availability of simulated intrusion detection datasets, Machine Learning has been increasingly used in cyber attack detection work. Despite the fact that the performance (precision and recall) has been highlighted, on the other hand, there has been a lack of critical analysis of what was actually learned by the model, with the intention to conclude whether or not this performance will be maintained in real applications. In this sense, explainability techniques appear as a promising possibility in the execution of this task, since the analysis of the False Positive Rate of these models has usually been neglected. This can become an important problem, with the increase in speed and amount of data transmitted over the internet. This research proposes to raise discussions about these problems, presenting some articles related to them.*

KEYWORDS: *Intrusion Detection. Machine Learning. Explainability. False Positive Rate.*

RESUMO: *Com a disponibilização pública de bases de dados simuladas de detecção de intrusão, o Aprendizado de Máquina vem sendo empregado, cada vez mais, em estudos de detecção de ataques cibernéticos. Se, por um lado, tem-se destacado o desempenho (precisão e abrangência) obtido, por outro, tem havido uma carência na análise crítica sobre o que de fato foi aprendido pelo modelo, visando concluir se haverá ou não a manutenção desse desempenho em aplicações reais. Nesse sentido, técnicas de explicabilidade surgem como uma possibilidade promissora na execução dessa tarefa, uma vez que, usualmente, vem sendo negligenciada a análise da Taxa de Falso Positivo desses modelos, o que pode se tornar um problema importante, com o aumento da velocidade e quantidade de dados trafegados pela internet. Esta pesquisa se propõe a levantar discussões sobre esses problemas, apresentando alguns artigos a eles relacionados.*

PALAVRAS-CHAVE: *Detecção de Intrusão. Aprendizado de Máquina. Explicabilidade. Taxa de Falso Positivo.*

1. Introduction

The emergence of internet has revolutionized modern life, making it possible to carry out numerous activities online. Purchases, banking transactions, meetings, classes, courses, interactions via social networks, text, voice and video communication, remote management, etc., have become everyday activities in society [1]. On the other hand, the increased use of informatics and online tools has brought with it vulnerabilities widely exploited by ill-intentioned individuals and organizations through malicious actions in cyberspace. These actions are known as cyberattacks or intrusions, and cause considerable harm to users and businesses that inevitably use the Internet. Typically, these attacks have diverse purposes, such as illicitly obtaining financial benefits, harming institutions, propagating ideologies and even terrorism [2].

Thus, cyberattacks have been an increasing problem as society becomes more dependent on information technology. Two factors are responsible for this problem: the existence of vulnerabilities in information systems and the presence of agents with the potential to exploit such vulnerabilities. These agents can be individuals, groups, and even nations.

In addition, the increasing insertion of computers and data networks in processes of management, monitoring, automation and control of critical infrastructures, such as power generation plants, transportation systems, water collection, storage and distribution stations, emergency services, etc., is inevitable. In this sense, the problem can worsen considerably in cases related to this type of infrastructure and, depending on the vulnerabilities in them and the criticality of the attack, such damage can be catastrophic. An example of this occurred recently in Florida, United States, where a *hacker* gained access to the water treatment system,

increasing the proportion of caustic soda and exposing the local population to the risk of chemical contamination [3]. Fortunately, a local employee noticed what had happened and reversed the action in a timely manner.

Thus, the detection of cyberattacks has assumed a leading role in issues related to the prevention and mitigation of threats in the field of information technology.

2. Attack detection

Cyber attacks are offensive and malicious maneuvers directed at information systems, computer infrastructures, data networks and personal computing equipment. They have the purpose of exposing, changing, disabling, destroying, stealing or obtaining unauthorized access to data or computational resources [4]. Thus, a cyber attack compromises at least one of the three aspects of information security: confidentiality, integrity or availability [5].

Attack detection is a task performed by an *Intrusion Detection System* (IDS) – in this context, the terms attack and intrusion will be used interchangeably. This detection arises from monitoring events that occur in a computer system or data network. These events are then analyzed, to find signs of possible incidents representing violations or imminent threats of security policies breach [6]. As illustrated in **Figure 1**, there are two basic detection methods:

- Signature-based detection: detection of traces, called signatures, that uniquely identify a given attack. These traces are stored in a database that needs to be constantly updated as new types of attacks emerge.
- Anomaly-based detection: detection of patterns that are outside those considered normal or acceptable. This normal behavior can be defined by a data network security expert or can be learned by some Machine Learning (ML) technique.

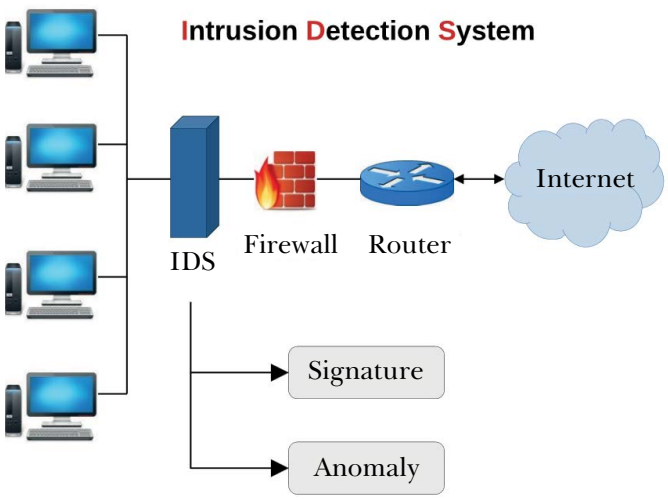

Fig. 1 – Example of IDS use. **Source: Adapted from [7].**

The schematic in **Figure 1** shows an IDS composed of only one equipment at a specific location on the network, typically between the *firewall* and the local network to be protected. The advantage of this schematic is that it considerably decreases the amount of malicious data to be analyzed by IDS, since much of the traffic considered inappropriate is barred in the *firewall*. In addition, larger institutions usually have a very extensive local network, often composed of numerous subnets. In this case, the IDS deployment is more complex, requiring the installation of other components, such as:

- Sensors: equipment connected at different points in the network, to collect data, for example, IP packets;
- Agents: software with functions similar to sensors, but installed on *hosts*. In this way, agents monitor computers and may collect data other than network-specific data, for example, access to the file system;
- Management server: responsible for receiving, processing and correlating the data sent by sensors and agents. It is usually in this equipment that the intrusion alert is generated;
- Database server: responsible for storing the data collected by agents and sensors;
- Console: responsible for providing an interface for IDS users and administrators.

Typically, the management and data exchanged between these devices takes place on an independent

network, which provides greater protection against attacks directed at an IDS. This independent network can ideally be a second physical network, providing more security, but has a higher installation cost. An alternative is to use a *Virtual Local Area Network* (VLAN) that shares the same physical network, or to traffic management data without the use of VLAN, but with encryption. Both alternatives have a lower installation cost, although they increase the bandwidth consumption of the main data network, and also reduce the safety of the equipment that makes up the IDS.

Regardless of the size of the network and the number of sensors and agents, in practice IDS based on signature and anomaly are used together, since one type complements the other. This is because a signature-based IDS has the advantage of having a low False Positive Rate (FPR), although it is inefficient in detecting new types of attacks. This inefficiency remains even when these new attacks are only minor variations of those already catalogued. In the case of an anomaly-based IDS, the opposite occurs: it has the advantage of detecting new attacks, but with a trend of higher FPR. This is because it is difficult to precisely model the normal behavior of the network, which can vary considerably depending on its size and complexity. In addition, this normal behavior can evolve over time, requiring revisions and updates of the model that represents it.

In short, the IDS should ideally provide a high coverage (also known as sensitivity, *recall*, True Positive Rate (TPR), or detection rate), with a low FPR. This characteristic is highly desirable, being represented by the largest possible area under the ROC (*Receiver Operating Characteristic*) curve. This curve was developed by military radar operators, showing the trade-off between the FPR and the recall of the radar receiver, which explains its nomenclature. **Figure 2** characterizes the ROC curves of three different classifiers. Different points belonging to the same curve mean different decision thresholds for the same classifier. Note that the best classifiers can achieve greater recall with low FPR and that this results in a greater area under the ROC curve. In addition, the smallest area occurs when the classifier is purely random. In this case, where the area is equal to 0.5, the classifier always has the same probability of detection, regardless of the sample to be classified.

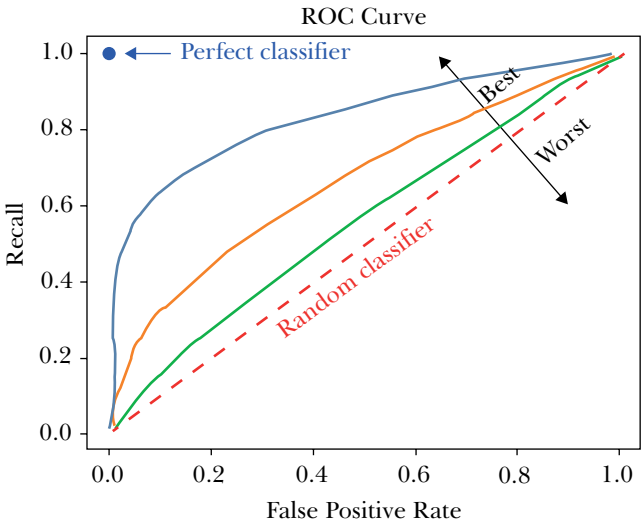

Fig. 2 – Examples of ROC curve. **Source: Adapted from [8].**

3. Machine Learning

ML is a field of Artificial Intelligence that is strongly related to computational statistics, focusing on prediction activity through computational optimization methods [9], which enables the execution of classification or regression tasks by the computer. Classifying means determining to which class a given sample belongs. As there is a finite number of classes, these can be represented by discrete values. An example would be spam or non-spam emails classification. In the regression task, the sample is related to a continuous value, such as prediction of real estate values.

Such samples are represented by their important characteristics, also called attributes. In the case of a *spam* rating, for example, each email is considered a sample. A possible attribute may be the number of times a given word, or word composition, appears in the email. Examples of such words are: amazing, satisfaction, now, bonus, win, offer, discount etc. In the case of real estate, some examples of attributes are the area, the number of rooms, the location and the age.

In this way, computational algorithms inductively "learn" relationships between existing attributes in

a database [10]. The two main forms of learning are: supervised and unsupervised. In the first case, examples are provided, composed of attributes and responses, so that the model learns the target function in an inductive and approximate way. This function represents the mapping between attributes and responses (also known as target attribute or labels), being usually complex and unknown [11]. *Spam* email classification and property value prediction are supervised tasks since target values (*spam*, *non-spam*, and property price) are provided. In unsupervised learning, there is no provision of labels to the algorithm. In this case, the objective of the model is to learn, by itself, some structure inherent to the database [11]. For example, there are sample grouping techniques based on similarity metrics, known as clustering. Projection techniques, such as *Principal Component Analysis* (PCA), also belong to this category. Such techniques aim at grouping and reducing data dimensionality, two pillars of unsupervised learning [12].

For an effective application of ML, the database is divided into three sets: training, validation, and testing. In the training set, computational algorithms are applied iteratively, in order to minimize a function known as cost. This function indicates whether the predicted values from the attributes are, on average, close to those contained in the labels [11]. The lower the value of the cost function, the greater this proximity will be. A technique widely used in this minimization is the gradient descent, which adjusts the model parameters automatically, based on the gradient of the cost function in relation to these parameters.

The test set is used to verify the performance of the model in different examples of the training set, indicating whether the relationship between attributes and labels has been adequately learned. In this ideal case, the model can generalize this relationship well, being able to disregard possible noises inherent in the training set.

When this ideal situation does not occur, it is likely that the model has a high bias or overfitting. Although both degrade performance in the test set, their causes are quite distinct. High bias occurs when the model is very simple, unable to approximate the target function, regardless of any adjustment in its parameters. This problem is verified when the performance is low in both the training set and the test set. To solve it, one must replace the model with a more complex one, and it may also be necessary to obtain a greater number of attributes. On the other hand, overfitting occurs when the model is able to approximate functions more complex than the target function itself [13]. Consequently, the model ends up generating a function that fits too much to the training set, influenced by noise and imperfections, which do not generalize the target function properly [10, 14]. This problem is verified when there is a considerably higher performance in the training set compared to the test set. Overfitting can be reduced by obtaining a greater amount of training examples, or even by regularization techniques.

In addition to the model parameters, which are automatically adjusted by optimization algorithms, there are others that need to be adjusted manually and empirically, called hyperparameters. Some examples of hyperparameters are: depth of the decision tree, number of neurons and layers of the neural network, learning rate, regularization rate, etc. The main function of the validation set is to assist in the adjustment of hyperparameters [15]. These receive different values, which are used to train the model in the training set and then validated in the validation set, where the hyperparameters that obtained the best performance are selected. This procedure can be performed by means of a Grid Search.

One application of ML that has been gaining ground in academia is the detection and classification of cyberattacks. This is mainly due to the availability of public databases, as presented in section 4.

4. Database with anomalous traffic

Databases are essential for ML, as they contain the information to be learned inductively by the model. However, there is a great lack of quality databases, obtained by real traffic collection. One reason for this is that companies may end up exposing some

vulnerability or sensitive information by making data available for research on the event of a suffered invasion. Another reason is that it is difficult to obtain real traffic labels. A *hacker* would probably not have the goodwill to inform the victim of the class of attack employed. In fact, he won't even report an attack. An alternative is to obtain the labels through a signature IDS; however, unknown attacks (yet without signature) would be labeled as normal traffic, negatively influencing the learning of the algorithm.

Thus, it is no coincidence that practically all public cyberattack databases were created through simulation. An important factor to be considered in these bases is the level of similarity in relation to a real situation. Probably, a high-performing IDS on an unrealistic basis may not work properly in a real job.

Another relevant aspect refers to the examples contained in these bases. They can be provided in raw data (in the form of IP packets), or in processed data (in the form of traffic or network flow). In the first case, each IP packet represents an example of the database. Some attributes are common to all packages, such as header size, total package size, protocol, source and destination IP address. Other attributes are protocol-specific (TCP, UDP, etc.) such as source and destination ports, which do not exist in all protocols. These packages can be captured and stored in pcap format, through specific applications. Some popular applications are TCPDump, Wireshark, Snort and Nmap [16].

In the form of network flow, these IP packets are processed and formatted in such a way that each example of the database consists of information that defines a uni- or bi-directional sequence of packets that share the following attributes: source IP address, destination IP address, protocol, source port and destination port [16]. TCP connections are an example of bidirectional data flow. In addition, other statistical data may be added to these attributes, such as, for example, number of bytes emitted by the source, number of bytes emitted by the destination, average time between packets, packet rate, etc. Compared to databases in the form of IP packets, network flow databases have a smaller size because they disregard information in the packet *payload*.

From 1998, intrusion databases began to be made available, which enabled the introduction of ML techniques in the field of cybersecurity. These bases were generated by simulating malicious and benign traffic. Some of them also included background traffic, i.e., real and anonymized traffic, which are not precisely known whether benign or malignant. In some cases, malignant traffic classes are also provided. The main public bases available are:

- DARPA 1998/1999 [17]: *Defense Advanced Research Projects Agency* DARPA 1998 and DARPA 1999, created by the Massachusetts Institute of Technology (MIT), being widely used and discussed in several articles. The first was obtained in a simulation that lasted nine weeks. The first seven weeks gave rise to the training set, and the last two, to the test set. The attributes of this database are provided in the form of raw data, i.e. IP packets. One year later, a second base, entitled DARPA 1999, was available. It was generated by a five-week simulation, the first three being training and the last two for testing. This base has a significantly greater amount of attack types compared to the first. The attributes of this base are also provided through IP packets.
- KDD Cup 1999 [18]: used in a cyber attack classification competition, this base has 41 network flow-oriented attributes (NetFlow), derived from DARPA 1998. The KDD Cup 1999 database, however, has serious limitations [16] such as high sample redundancy and inaccuracies derived from packet loss during its creation, caused by excessive data traffic. In addition, the vast majority of examples, both in the training set and in the test set, were easy to classify, not requiring very complex models [19].
- NSL-KDD [20]: it was built in 2009 by a careful sampling of the KDD 1999 base, where redundancies were eliminated and the number of easy examples reduced. Thus, high accuracy was no longer possible to be obtained with overly simple models. The main simulated attacks on this base are DoS (Denial of Service), unauthorized administrator access (U2R – *User to Root*), access to a local network *host* by an unauthorized remote machine (R2L –

Remote to Local), and scanning of network resource information (*Probe* or *Scan*).

- CTU-13 [21]: was established in 2011 by the CTU University, Czech Republic. The term 13 comes from this base having thirteen different scenarios where *Botnet* attacks occur. Such scenarios vary greatly in difficulty level of attack detection. Data from this database is provided in raw data (IP packets) and processed data (NetFlow standard, WebLogs and others). Delplace, Hermoso and Anandita [22] used this basis to verify the performance of various ML models, in addition to analyzing the relative importance of each attribute.
- CIC-IDS [23]: these are public databases provided by the Canadian Institute of Cybersecurity, located at the University of New Brunswick. There are three: one launched in 2012, one in 2017 and another, more recent, in 2018. The latter two contain benign traffic and more current attacks. Data is available in raw form (pcap) and in the form of network flow, through the processing of the raw data by a tool of this Institute, called CICFlowMeter. This tool characterizes the network flow through six attributes: *time stamp*, source and destination IP address, source and destination port, and protocol. An interesting feature on this database was the method to generate benign traffic, which mimics the behavior of human interactions, therefore being more similar to actual traffic.
- AWID [24]: it is a project of the University of Aegean, located in Greece, where two public databases (AWID2 and AWID3) extracted from Wi-Fi traffic are made available. They make it possible to develop and analyze IDSs specific to the wireless environment, which has vulnerabilities different from cabled. AWID2, made available in 2015, is considered the first public base of wireless networks [25], and has three categories of attack: injection, denial of service and personification, in addition to normal traffic. These attacks are designed to exploit existing vulnerabilities in the authentication environment, which uses WEP (*Wired Equivalent Protection*), WPA, and WPA2 (*Wi-Fi Protected Access*). The AWID3, launched in 2021, used a more modern authentication environment, called the *Extensible Authentication Protocol* (EAP), enabling the study of attacks designed to exploit vulnerabilities in this new environment. Multilayer attacks have also been included, which exploit vulnerabilities of the *link* layer (802.11) and higher layers.
- UNSW-NB15 [26]: this is a public cyberattack base launched in 2015, available in pcap, BRO, Argus, and csv formats. Examples of this base were created through a tool called *IXIA Perfect Storm*, which belongs to the *Australian Center for Cyber Security* (ACCS). This base has nine attack types, available in the form of 49 attributes, in addition to the class label. It is available in the full version with a total of 2,540,044 examples and in the reduced version, with the training and test sets having 175,341 and 82,332 examples, respectively.
- IoT-23 [27]: this is a base that contains traffic captured from 2018 to 2019, and was published in January 2020 at the University of CTU, Czech Republic, being funded by the antivirus company Avast. The term 23 comes from this base having 23 different scenarios where attacks related to IoT (*Internet of Things*) equipment occur. This is a labeled database, made available in the form of pcap files and in the form of data traffic, obtained through the Zeek *software*. It has eight types of attacks, in addition to benign traffic, which was generated by three known uninfected IoT equipment.
- USB-IDS-1 [28]: this is a database made available in 2021 by the University of Sannio, Benevento (USB), Italy, consisting of a variety of DoS attacks (Hulk, TCPFlood, Slowloris and Slowhttptest) performed against a web server. It can be obtained through raw data (pcap format) or data traffic (csv format), the latter generated by the same tool used in the CIC-IDS: CICFlowMeter databases. For this reason, the existing examples in USB-IDS-1 have the same attributes as those databases, allowing to test the ML models ability to generalize on different bases. An innovative fact in the USB-IDS-1 base was the inclusion of defensive tools and configurations in the web server, which is very common in real applications. Such tools alter the traffic profile and may decrease the effectiveness

of algorithms trained in other environments. In addition, the authors showed that these tools do not guarantee a perfect mitigation of attacks, which reinforces the need to use IDS.

5. Explainability

Recently, methods that use deep learning, such as convolutional and recurrent networks, have obtained significant accuracy in image classification tasks and natural language processing. This evolution was used in the area of intrusion detection, allowing a high accuracy in the detection of cyberattacks, with reduced FPR [29]. On the other hand, such models are not very transparent, which has increased the importance of the development of methods that reasonably explain the reasons behind decisions taken by these models. However, most articles in this field are still exclusively concerned with obtaining high accuracy, high detection rates and low FPRs, without considering the understanding of the mechanisms that led the model to perform a certain classification. With the emergence of public databases, there was a tendency to a certain competition among authors of articles in this area. Finding phrases like "Our method obtained a higher accuracy, with a lower false positive rate" is very common. Also very common is to display, in these articles, a table listing the performance of several other methods, extracted from other works, usually with lower performance than proposed by them.

This search for high performance resulted in a considerable increase in the complexity of the models, which made it virtually impossible to understand the mechanism behind the decision of these algorithms in intrusion detection tasks. This does not mean there is no knowledge about how the algorithm performs the classification. For example, in *Multi Layer Perceptron* (MLP) networks, also known as neural networks, however deep and complex they may be, algorithm decisions continue to be generated by an intricate weighted combination of layer-by-layer functions. The term "weighted" refers to the weights that multiply the attributes and values of each neuron. These weights are adjusted automatically, usually by the descending gradient technique or some variation of it, to minimize the difference between the predicted class and the label to which each example of the database actually belongs. The problem is that this alone does not clarify which attributes and which relationships between these attributes led to a given classification. For the algorithm, what matters is to minimize the error, regardless of any rationality in relation to the characteristics of the examples.

This lack of transparency in complex models causes distrust, especially in non-obvious classifications, which may contradict the opinion of a human. In this case, if the model's decision does not inform its reasons, it is difficult to know if there was an error or if the model was able to perceive some important detail that escaped human perception. This problem is more relevant in applications with low fault tolerance, as occurs in the detection of an intrusion. In this case, there is a low tolerance for false negatives, as no timely action will be taken to stop an undetected attack whose presence will only be noticed for its consequences. An "opaque" model does not allow the analyst to know when and why certain attacks are not detected. The application in network intrusion is quite distinct from, for example, recommendation systems. In this case, it is not so important to know why the model recommends a particular product to a user, as long as the algorithm has a high accuracy. These cases in which the model errs, which are a minority, do not bring serious consequences, only causing the user to disregard such recommendations.

In addition, understanding the right decisions made by complex models can help the expert discover non-trivial hitherto unknown relationships between attributes and cyberattacks. On the other hand, the understanding of erroneous decisions enables model improvement. Finally, attackers can make specific changes to their attacks to evade detection. Thus, knowledge of the classification mechanism allows specialists to make adaptations that strengthen the model against such vulnerability.

Consequently, new studies and new tools have emerged to open this black box and provide a more rational understanding of the most relevant factors in

the decision-making of these models. These studies and tools provide what has been conventionally called the explainability of the model. Thus, the most recent studies dealing with intrusion detection using ML began to address not only the performance of the model, but also its explainability, allowing the analyst to verify the reasonableness of the relationships between the attributes abstracted by the model in the classification activity. **Figure 3** illustrates this procedure.

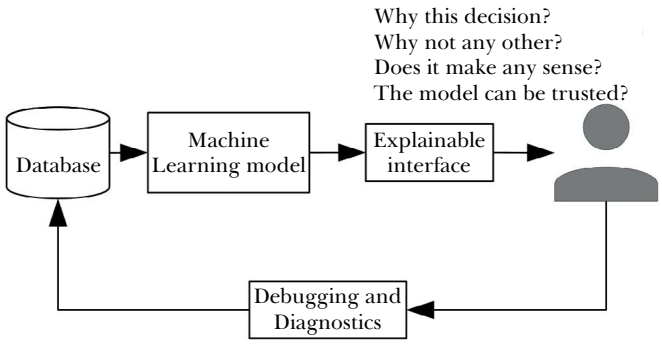

Fig. 3 – Explainable interface. **Source: Adapted from [30].**

6. Studies on IDS

Due to the importance of this subject, several review articles on intrusion detection using ML have been published. Some, with an outstanding number of citations [16, 31], synthesize most of the review objectives, in addition to offering a relevant theoretical framework and contributing to improvements in the dissemination and analysis of results. Additionally, Buczak and Guven [16] highlighted the variety of ML techniques available for application in IDS, showing how complex it is to establish which of them would be the most appropriate, considering the type of attack the system should detect. Another important contribution, presented by Khraisat *et al.* [31], is the study of evasion techniques used by attackers, one of the main challenges in intrusion detection research. Although such aspects are essential, these review articles do not include works of analysis through explainability, which can only be found in review articles from different areas of cybernetics. A second gap refers to the inclusion of studies that address the potential that IDS models have in generating false positives in real applications. Thus, this research seeks to complement the existing review articles through the inclusion and analysis of these two topics. Thus, subsection 6.1 lists the works that use ML, favoring the performance of the algorithm; however, they do not present a critical analysis of other relevant factors, such as, for example, the impacts related to FPR, or even the explainability of the model. These factors were analyzed by the studies in subsections 6.2 and 6.3, respectively. Finally, subsection 6.4 presents a comparison between these studies.

6.1. Studies with a focus on performance

Buczak and Guven [16] gathered works that seek to develop IDS using signature and anomaly detection. The latter has aroused a greater interest of the scientific community for employing classical techniques of ML and data mining. Studies that used public databases and presented some performance information are also of greater interest. One should consider that, in DARPA and derivative bases (with the exception of NSL-KDD), it is common to obtain results with very high performance, outside the context of real applications. This is due to a large number of repeated records, as well as a predominance of easily classified attacks [19]. On the other hand, Khraisat *et al.* [31] conducted a more current research in the area of intrusion detection, including studies that used more recent bases, especially the CIC-IDS 2017. The most relevant works in these studies are shown in **Table 1**, and are briefly described in subsection 6.1.

Tab. 1 – Studies cited in research articles.

Research Articles	Relevant Studies
Buczak and Guven [16]	Fan *et al.* [32] Hu, Liao and Vemuri [33] Bivens *et al.* [34] Kruegel *et al.* [35] Shon and Moon [36] Tajbakhsh, Rahmati and Mirzaei [37] Amor, Benferhat and Elouedi [38]
Khraisat *et al.* [31]	Chen, Hsu and Shen [39] Adebowale, Idowu and Amarachi [40] Thaseen and Kumar [41] Ustebay, Turgut and Aydin [42]

Source: Prepared by the authors.

Fan *et al.* [32] used a rule-making inductive learning technique called RIPPER (*Repeated Incremental*

Pruning to Produce Error Reduction). The authors developed an artificial anomaly generator (attacks) to improve the generalization ability of the classifier. The 1998 DARPA database was used, obtaining a recall of 94%, with a FPR of 2%. Still on the same basis, Hu, Liao and Vemuri [33] used a variation of the support vector machines, called RSVM (*Robust Support Vector Machine*). This technique obtains an average of discriminant hyperplanes, smoothing the classification frontier, in addition to obtaining the regularization parameter and the classifier automatically. Good performance was obtained even in the presence of noise (although some training base labels are incorrect): 75% recall without false alarms, and 100% recall with FRP of 3%.

Bivens *et al*. [34] used SOM (*Self-Organizing Map*) to learn the normal behavior of traffic, together with MLP, to classify intrusions on the DARPA 1999 basis. The authors sought to classify attacks into different categories and reported having obtained 100% of TNR (True Negative Rate) and FPR of 76%. This information is somewhat inconsistent, since a TNR of 100% implies a FPR equal to zero. In fact, what the authors call FPR is the proportion of attacks misclassified in another type of attack. On the same basis, *Kruegel et al*. [35] used Bayesian networks, obtaining a FPR slightly greater than 0.2% and a recall of 100%. On the other hand, Shon and Moon [36] used an anomaly detection technique called *Enhanced* SVM to detect attacks in the 1999 DARPA database. This technique was derived from *One-Class* SVM and *Soft-Margin* SVM. For a more realistic application, the base imbalance was increased, consisting of 1% to 1.5% of attacks and 98.5% to 99% of normal traffic. A false negative rate (FNR) of 27.27% was obtained, with a FPR of 10.2%. Although the recall was not directly provided, it is possible to derive it from the FNR, being, therefore, 72.73%.

Tajbakhsh, Rahmati, and Mirzaei [37] used Fuzzy Association Rule Mining to find patterns in the relationships of the 1999 KDD base attributes. The article highlights some benefits of the technique, such as the creation of humanly interpretable rules; however, it obtained a FPR of 13% to a recall of 100%. Still on the same basis, Amor, Benferhat and Elouedi [38] used the Naive Bayes (NB) algorithm, reporting a recall of 89% and a TNR of 98%. Although FPR has not been reported, it can be inferred, based on TNR, that it is 2%.

Chen, Hsu and Shen [39] extracted system call data from the 1998 DARPA database, coding it with the tf-idf (*term frequency – inverse document frequency*) method, to then train an SVM classifier. A 100% recall was obtained with FPR of 8.53%.

Adebowale, Idowu and Amarachi [40] used SVM, MLP, NB and Decision Trees (DT) for the detection of intrusion in the NSL-KDD base, obtaining the accuracies of 97.3%, 95.8%, 89.6% and 99.6%, respectively. The author used Weka 3.6.7 (Waikato *Environment for Knowledge Analysis*), a software developed in Java by the University of Waikato, New Zealand, which incorporates various ML techniques. It was not informed if the hyperparameters of the models were the default values, or if there was any adjustment to obtain such accuracy. These were raised via 10-fold cross-validation on the "full NSL-KDD" base. Although the NSL-KDD base is provided in three sets, 20% training (only twenty percent of the samples in the training set), training and testing, the author does not clarify what the "full NSL-KDD" is. One possibility, therefore, is that he joined the training and testing parts on a single basis and applied cross-validation, only to train and evaluate the models performance (and not to adjust hyperparameters). Accuracies obtained were well above those raised by Tavallaee *et al*. [19], who were the creators of the NSL-KDD base. Probably, the reason for this was that, in this last article, the training base used was 20% of the training, and the accuracy was raised in the test set. Adebowale, Idowu and Amarachi [40] also provided the detection and FPR indexes: the SVM obtained a recall of 95.9% at an FPR of 1.4%; the MLP, NB and DT obtained, respectively, a recall of 95.9% at an FPR of 4.4%, a recall of 87.7% at an FPR of 8.8%, and a recall of 99.6% at an FPR of 0.4%.

Thaseen and Kumar [41] used the NSL-KDD base to test all tree-based classifiers contained in the Weka software, namely: ADTree, C4.5, J48graft, LADTree, NBTree, RandomTree, RandomForest and REPTree.

A preprocessing was performed in the database, where the continuous attributes were discretized by a supervised attribute filtering technique called *Discretize*. In addition, there was a reduction from 41 to only 8 attributes, obtained via CFS (*Correlation Feature Selection*) analysis. This is a method of selecting attributes that favors those with higher correlation with the output variable and, at the same time, lower correlation with the other attributes. Although there was no adjustment of hyperparameters, overall the classifiers obtained high accuracy in the test set. NBTree was the one that obtained the best performance with a recall of 97.8% to an FPR of 4.8%.

Ustebay, Turgut and Aydin [42] obtained a considerable reduction in the attributes of CIC-IDS 2017 database examples, using the recursive elimination of attributes with the *Random Forest (RF)* algorithm. Thus, the total number of attributes was reduced from 81 to only four, selected according to their importance. The authors raised a relevant point by stating that the attributes *Flow_ID*, *Source_IP*, *Destination_IP*, *Timestamp* and *External_IP* were not used in the training, that is, none of them were present in the four selected. This is an indication that the classifier may perform reasonably closely when applied to an actual network, as such attributes are specific to the CIC-IDS 2017 base and should not be considered in the classification task. However, there was no critical analysis of how general the meaning of each of the four selected attributes is, i.e., whether they remain the same in real networks. After the selection of attributes, the base was partitioned into 80% for training and 20% for testing. Then, a three-layer MLP with activation functions of the ReLu type was used, with the exception of the output layer, whose activation function was Adaptive Moment Estimation. By analyzing the ROC curve, it is possible to infer an 18% FPR to a 100% recall.

There are also studies that used recurrent neural networks in the intrusion detection task. Le, Kim and Kim [43] used the sequence prediction algorithm LSTM (*Long Short Term Memory*) to perform the classification of attacks in DoS, Scan, U2R and R2L. The database used was KDD 1999, and the result obtained was a recall of 98.95%, with 9.98% FPR. However, the authors do not detail the application method of the algorithm.

Xu *et al*. [44] performed a similar study to that of Le, Kim and Kim [43], but replacing the LSTM algorithm with a Recurrent Neural Network of sequence prediction called GRU (*Gated Recurrent Units*). The authors suggested that GRU is superior to LSTM to classify attacks, and exemplified such comparison through an isolated application of both algorithms in the KDD 1999 and NSL-KDD databases. In both bases, GRU was actually superior to LSTM, although there is no theoretical support to generalize such superiority in other applications. In addition, such algorithms were used in conjunction with a MLP Neural Network. The best results, obtained with the GRU, were a recall of 99.42%, with a FPR of 0.05%, for the KDD 1999 base. In the case of the NSL-KDD database, the recall was 99.31%, with a FPR of 0.84%. Although Le, Kim and Kim [43] have been cited in the bibliographic references of Xu *et al*. [44], at no time do the authors emphasize the fact they obtained an extremely reduced FPR compared to the first study. In addition, again, there is no detailing of the methodology for algorithm application, where the sequences of attributes used could be presented, since it is a sequential prediction algorithm.

Papamartzivanos [45] developed a rule-inducing methodology based on a combination of genetic algorithms (GA) and decision trees. This method, called *Dendron*, aims to evolve a population of decision trees, resulting in a set of rules for detection and classification of attacks. According to the author, this methodology allows the development of a signature-based IDS with a balanced performance, i.e., with similar accuracy in all classes (including the rarest) in which network traffic is classified. Another advantage of this method is that, as it used decision trees, the rules generated for classification are humanly intelligible, facilitating decision making by experts in relation to countermeasures. The model was tested in the KDD 1999, NSL-KDD and UNSW-NB15 databases, obtaining, respectively, the following performance metrics: recall of 98.24%,

with an FPR of 0.75%; recall of 95.97%, with an FPR of 1.08%; and recall of 63.76%, with an FPR of 2.61%.

Still in this article, Papamartzivanos [45] based the study on a methodology developed by Raina *et al.* [46] called STL (*Self-Taught Learning*), with the objective of making IDS self-adaptive, allowing it to maintain performance even in the occurrence of significant changes in the environment where it operates. These changes can have different causes, such as the insertion and removal of network assets, including IoT equipment, software updates and the emergence of new attacks. Therefore, unlike common (static) signature IDS, there is no need for manual updates to adapt to these changes. To demonstrate this capability, the author trained an IDS on an initial base containing 10% of normal traffic from the 1999 KDD base and a small portion of attacks from each class of the same base. The algorithm used was a DSAN network (*Deep Sparse Autoencoder Network*) in conjunction with Softmax regression. The remainder of the 1999 KDD database was used to generate 100 different environments through random sampling. Each environment could contain, or not contain, classes or instances belonging to the initial base; therefore, depending on these differences, a given environment could be little or very distinct from the initial base. Thus, starting from the trained IDS, performance data were obtained in these varied environments in two ways: the first, keeping the IDS static, i.e., without changing it after the initial training; and the second, using STL to test the ability of the classifier to adapt to changes. The static IDS obtained, on average (considering 100 different environments), a recall of 38.54%, well below 60.34%, which was obtained by the self-adaptive IDS. A discordant aspect in this study is that the experiment was conducted in an ML-based IDS, using DSAN and Softmax, and the author argues that this technique allows for making a signature-based IDS self-adaptive.

Stefanova [47] developed an *Intrusion Detection and Prevention Systems* (IDPS), which has two layers: the first, responsible for performing the detection through ML, and the second, responsible for actively preventing the detected attack, through Reinforcement Learning. In this case, the author applies an approximation of *Q-learning* in the context of Game Theory, where the attacker and the defender (of the data network) participate in the same game whose solution is an optimal balance point, from the point of view of defense. The database used was NSL-KDD. Regarding the first layer, there was a selection of attributes using information gain, reducing them from 41 to 30. The classification was performed by a RF, obtaining an FPR of approximately 0.16% and a recall of 99.9%. These values were obtained by inspection of the ROC curve present in the study.

6.2. Studies focused on reducing FPR

The following studies addressed the problem of FPR, which is one of the main obstacles in the implementation of an IDS in real networks, especially in those with high traffic density.

Subba, Biswas and Karmakar [48] proposed a method of reducing the FPR in signature IDS, whose main idea is to scan all assets belonging to the local network to be protected and to list all existing vulnerabilities, creating a "vulnerability profile". Thus, all alarms related to attacks exploiting vulnerabilities that do not exist in this profile are discarded, considerably reducing the FPR, without impacting the recall. The experiment used the Snort IDS, whose signature database was in the VRT-certified (*Vulnerability Research Team*) Snort V2.8 version, in the default configuration, with all signatures enabled. Initially, performance data of the IDS Snort were collected in the DARPA 1999 database and, after that, the proposed technique was applied to minimize FPR. The author did not make clear how much FPR was minimized. On the other hand, he reported that, in the case of critical vulnerabilities, the accuracy of the detector increased from 83.24% to 97.85%, without changing the recall, which remained at 37.47%. In the case of non-critical vulnerabilities, the accuracy increased from 71.31% to 95.56%, with a small degradation in recall (from 35.67% to 32.43%). Thus, since the accuracy improved without

significantly changing the recall, one may say there was a decrease in the FPR.

Another article that also used Snort to study the false positive problem was Tjhai *et al.* [49]. In this case, IDS has been installed to monitor inbound and outbound traffic on the University of Plymouth (UK) web server. One of the problems with this approach is that only events that generated alarms were analyzed. This means there was no knowledge about the number of true or false negatives. Thus, it was agreed to call the True Positive Rate (TPR) the proportion of true alarms in relation to the total amount of alarms. In fact, this ratio denotes the precision of the IDS, not the TPR. Similarly, the proportion of false alarms in relation to the total number of alarms was conventionally called FPR, which, in fact, is the complement of precision. Despite this, the data presented were sufficient to show the extent of the problem of false positives, since these were considerably more numerous in relation to true alarms, in this case, in an approximate proportion of 24:1. Another aspect presented was a third category of alarms (in addition to true and false positive), called irrelevant positive, which, in this article (and in Subba, Biswas and Karmakar [48]), was included in the category of false positive. An irrelevant positive is an attack known to be unsuccessful, as it exploits vulnerabilities that do not exist in the network in question. For example, an attack that exploits a particular Windows OS specific vulnerability is an irrelevant positive if all machines connected to the network only run Linux OS. In summary, the vast majority of false alarms were noticed to be generated by three types of signatures. Thus, through the analysis of an expert, adjustments were made to the rules related to these signatures, resulting in a decrease in the proportion of false positives (in relation to true positives) from 95.5% to 86.8%.

Zohrevand and Glässer [50] analyzed several studies aimed at decreasing FPR of anomaly-based IDSs to acceptable levels. According to the authors, considerable attention has been given only to obtaining a model trained in anomaly detection, neglecting aspects related to the analysis of decisions made by these models, especially analyses aimed at mitigating false alarms, such as the "anomaly score", which measures the level of difference regarding normal events, and adjustment of the decision threshold. Thus, different ways of computing the anomaly score were presented, addressing probabilities, correlations and similarities between anomalies. Unfortunately, the survey did not present numerical data for comparison purposes.

6.3. Studies with a focus on model explainability

In this section, studies related to the explainability of algorithms applied in cyberattack detection will be presented. Usually, in this case, there are two most common approaches: a broad analysis of the attributes, where the most important ones are shown in general, and a more specific analysis, where, for a given classification (or group of classifications), the attributes that most influenced it are verified. In addition, performance data is no longer as relevant as in the studies cited in subsection 6.1. Thus, studies on explainability are more focused on the interpretation of attributes and how reasonable is the importance given to them by the algorithm in a given classification task. Marino, Wickramasinghe and Manic [30] analyzed examples misclassified by the model through a method called Adversarial Approach. The idea of this technique is to verify the smallest possible change in attributes necessary to correct a wrong classification. This method works for any type of model (linear, neural network, SVM, etc.) as long as the loss function – typically the cross-entropy for the classification case – has a gradient defined in relation to the input attributes. One should note that a *hacker* can use this same method to do the opposite, i.e., to determine which is the smallest change in the attributes of an attack (which has been detected) necessary for the model to consider it as legitimate traffic. However, there are correlations between attack attributes that impose restrictions capable of preventing changes independently. Thus, the smallest theoretical change can result in an unfeasible attack, since the technique of Adversarial Approach disregards interdependence relations between attributes.

Figure 4 elucidates an Adversary Approach application referring to a particular case of the

NSL-KDD database. These are the normal (legitimate) traffic samples wrongly classified as DoS by an already trained neural network model, with an accuracy of 95.5%. On the vertical axis, there are the attributes, and on the horizontal axis, the minimum values on average that should be subtracted from these attributes so the model would correctly classify these samples. This means that, by making such minimal changes in attributes, samples erroneously classified as DoS would be classified correctly in normal traffic. As a result, one can explain the wrong classification. One may see that a relatively high number of connections occurred for the same *host* (*count*) and for the same destination address (*dst_host_count*), and these connections had a short duration (*duration*). In addition, there were few operations performed as *root* on these connections (*num_root*), as well as a low percentage of samples that were able to log in successfully (*logged_in*, *is_guest_login*). These characteristics are common in denial-of-service attacks, hence the reason for misclassification. It would be up to the analyst to verify if there was a label error at the base, or if there was only a coincidence, or even whether there is any logical reason for records labeled as normal to have such attributes.

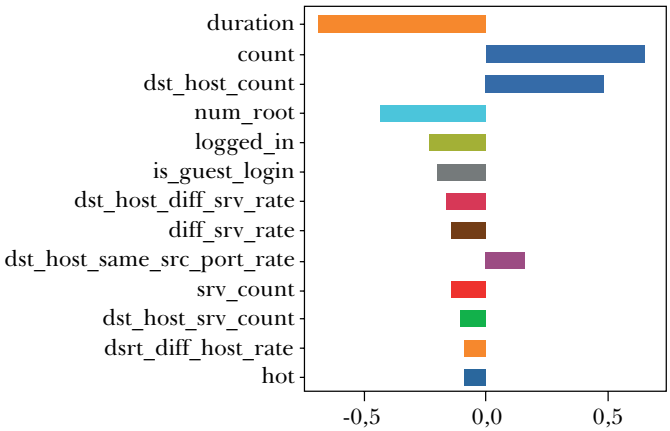

Fig. 4 – Normal samples misclassified as DoS. **Source: [30].**

Reyes *et al.* [51] used ML in the detection of intrusion in Wi-Fi networks, constant in the AWID2 base. Detection occurs in two stages: the first, through the RF, classifying the events in normal, flood, and personification/injection; and the second stage separates the personification of the injection, via NB. This model was conceived as a consequence of a previous study, where the authors, when classifying the attacks in a single stage, realized that many samples of the personification class were classified as injection and vice versa. Considering only the task of detecting the first stage, that is, classifying it as normal or attack, the proposed model obtained an accuracy of 99.41%, a recall of 94.1% and an FPR of 0.13%. Additionally, the authors dedicated a section to the analysis of attributes using the SHAP library (*SHapley Additive exPlanations*) [52], which offers a variety of tools for analysis of global and local explainability of the model, including graphically. However, this study does not present a domain explanation, i.e., the meaning of each attribute (or, at least, the main ones) and whether or not the behavior of the model is coherent, clarified by explainability techniques. Despite this, the authors limited themselves to citing which attributes positively or negatively impact the classification of a given label. However, a more in-depth analysis of the coherence of these impacts would be necessary, which is only possible with an understanding of the meaning of the attributes involved.

Wang *et al.* [53] also used SHAP to explain the decisions of an IDS, in addition to claiming pioneering in the application of this technique of explainability in intrusion detection. The NSL-KDD database was used to train two distinct neural network models: one-against-all and multiclass, which obtained, respectively: 80.6% accuracy, 80.6% recall and 19.4% FPR; 80.3% accuracy, 80.3% recall and 19.7% FPR. In these models, a local explainability analysis was made regarding DoS attacks of the Neptune type, which attempts to saturate a server by sending a high number of SYN packets on all ports. These attacks therefore have a high rate of connections with SYN error and, using local SHAP analysis, it was found the one-against-all and multiclass models classified Neptune attacks in DoS with, on average, 93% and 89% certainty, respectively. However, the attributes that guided this classification were quite different. In fact, the one-against-all model used more reasonable attributes and directly related to Neptune. Such attributes indicated high SYN connection error

rates, demonstrating that this feature was primarily responsible for the classification. The multiclass model did not predominantly use attributes related to this aspect of SYN error. Therefore, the authors concluded that the multiclass model does not provide consistent reasons for an expert's confidence in the results. On the other hand, there was no critical analysis of the attributes that guided the classification made by the multiclass model, even though they were not directly related to Neptune. A global analysis of the importance of attributes using SHAP was also presented. **Figure 5** shows 20 of the 41 most important attributes for DoS classification for the one-against-all model case. Each line in the figure contains an attribute and all examples of the test set in the form of points, whose color varies from blue to red, representing the lowest and highest values of the attribute, respectively. The location of the points on the horizontal axis denotes how much the respective attribute contributed for or against the classification in DoS.

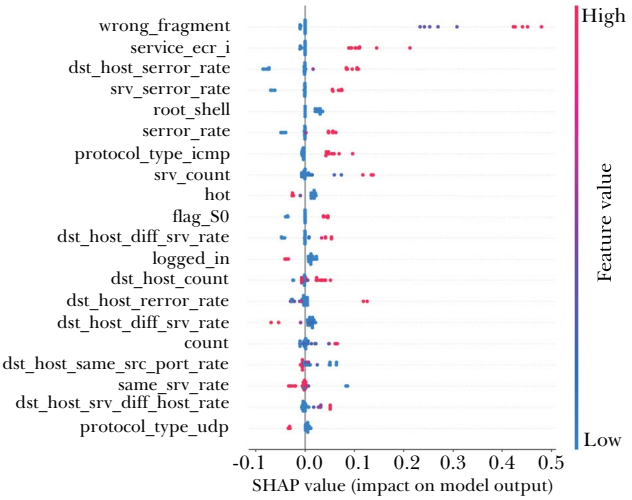

Fig. 5 – 20 most relevant attributes in the DoS classification by the one-vs-all model. **Source: [53].**

Table 2 lists the four most relevant attributes of **Figure 5** with DoS-type attacks, demonstrating that, in fact, such attributes are quite pertinent to an effective DoS classification.

Tab. 2 – DoS attack types.

Type	Description	Attribute
Land	This attack locks SunOS 4.1 by sending a masked TCP SYN packet, with the same source address as the destination.	srv_serror_rate dst_host_serror_rate
Neptune	Also known as SYN *flood* or *half open* attack, it floods web servers with masked TCP SYN packets, depleting memory. Consequently, new connections will be rejected until expiration of the time of these connections opened by the masked packages.	
PoD	Known as "*Ping of Death*," this attack sends fragmented IP packets so that when they are rebuilt on the target *host*, they result in an IP packet over 65,535 bytes in size, which is the maximum allowed. This causes old operating systems to crash.	*wrong_fragment*
Teardrop	This attack sends fragmented IP packets that are impossible to rebuild on the target *host*, because there are overlaps between the fragments, causing some operating systems to crash.	
Smurf	This attack sends ICMP packets of the *Echo Request* type, known as ping, to all *hosts* present on the victim's network, via *broadcast*. As the source address field of these packets was masked with the IP address of the victim, it receives an excessive amount of *Echo Reply* packets, coming in response from all *hosts* contained in the *broadcast* address.	*service_ecr_i*

Source: Adapted from [53].

A slightly different approach was presented in Mahdavifar and Ghorbani [54]. In this article, the authors initially trained a DNN neural network to detect cyberattacks and later derived from this network an expert model, called DeNNeS (*Deep Embedded Neural Network Expert System*). Expert models are defined as a computer program that performs a task typically performed by an expert person through if-then rules [55]. In this case, these rules were extracted from the DNN model, which is not interpretable, to compose the knowledge base of an expert system, which is interpretable.

For a formal definition of these rules, consider T: $\{X, \mathbf{y}\} \rightarrow \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$ a database with m examples $(\mathbf{x}^{(i)}; y^{(i)})$ where $\mathbf{x}^{(i)}$ represents a vector of attributes of example i, and $y^{(i)}$ the label of the associated class. $\mathbf{x}^{(i)}$ has n attributes $x_k^{(i)}$, as follows: $\mathbf{x}^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}\}$; and $y^{(i)}$ is an integer: $y^{(i)} \in Z$.

Thus, a classification rule r_i, with respect to $(\mathbf{x}^{(i)};y^{(i)})$, is defined as r_i: $P_i \rightarrow Q_i$, where the antecedent of the rule P_i is a combination of $l < n$ attribute values $\{k_1, k_2, \ldots, k_l\}$:

$$P_i = \{x_{k_1}^{(i)} \wedge x_{k_2}^{(i)} \wedge \ldots \wedge x_{k_l}^{(i)}\},$$

And the consequence of the rule $Q_i = \{y^{(i)}\}$ is the class label.

Thus, Mahdavifar and Ghorbani [54] developed extensions of the MIJ (MACIE's *Inference Justification*) algorithm [55], in order to extract these rules from the DNN model. Different rules applied to the same sample may result in different classifications. Thus, to carry out the final classification based on rules, a voting system was established, considering all the rules applicable to the sample in question. In addition, the voting weight of each rule varies according to the strength of the rule (η_{ri}), measured by the product of two factors: confidence (Cf_{ri}) and coverage (Cr_{ri}), something somewhat analogous to the performance indicators precision and recall of classifier models. The first factor is equal to the proportion of examples that obey the rule completely in relation to all that obey the antecedent of the rule. The second is equal to the proportion of examples that obey the rule completely in relation to all that obey the consequent of the rule. Formally, we have:

$$\eta_{r_i} = Cf_{r_i} \cdot Cr_{r_i},$$

$$\text{Where} \quad Cf_{r_i} = \frac{|\{(x^{(i)}, y^{(i)}) \in T \mid P_i \subseteq x^{(i)} eQ = \{y^{(i)}\}\}|}{|\{(x^{(i)}, y^{(i)}) \in T \mid P_i \subseteq \boldsymbol{x}^{(i)}\}|}$$

$$\text{And} \quad Cr_{r_i} = \frac{|\{(x^{(i)}, y^{(i)}) \in T \mid P_i \subseteq x^{(i)} eQ_i = \{y^{(i)}\}\}|}{|\{(x^{(i)}, y^{(i)}) \in T \mid Q_i = \{y^{(i)}\}\}|}.$$

Two databases were used to apply this technique: Phishing Websites, from the University of California Irvine (UCI), and an Android malware database, collected through the Virustotal and Contagio Security Blog websites. On these bases, DNN, DeNNeS and classical ML algorithms were applied. DeNNeS obtained the second best performance, being very close to the DNN model, which obtained the best. In the Phishing Websites database, DeNNeS obtained an accuracy of 97.5% and an FPR of 1.8%, surpassing another rule extraction technique known as JRip, in addition to also surpassing other ML techniques (RF, SVM, KNN and NB). In the malware base, DeNNeS achieved an accuracy of 95.8% and an FPR of 8%; likewise, it surpassed JRip and other ML techniques.

6.4. Comparison between related studies

Table 3 presents the performance data of the studies contained in subsection 6.1. As some studies in subsection 6.3 also provided performance data [51, 53, 54], they were included at the end of this table. The purpose of this comparison of results is to show that, in general, FPR is not negligible, and may become an important problem in cases of high and unbalanced traffic in favor of legitimate events. In addition, there is a very large variation of these values, including in studies that used the same database. As practically all the articles addressed do not provide in detail the methodology and all the data necessary for the replication of the experiment, there is no way to guarantee a completely fair comparison.

Tab. 3 – Performance obtained in intrusion detection articles using ML.

Article	Database	Technique	Recall (%)	FPR (%)	Accuracy (%)
Fan *et al.* [32]	DARPA 1998	RIPPER	94	2	-
Hu, Liao and Vemuri [33]	DARPA 1998	RSVM	100	3	-
Bivens *et al.*[34]	DARPA 1999	SOM/ MLP	-	76	-
Kruegel *et al.* [35]	DARPA 1999	Bayesian Networks	100	0.2	-
Shon and Moon [36]	DARPA 1999	Improved SVM	72.73	10.2	-
Tajbakhsh, Rahmati and Mirzaei [37]	KDD 1999	Fuzzy Rules	100	13	-
Amor, Benferhat and Elouedi [38]	KDD 1999	NB	89	2	-
Chen, Hsu and Shen [39]	DARPA 1998	SVM	100	8.53	-
Adebowale, Idowu and Amarachi [40]	NSL-KDD	SVM	95.9	1.4	97.3
		MLP	95.9	4.4	95.8
		NB	87.7	8.8	89.6
		DT	99.6	0.4	99.6
Thaseen and Kumar [41]	NSL-KDD	NBTree	97.8	4.8	-
Ustebay, Turgut and Aydin [42]	CIC-IDS 2017	MLP	100	18	-
Le, Kim and Kim [43]	KDD 1999	LSTM	98.95	9.98	-
Xu *et al.* [44]	KDD 1999	GRU/ MLP	99.42	0.05	-
	NSL-KDD		99.31	0.84	-
Papamartzivanos [45]	KDD 1999	GA/DT	98.24	0.75	98.85
	NSL-KDD		95.97	1.08	97.55
	UNSW-NB15		63.76	2.61	84.33

(continue...)

Tab. 3 - Continuation

Article	Database	Technique	Recall (%)	FPR (%)	Accuracy (%)
Stefanova [47]	NSL-KDD	RF	99.9	0.16	-
Reyes *et al*. [51]	AWID	RF	94.1	0.13	99.41
Wang *et al*. [53].	NSL-KDD	MLP	80.6	19.4	80.6
Mahdavifar and Ghorbani [54]	Private	DNN	-	1.8	97.5

Source: Prepared by the authors.

Also in the **Table 3**, it is verified that some studies obtained an FPR of less than 1%, with a recall of 99%. This high performance occurred once in the 1999 DARPA base, once in the 1999 KDD and three times in the NSL-KDD. The algorithms used were Bayesian Networks, DT, RF and a combination of GRU with MLP. Without doing an explainability analysis on these models, there is no way to know whether such performances will be maintained in real applications. Although the development of the methods and the performances achieved were of high relevance, one must also understand whether the attributes used for detection represent general relationships that are not restricted to the databases used.

Even if such performances are maintained in real applications, even the lowest FPR obtained, in this case, 0.05% for the 1999 KDD base, can generate a considerable amount of false positives in highly fast and unbalanced networks, i.e., with a much higher amount of benign traffic than those of malicious traffic. It should also be noted that the KDD 1999 database, in addition to being old, has imperfections that facilitate the task of classifying the model [19]. In addition, the article that employed it [44] is much more recent than the date of its construction. This all supports overrated performance. On the other hand, one should see the 100% recall obtained in the CIC-IDS 2017 base was only possible with an FPR of 18%, well above 0.05%. In the UNSW-NB15 base, made available in 2015, the FPR was 2.61%, however, with a low recall (63.76%). Certainly, an adjustment in the detector threshold to increase this recall would cause a considerable increase in FPR. However, the research must be expanded to get more data about the performances obtained in more recent databases.

Table 4 summarizes the studies that presented other proposals, in addition to only obtaining a high performance index. With the exception of [45], contained in subsection 6.1, all studies in this table were extracted from subsection 6.2 and 6.3. Therefore, in general, the proposals of these studies are related to strategies to treat the occurrence of false positives, or to aspects of explainability, to obtain knowledge about which combinations of attribute values represent an attack, and whether there is coherence in this combination.

Table 4 – Articles on intrusion detection with objectives other than performance.

Article	Databases	Technique	Objective	Result
Papamartzivanos [45]	KDD 1999 and NSL-KDD	STL	To develop a self-adaptive signature-based IDS.	The self-adapting IDS overcame static IDS in diverse environments.
Subba, Biswas and Karmakar [48]	DARPA 1999	Vulnerability-based detection filtering.	Reduction of FPR in signature IDS.	Increased accuracy, without considerable degradation in recall.
Tjhai *et al*. [49]	Private	Adjustments to IDS rules related to signatures that most generate false positives.	Reduction of FPR in signature IDS.	Decrease in the proportion of false positives (in relation to the total amount of alarms) from 95.5% to 86.8%.
Marino, Wickramasinghe and Manic [30]	NSL-KDD	Approach Adversarial	Understanding the reason for false positive occurrence in a group of examples, of normal traffic, classified in DoS attack.	Obtained the attributes that most contributed to the erroneous classification, as well as how much such attributes should be changed for a correct classification.
Wang *et al*. [53]	NSL-KDD	SHAP	Check which attributes were most used by the model in classifying Neptune attacks in DoS, and whether there is consistency in this attribute choice.	Obtained the attributes that most contributed to Netpune's classification in DoS. There was more coherence in the one-vs-all MLP model than in the multiclass MLP.
Mahdavifar and Ghorbani [54]	Private	Extraction of model rules	Replacing an uninterpretable DNN model with an interpretable one.	Obtained an expert model, by extracting rules from the DNN model, with little degradation in performance.

Source: Prepared by the authors.

7. Conclusion

ML has shown to be a promising research field in intrusion detection activity, due to the high performance obtained in simulated databases. However, there is no guarantee that the same level of performance will be achieved in real applications. Therefore, studies are needed to verify the level of similarity between simulated bases and real traffic. An obstacle that arises is the possibility of exposing sensitive information contained in this type of traffic by making it public for studies.

Another approach to analyze possibilities for maintaining performance in real applications would be through explainability. This technique allows us to verify the factors considered most important by complex ML models in the task of detecting intrusion. Thus, the study of these factors can conclude whether they really represent general characteristics, related to attacks, or benign traffic, regardless of whether the environment is real or simulated.

Finally, the problem of the occurrence of false positives was also verified, which can lead to the neglect of real attacks, camouflaged by an excessive amount of alarms. IDSs using ML are more susceptible to these circumstances, which in general has been an obstacle in the implementation of this technology in intrusion detection.

References

[1] ALYASIRI, H. Developing Efficient and Effective Intrusion Detection System using Evolutionary Computation. Thesis (Computer Science PhD) – University of York, Heslington, 2018.

[2] AKSU D.; AYDIN M. A. Detecting Port Scan Attempts with Comparative Analysis of Deep Learning and Support Vector Machine Algorithms. In International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT). Piscataway: IEEE, 2018. p. 77–80. http://dx.doi.org/10.1109/IBIGDELFT.2018.8625370.

[3] HACKER tries to poison water supply of Florida city. BBC News, 8 fev. 2021. Available in: https://www.bbc.com/news/world-us-canada-55989843. Accessed on: February 2021.

[4] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION – ISO; INTERNATIONAL ELECTROTECHNICAL COMMISSION – IEC. Joint Technical Committee ISO/IEC JTC 1/SC 127. ISO/IEC 27032:2012(en) Information technology — Security techniques — Guidelines for cybersecurity. Whashington, DC: ISO: IEC, 2012. Available in: https://www.iso.org/obp/ui/#iso:std:iso-iec:27032:ed-1:v1:en. Accessed on: Sept. 24, 2021.

[5] AMOROSO, E. G.; AMOROSO, M. E. From CIA to APT: An Introduction to Cyber Security. 2017.

[6] SCARFONE, K. A.; MELL, P. M. NIST Special Publication 800-94 – Guide to Intrusion Detection and Prevention Systems (IDPS): Recommendations of the National Institute of Standards and Technology. Gaithersburg, MD: National Institute of Standards and Technology, 2007.

[7] INDIAN CYBER SECURITY SOLUTIONS. Intrusion Detection System and its Detailed Working Function. ICSS, 2021. Available in: https://indiancybersecuritysolutions.com/intrusion-detection-system-working-function. Accessed on: Aug, 17, 2021.

[8] THOMA, M. Receiver Operating Characteristic (ROC) curve with False Positive Rate and True Positive Rate. Wikimedia Commons, 2018. Available in: https://commons.wikimedia.org/w/index.php?title=File:Roc-draft-xkcd-style.svg&oldid=491003296. Accessed on: Jan. 7, 2021.

[9] XIN, Y.; KONG, L.; LIU, Z.; CHEN, Y.; LI, Y.; ZHU, H. et al. Machine Learning and Deep Learning Methods for Cybersecurity. IEEE Access, v. 6, p. 35365–35381, 2018. https://doi.org/10.1109/ACCESS.2018.2836950.

[10] KELLEHER, J. D.; NAMEE B. M.; D'ARCY, A. Fundamentals of Machine Learning for Predictive Data Analytics. 2. ed. Cambridge: MIT Press, 2020.

[11] MUELLER, J. P.; MASSARON, L. Machine Learning for Dummies. Hoboken: John Wiley & Sons, 2016.

[12] GHAHRAMANI, Z. Unsupervised Learning. In BOUSQUET, O.; VON LUXBURG, U.; RÄTSCH, G. (ed.). Advanced Lectures on Machine Learning. ML Summer Schools 2003. Amsterdam: Springer, 2003. https://doi.org/10.1007/978-3-540-28650-9_5.

[13] EVSUKOFF, A. G. Inteligência Computacional: Fundamentos e Aplicações. Rio de Janeiro: E-papers, 2020.

[14] VANDERPLAS, J. Python Data Science Handbook: Essential Tools for Working with Data. Sebastopol: O'Reilly, 2016.
[15] RASCHKA S.; MIRJALILI, V. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow. 2. ed. Birmingham: Packt Publishing, 2017.
[16] BUCZAK, A. L.; GUVEN, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials, v. 18, n. 2, p. 1153–1176, 2016. https://doi.org/10.1109/COMST.2015.2494502.
[17] DARPA Intrusion Detection Evaluation Dataset. MIT Lincoln Laboratory, 1998/1999. Available in: https://www.ll.mit.edu/r-d/datasets. Accessed on: Aug. 27, 2021.
[18] DATA – KDD Cup 1999: Computer network intrusion detection. SIGKDD, 1999. Available in: https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data. Accessed on: Aug. 27, 2021.
[19] TAVALLAEE, M.; BAGHERI, E.; LU, W.; GHORBANI, A. A. A detailed analysis of the KDD CUP 99 data set. In IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). Piscataway: IEEE, 2009. p. 1–6. https://doi.org/10.1109/CISDA.2009.5356528.
[20] NSL-KDD DATASET. Canadian Institute for Cybersecurity. University of New Brunswick, 2009. Available in: https://www.unb.ca/cic/datasets/nsl.html. Accessed on: Aug. 27, 2021.
[21] CTU-13 DATASET. A Labeled Dataset with Botnet, Normal and Background Traffic. Stratosphere Lab, 2011. Available in: https://www.stratosphereips.org/datasets-ctu13. Accessed on: Aug. 29, 2021.
[22] DELPLACE, A.; HERMOSO S.; ANANDITA, K. Cyber Attack Detection thanks to Machine Learning Algorithms. arXiv preprint, 2001.06309, 2020.
[23] DATASETS. Canadian Institute for Cibersecurity, University of New Brunswick, 2020. Available in: https://www.unb.ca/cic/datasets/index.html. Accessed on: Aug. 29, 2022.
[24] WIRELESS DATASETS. University of the Aegean, 2021. Available in: https://icsdweb.aegean.gr/awid/download-dataset. Accessed on: Dec. 23, 2021.
[25] KOLIAS, C.; KAMBOURAKIS, G.; STAVROU, A.; GRITZALIS, S. Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset. IEEE Communications Surveys & Tutorials, v. 18, n. 1, p. 184–208, 2016. https://doi.org/10.1109/COMST.2015.2402161.
[26] MOUSTAFA, N.; SLAY, J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Military Communications and Information Systems Conference (MilCIS). Piscataway: IEEE, 2015. p. 1–6. https://doi.org/10.1109/MilCIS.2015.7348942.
[27] GARCIA, S.; PARMISANO, A.; ERQUIAGA, M. J. IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0). Zenodo. Stratosphere Lab, 2020. Available in: https://www.stratosphereips.org/datasets-iot23. Accessed on: Dec. 23, 2021.
[28] USB-IDS-1. Università degli Studi del Sannio di Benevento, 2021. Available in: http://idsdata.ding.unisannio.it/datasets.html. Accessed on: Dec. 25, 2021.
[29] NISIOTI, A.; MYLONAS, A.; YOO, P. D.; KATOS, V. From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods. IEEE Communications Surveys & Tutorials, v. 20, n. 4, p. 3369–3388, 2018. https://doi.org/10.1109/COMST.2018.2854724.
[30] MARINO, D. L.; WICKRAMASINGHE, C. S.; MANIC, M. An Adversarial Approach for Explainable AI in Intrusion Detection Systems. In 44th Annual Conference of the IEEE Industrial Electronics Society. Piscataway: IEEE, 2018. p. 3237–3243. https://doi.org/10.1109/IECON.2018.8591457.
[31] KHRAISAT, A.; GONDAL, I.; VAMPLEW, P.; KAMRUZZAMAN, J. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, v. 2, n. 20, 2019. https://doi.org/10.1186/s42400-019-0038-7.
[32] FAN, W.; MILLER, M.; STOLFO, S. J.; LEE W.; CHAN, P. K. Using artificial anomalies to detect unknown and known network intrusions. In IEEE International Conference on Data Mining. Piscataway: IEEE, 2001. p. 123–130. https://doi.org/10.1109/ICDM.2001.989509.
[33] HU, W.; LIAO, Y.; VEMURI, V. R. Robust Support Vector Machines for Anomaly Detection in Computer Security. In International Conference on Machine Learning and Applications (ICMLA). Piscataway: IEEE, 2003. p. 168–174.
[34] BIVENS, A.; PALAGIRI, C.; SMITH, R.; SZYMANSKI, B.; EMBRECHTS, M. Network-Based Intrusion Detection Using Neural Networks. In Intelligent Engineering Systems through Artificial Neural Networks ANNIE-2002, v. 12. New York: ASME Press, 2002. p. 579–584.

[35] KRUEGEL, C.; MUTZ, D.; ROBERTSON, W.; VALEUR, F. Bayesian event classification for intrusion detection. In 19th Annual Computer Security Applications Conference (ACSAC). Piscataway: IEEE, 2003. p. 14–23. https://doi.org/10.1109/CSAC.2003.1254306.

[36] SHON, T.; MOON, J. A hybrid machine learning approach to network anomaly detection. Information Sciences, v. 177, n. 18, p. 3799–3821, 2007. https://doi.org/10.1016/j.ins.2007.03.025.

[37] TAJBAKHSH, A.; RAHMATI, M.; MIRZAEI, A. Intrusion detection using fuzzy association rules. Applied Soft Computing, v. 9, n. 2, p. 462–469, 2009. https://doi.org/10.1016/j.asoc.2008.06.001.

[38] AMOR, N. B.; BENFERHAT S.; ELOUEDI, Z. Naive Bayes vs decision trees in intrusion detection systems. In Proceedings of the 2004 ACM Symposium on Applied Computing, Association for Computing Machinery. New York: ACM Digital Library, 2004. p. 420–424. https://doi.org/10.1145/967900.967989.

[39] CHEN, W. H.; HSU, S. H.; SHEN, H. P. Application of SVM and ANN for intrusion detection. Computers & Operations Research, v. 32, n. 10, p. 2617–2634, 2005. https://doi.org/10.1016/j.cor.2004.03.019

[40] ADEBOWALE, A.; IDOWU, S. A.; AMARACHI, A. A. Comparative study of selected data mining algorithms used for intrusion detection. International Journal of Soft Computing and Engineering, v. 3, n. 3, p. 237–241, 2013.

[41] THASEEN, I. S.; KUMAR, C. A. An analysis of supervised tree based classifiers for intrusion detection system. In International Conference on Pattern Recognition, Informatics and Mobile Engineering. Piscataway: IEEE, 2013. p. 294–299. https://doi.org/10.1109/ICPRIME.2013.6496489.

[42] USTEBAY, S.; TURGUT, Z.; AYDIN, M. A. Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier. In International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT). Piscataway: IEEE, 2018. p. 71–76. https://doi.org/10.1109/IBIGDELFT.2018.8625318.

[43] LE, T. T. H.; KIM, J.; KIM, H. An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization. In International Conference on Platform Technology and Service (PlatCon). Piscataway: IEEE, 2017. p. 1–6. https://doi.org/10.1109/PlatCon.2017.7883684.

[44] XU, C.; SHEN, J.; DU, X.; ZHANG, F. An Intrusion Detection System Using a Deep Neural Network with Gated Recurrent Units. IEEE Access, v. 6, p. 48697–48707, 2018. https://doi.org/10.1109/ACCESS.2018.2867564.

[45] PAPAMARTZIVANOS, D. C. Advanced machine learning methods for network intrusion detection. Tese (Doutorado em Filosofia) – University of the Aegean, Mitilene, 2019.

[46] RAINA, R.; BATTLE, A.; LEE, H.; PACKER B.; NG, A. Y. Self-Taught Learning: Transfer Learning from Unlabeled Data. In Proceedings of the 24th International Conference on Machine Learning, Association for Computing Machinery. New York: ACM Digital Library, 2007. p. 759–766. https://doi.org/10.1145/1273496.1273592.

[47] STEFANOVA, Z. S. Machine Learning Methods for Network Intrusion Detection and Intrusion Prevention Systems. Tese (Doutorado em Filosofia) – University of South Florida, Tampa, 2018.

[48] SUBBA, B.; BISWAS, S.; KARMAKAR, S. False alarm reduction in signature-based IDS: game theory approach. Security and Communication Networks, v. 9, n. 18, p. 4863–4881, 2016. https://doi.org/10.1002/sec.1661.

[49] TJHAI, G. C.; PAPADAKI, M.; FURNELL, S. M.; CLARKE, N. L. Investigating the problem of IDS false alarms: An experimental study using Snort. In Proceedings of The Ifip Tc 11 23rd International Information Security Conference, v. 278. Laxenburg: IFIP, 2008. p. 253–267.

[50] ZOHREVAND, Z.; GLÄSSER, U. Should I Raise The Red Flag? A comprehensive survey of anomaly scoring methods toward mitigating false alarms. arXiv preprint, 1904.06646, 2019.

[51] REYES, A. A.; VACA, F. D.; AGUAYO, G. A. C.; NIYAZ, Q.; DEVABHAKTUNI, V. A Machine Learning Based Two-Stage Wi-Fi Network Intrusion Detection System. Electronics, v. 9, n. 10, p. 1689, 2020. https://doi.org/10.3390/electronics9101689.

[52] LUNDBERG, S. M.; LEE, S. I. A Unified Approach to Interpreting Model Predictions. In 31th Conference on Neural Information Processing Systems. New York: ACM Digital Library, 2017. p. 4768–4777.

[53] WANG, M.; ZHENG, K.; YANG, Y.; WANG, X. An Explainable Machine Learning Framework for Intrusion Detection Systems. IEEE Access, v. 8, p. 73127–73141, 2020. https://doi.org/10.1109/ACCESS.2020.2988359.

[54] MAHDAVIFAR, S.; GHORBANI, A. A. DeNNeS: deep embedded neural network expert system for detecting cyber attacks. Neural Computing and Applications, v. 32, n. 18, p. 14753–14780, 2020. https://doi.org/10.1007/s00521-020-04830-w.

[55] GALLANT, S. I. Neural Network Learning and Expert Systems. 3. ed. Cambridge: MIT Press, 1995.