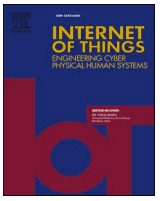
Internet of Things 28 (2024) 101367

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.sciencedirect.com/journal/internet-of-things



Empirical evaluation of feature selection methods for machine learning based intrusion detection in IoT scenarios

José García *, Jorge Entrena, Álvaro Alesanco

Aragón Institute of Engineering Research (I3A), University of Zaragoza, Zaragoza, Spain

ARTICLE INFO

Keywords:
Datasets
Feature selection
Intrusion detection
IoT
Machine learning

ABSTRACT

This paper delves into the critical need for enhanced security measures within the Internet of Things (IoT) landscape due to inherent vulnerabilities in IoT devices, rendering them susceptible to various forms of cyber-attacks. The study emphasizes the importance of Intrusion Detection Systems (IDS) for continuous threat monitoring. The objective of this study was to conduct a comprehensive evaluation of feature selection (FS) methods using various machine learning (ML) techniques for classifying traffic flows within datasets containing intrusions in IoT environments. An extensive benchmark analysis of ML techniques and FS methods was performed, assessing feature selection under different approaches including Filter Feature Ranking (FFR), Filter-Feature Subset Selection (FSS), and Wrapper-based Feature Selection (WFS). FS becomes pivotal in handling vast IoT data by reducing irrelevant attributes, addressing the curse of dimensionality, enhancing model interpretability, and optimizing resources in devices with limited capacity. Key findings indicate the outperformance for traffic flows classification of certain tree-based algorithms, such as J48 or PART, against other machine learning techniques (naive Bayes, multi-layer perceptron, logistic, adaptive boosting or k-Nearest Neighbors), showcasing a good balance between performance and execution time. FS methods' advantages and drawbacks are discussed, highlighting the main differences in results obtained among different FS approaches. Filter-feature Subset Selection (FSS) approaches such as CFS could be more suitable than Filter Feature Ranking (FFR), which may select correlated attributes, or than Wrapper-based Feature Selection (WFS) methods, which may tailor attribute subsets for specific ML techniques and have lengthy execution times. In any case, reducing attributes via FS has allowed optimization of classification without compromising accuracy. In this study, F1 score classification results above 0.99, along with a reduction of over 60% in the number of attributes, have been achieved in most experiments conducted across four datasets, both in binary and multiclass modes. This work emphasizes the importance of a balanced attribute selection process, taking into account threat detection capabilities and computational complexity.

1. Introduction

The proliferation of Internet of Things (IoT) devices has revolutionized how we interact with technology, embedding connectivity into everyday objects ranging from household appliances to industrial machinery. However, as this interconnected web expands, so too

* Correspondence author at: Aragón Institute of Engineering Research (I3A), University of Zaragoza, Zaragoza, Spain.
E-mail addresses: jogarmo@unizar.es (J. García), 779786@unizar.es (J. Entrena), alesanco@unizar.es (Á. Alesanco).

https://doi.org/10.1016/j.iot.2024.101367

Available online 7 September 2024

does the threat landscape surrounding IoT cybersecurity. In recent years, a myriad of vulnerabilities has come to light, exposing the fragility of these devices and the potential risks they pose to individuals, organizations, and entire networks [1]. One of the most pressing concerns in the realm of IoT cybersecurity is the pervasive lack of default security measures inherent in many IoT devices. Unlike traditional computing devices, which often come equipped with robust security features, IoT devices frequently enter the market with minimal to no built-in security protocols. This absence leaves them vulnerable to exploitation by malicious actors seeking to compromise their functionality for nefarious purposes. Moreover, the prevalence of exportable default credentials exacerbates this vulnerability, providing cybercriminals with easily accessible entry points to IoT devices. These default credentials, often left unchanged by users or manufacturers, serve as open doors for unauthorised access, allowing attackers to infiltrate devices and manipulate them to their advantage. Compounding these concerns is the alarming trend of IoT devices being recruited into botnets, effectively transforming them into foot soldiers in large-scale cyberattacks. By commandeering thousands or even millions of compromised devices, attackers can orchestrate coordinated assaults on networks, overwhelming defenses and causing widespread disruption. Furthermore, the interconnected nature of IoT ecosystems means that compromised devices can serve as entry vectors to local area networks (LANs), providing unauthorised access to sensitive data and resources. Once inside a LAN, attackers can exploit interconnected devices and systems, potentially compromising critical infrastructure or compromising personal information. Considering these challenges, it is evident that securing IoT devices is paramount to safeguarding the integrity and functionality of connected systems. Certain security and privacy concerns have been thoroughly investigated with efforts focused on addressing limitations and establishing best practices to mitigate current IoT security threats [2,3].

One of the keys to safeguarding IoT networks lies in continuous threat monitoring through an Intrusion Detection System (IDS). Anomaly-based IDS construct a model of normal behavior based on training data, whereby any data that deviates from this model is categorized as an anomaly [4]. Machine learning (ML)-based IDSs utilize these techniques to derive a model, which is subsequently employed to classify the traffic. Several previous researches offer systematic reviews of cybersecurity attack detection in IoT, utilizing different AI methods including deep learning (DL) and machine learning (ML) techniques [5–7]. The work in [5] presents a survey of IDS research efforts for IoT identifying leading trends, open issues, and future research possibilities. Besides, the work in [6] focuses on state-of-the-art literature on ML techniques applied in IoT and intrusion detection for computer network security. The study in [7] revealed that Support Vector Machines (SVM) and Random Forest (RF) are frequently employed methods, primarily attributed to their high detection accuracy, with another contributing factor being their memory efficiency.

The datasets proposed to derive the traffic models and to evaluate the ML techniques usually contain a high number of attributes which characterize the traffic connections. However, in the context of IoT, it may be imperative to consider attribute reduction by means of feature selection (FS) methods as a fundamental step prior to applying machine learning (ML) techniques. The proliferation of connected IoT devices has resulted in the generation of vast amounts of data, which may contain redundant, irrelevant, or potentially detrimental information, compromising the quality of subsequent ML models. The need for attribute reduction is driven by several pivotal factors. Firstly, the simplification of data serves as a potent strategy to mitigate the challenges posed by the "curse of dimensionality." This phenomenon arises when high data dimensionality leads to performance degradation, overfitting, and very long processing times in ML algorithms. By judiciously eliminating irrelevant or superfluous attributes, the complexity of the model is significantly reduced, thereby enhancing its capacity to identify and generalize meaningful patterns within the data. Secondly, attribute reduction assumes paramount significance in augmenting the interpretability of ML models, especially within the intricate realm of IoT. Models characterized by simplicity, stemming from a reduced number of attributes, are inherently more comprehensible and easier to articulate. This aspect is of utmost importance in applications where decision-making processes necessitate transparency and intelligibility. Furthermore, the process of attribute reduction plays an essential role in optimizing resources within IoT devices with limited processing and storage capacity. By selectively transmitting and storing solely the most pertinent data, this approach leads to the efficient utilization of bandwidth and memory resources, ultimately enhancing operational efficiency and cost-effectiveness. The aim of this work was to comprehensively evaluate different machine learning techniques and feature selection methods for classifying traffic flows in datasets that include intrusions in IoT environments. An extensive benchmark study of ML techniques and FS methods has been conducted, wherein feature selection methods based on Filter Feature Ranking (FFR), Filter-Feature Subset Selection (FSS), and Wrapper-based Feature Selection (WFS) have been assessed.

The main contributions of the paper are as follows:

- The results obtained from the multiple tests conducted have revealed a better classification ability of certain tree-based algorithms, such as J48, over other machine learning techniques (such as multilayer perceptron, logistic regression, adaptive boosting, etc.). This demonstrates a favorable balance between their performance and execution time.
- The comprehensive analysis of the results obtained from diverse representative IoT datasets has allowed for a deeper exploration of the benefits and drawbacks associated with utilizing various families of FS methods. Results led to the conclusion that Filter-feature Subset Selection (FSS) approaches, like CFS, could be more appropriate than Filter Feature Ranking (FFR) or Wrapper-based Feature Selection (WFS) methods.
- Reducing attributes through FS has facilitated the improvement of binary and multiclass classification, achieving F1 scores exceeding 0.99 in most cases, with a reduction of over 60% in the number of features.

This study is organized as follows: Section 2 provides a summary of intrusion detection datasets in IoT scenarios. Section 3 outlines ML techniques and feature selection methods analyzed in this work. The methodology followed and benchmark description is presented in Section 4. Subsequently, Section 5 presents the experimental results, followed by discussion in Section 6, concluding with the final remarks in the paper's conclusions.

2. Intrusion detection datasets in IoT scenarios

2.1. Existing network datasets in IoT scenarios

During the past few years, several datasets have been proposed for evaluating various traffic detection and classification techniques. These datasets typically consist of individual connections and/or traffic flows comprising benign traffic as well as various types of attacks, serving as testbeds for intrusion detection systems. A comprehensive review of generic IDS datasets can be found in [8,9]. In the specific context of IoT, datasets generated in scenarios that include IoT devices and more common attacks in this environment have also been proposed. Some of the most well-known datasets are: BoT-IoT [10], LATAM-DoS-IoT and LATAM-DDoS-IoT [11], TON_IoT [12,13], CIC-IoT [14] or MQTTset [15], among others. In this work, we have chosen the first four datasets, as they have been deemed sufficiently representative of IoT scenarios, both in terms of devices included (real and simulated) and types of attacks. Below, the main characteristics of these datasets are briefly described.

2.2. LATAM-DoS-IOT and LATAM-DDoS-IoT datasets

The LATAM-DDoS-IoT dataset was designed and created during a collaboration among Aligo, Universidad de Antioquia, and Tecnologico de Monterrey [11]. They built and implemented a testbed for DoS and DDoS attacks, which is mainly based on physical IoT devices and real users consuming real services from a production network. Four physical IoT devices and one simulated IoT device were the victims of the DDoS attacks. The physical IoT devices were two Google Home Mini, one smart power strip, and one smart light bulb, connected via an access point. The simulated device using Node-RED was a thermostat running on a container in a virtual machine. Both the ground truth pcap files and the generated network flows (argus and csv files), complete with their respective features, labeled categories, and subcategories, are included. The total number of samples for the DoS and for the DDoS versions of the dataset are, respectively, 30,662,911 and 49,666,991 flows, and include 20 attributes (see Table 8 in Appendix for detailed feature description). The traffic-flows balanced datasets (used in this work) consist of 2407,102 instances for the DoS version and 2431,453 for the DDoS.

2.3. The BoT-IoT dataset

The BoT-IoT dataset was created by designing a realistic network environment in the Cyber Range Lab of UNSW Canberra [10]. It encompasses various IoT devices, including a weather station, a smart refrigerator, motion-activated lamps, a garage door, and a smart thermostat. The dataset's source files are provided in different formats, including the original pcap files, the generated argus files and csv flows derived files. The network environment incorporated a combination of normal and botnet traffic including DDoS and DoS attacks (organized according to the transport protocol used), OS Fingerprint, Service Scan, Keylogging and Data Exfiltration attacks. The captured pcap files are 69.3 GB in size, with more than 72.000.000 records, and the extracted traffic flows, in csv format is 16.7 GB in size. To facilitate the management of the dataset, a version containing 5 % of the original dataset is available, encompassing roughly 1 GB in total size and comprising approximately 3 million records (3668,522 instances). The dataset is composed of 46 attributes for each traffic flow (see Table 9 in Appendix for detailed feature description).

2.4. TON_IoT (UNSW-IoT20) dataset

The TON_IoT (UNSW-IoT20) datasets are new generations of Internet of Things (IoT) and Industrial IoT (IIoT) datasets built for evaluating the fidelity and efficiency of different cybersecurity applications [12]. They include heterogeneous data sources collected from Telemetry datasets of IoT and IIoT sensors, Operating systems datasets and Network traffic datasets. The datasets were collected from a realistic representation of a medium-scale network in the Cyber Range and IoT Labs at UNSW Canberra (Australia), generating several normal and cyber-attack events. The testbed was developed to connect virtual machines, physical systems, hacking platforms, cloud and fog platforms, IoT and IIoT sensors. Different hacking techniques, such as DoS, DDoS and ransomware, were launched against web applications, IoT gateways and computer systems across the IIoT network. The testbed has a combination of physical and simulated IoT/IIoT services. The real devices include two smartphones, a smart TV and physical weather sensors. The simulated IoT services include smart fridge, garage door, GPS tracker, motion light, modbus and thermostat sensors. The dataset considered consists of 461,041 instances with 45 attributes for each traffic flow (see Table 10 in Appendix for detailed feature description). It includes 9 types of attacks namely Scanning, DoS, Injection, DDoS, Password, XSS, Ransomware, Backdoor and MITM.

3. Machine learning techniques and feature selection methods

In this section, we present a brief description of the ML techniques and FS methods that have been evaluated in this work.

3.1. ML techniques

The following are some specific ML classifiers representative from the most commonly utilized ML techniques, accompanied by a brief explanation of how they operate.

- **Naive Bayes**: Bayesian methods offer a quantitative assessment of the probabilistic relevance of variables in the classification problem. This classifier is a probabilistic method based on computing conditional probabilities and Bayes' theorem, and operates under the assumption that all attributes are independent of each other [16].
- **Logistic**: These function-based methods generate a classification function without explicitly deriving a tree or set of rules. The logistic regression is generalized in the multinomial regression model for multiclass problems [16].
- **Multilayer perceptron (MLP)**: An artificial neural network allows to address problems that are not linearly separable The MLP used in this work consists of three layers: attributes enter an input layer and an output layer provides classification values for instances [16].
- **Adaptive boosting (AB)**: This metaclassifier is based on a meta-algorithm for statistical classification, which is used in conjunction with other learning algorithms to enhance performance [16]. Through this approach, the outputs of weak learners are amalgamated into a weighted sum, representing the final output of the non-linear classifier.
- **k-Nearest Neighbors (kNN)**: This non-parametric classification approach classifies an instance based on the classes of the k-most similar training instances [16]. These techniques involve a trivial learning process (lazy learners), with the classification process being the most time-consuming.
- **J48:** This algorithm is an implementation of C4.5 [17], which generates a decision tree by recursively partitioning the data. The procedure for creating the decision tree involves selecting one attribute as the root of the tree and branching out for each possible value of that attribute. In each resulting branch, a new node is generated, and the process is repeated until all instances have been classified along some path in the tree.
- **PART:** This algorithm represents a simplified (partial) method based on the C4.5 algorithm and is founded on the construction of rules [18]. In this algorithm, the branches of the decision tree are substituted with rules. Only the branches of the tree that are most effective in selecting a class are included. The strategy employed allows for great flexibility and speed. Using PART does not yield a complete tree, but rather a partial one, in which the construction and pruning functions are combined until a stable subtree that cannot be further simplified is found. A rule is generated from that subtree.
- **Random Forest (RF):** Random forests are constructed by assembling sets of random trees [16,19]. Trees created through this algorithm take into account a specific number of random features at each node, without undergoing any pruning. The final predictions of the random forest are derived by averaging the predictions obtained from each individual tree. The use of a random forest can mitigate the overfitting effect observed in individual decision trees, albeit at the cost of increased computational complexity.
- **OneR**: In spite of the fact that the OneR algorithm is not a decision tree algorithm, this technique stands out as one of the simplest and fastest classification algorithms [20]. In OneR, a rule is generated for each attribute, and the one that yields the least classification error is selected. It does not follow the hierarchical structure of a typical decision tree.

All tests have been conducted in offline mode in this work; however, time complexity is an important factor to consider in further implementations. It is necessary to distinguish between model creation and testing phases, as the last one is the most significant when evaluating the performance in potential real-time systems. A summary of computational complexity for ML techniques can be found in [25]. During the testing phase, computational complexity tends to be lower than in the training phase, resulting in faster execution times, typically on the order of linear time relative to the input data size. However, for lazy learner classifiers like kNN, the testing phase takes longer and may not be suitable for real-time use.

3.2. Feature selection methods

Prior to applying ML techniques, it is beneficial to select the most relevant attributes. Feature selection proves effective in tackling the dimensionality problem [21–23], and there are two primary reasons for employing FS methods. Firstly, certain techniques suffer performance setbacks when handling non-relevant attributes (as in the case of naive Bayes), and secondly, the insufficient instances for some attack classes do not justify using numerous features. Indiscriminate inclusion of all features easily leads to overfitted classifiers [24]. Previous studies have demonstrated that irrelevant and redundant features significantly impact classifier accuracy [22,23,25,26]. Additionally, a higher number of attributes escalates computational complexity and algorithm execution time. There exist various techniques for conducting feature selection, and comprehensive reviews of these methods can be found in [21–23,27,28]. Some FS methods in which a transformation of attributes occurs to move to another information space (as in the case of principal component analysis (PCA)) have not been included in this study since the priority has been to preserve the information and meaning of the attributes considered.

Among FS methods, the **Filter Feature Ranking (FFR)** methods assess and rank features based on their rank scores, considering the dataset's properties [27]. These methods select and evaluate attributes independently of the subsequent classification algorithm. Examples of FFR methods include information gain filter, gain ratio attribute filter, correlation filter, symmetrical uncertainty filter, and others.

- **Information Gain (InfoGain)**: Evaluates the worth of an attribute by measuring the information gain with respect to the class as expressed in Eq. (1). *H(Class)* represents the entropy of the dataset with respect to the class variable and measures the uncertainty in the distribution of classes in the dataset, whereas *H(Class | Attribute)* represents the conditional entropy of the dataset given the value of the feature and represents the dataset entropy after splitting it based on the feature value. The computational complexity of the FS method based on Information Gain generally correlates with the number of samples (n), the number of features (m), and the

number of classes (c) in the dataset. Generally, it could be expressed as $O(n{\cdot}m{\cdot}c)$. However, this is a simplified representation and the actual complexity may vary depending on the specific implementation and algorithm used in the calculation of Information Gain.

$$InfoGain(Class, Attribute) = H(Class) - H(Class \mid Attribute) \tag{1}$$

- **Gain Ratio**: Evaluates the value of each attribute in the dataset based on the information gain it provides along with its entropy (see Eq. (2)). It is an extension of InfoGain that addresses some of its limitations by taking into account the intrinsic information of a feature and helps to overcome the bias towards attributes with a large number of values. The computational complexity can also vary depending on the specific implementation and algorithms involved in the computation but can generally be expressed as $O(n{\cdot}m{\cdot}c)$, analogously to the previous method.

$$GainRatio(Class, Attribute) = \frac{H(Class) - H(Class \mid Attribute))}{H(Attribute)} \tag{2}$$

- **Correlation:** Evaluates the worth of an attribute by measuring the Pearson's correlation between the attribute and the class. Features with a high positive or negative correlation with the target variable can be considered more informative. It assumes a linear relationship and may not capture non-linear relationships between variables. It should be noted that some features may have a high correlation with the class without being truly relevant for prediction. The computational complexity can generally be expressed as $O(n{\cdot}m^2{\cdot}c)$, which increases quadratically concerning the number of features. However, in practice, there exist optimizations for calculating correlation that can enhance performance.
- **Symmetrical Uncertainty (SymUncert)**: Evaluates the worth of an attribute by measuring the symmetrical uncertainty with respect to the class as expressed in (3). This method relies on the concept of mutual information or shared information between two random variables. Features with a high symmetrical uncertainty with the target variable are considered more informative for classification. A higher value of uncertainty indicates a stronger association between the feature and the target variable. The computational complexity of this method can be expressed as $O(n{\cdot}m{\cdot}c)$, although there are specific approaches and optimizations that can enhance computational efficiency employing efficient data structures, algorithms optimized for mutual information calculation, or dimensionality reduction techniques.

$$SymmUncert(Class, Attribute) = 2\frac{H(Class) - H(Class \mid Attribute))}{H(Class) + H(Attribute)} \tag{3}$$

On the other hand, in **Filter-feature Subset Selection (FSS)** methods, features undergo evaluation and ranking via a search method acting as the subset evaluator. These search methods evaluate features based on their usefulness for better prediction performance. One of the most prominent FSS techniques is correlation-based feature subset selection (CFS).

- **Correlation-based Feature Subset Selection (CFS)**: Evaluates the worth of a subset of attributes by considering the individual predictive capacity of each feature along with the degree of redundancy between them. Hence, the CFS method identifies the optimal subset of features that work synergistically, distinct from a subset of features ranked solely based on their individual classification capacity, as would be the case of previous ranker selection methods. CFS ranks feature subsets according to a correlation based evaluation function (see Eq. (4)), where M_s is the heuristic merit of a feature subset S containing k features, $\overline{r_{cf}}$ is the mean feature-class correlation ($f \in S$), and $\overline{r_{ff}}$ is the average feature-feature inter-correlation [29]. The numerator provides a measure of how predictive the class of a set of features is, while the denominator provides a measure of how much redundancy there is among the features. The CFS method involves calculating the correlations between the features and the target class, as well as correlations among the features themselves. The computational complexity of CFS can be approximately $O(n{\cdot}m^2{\cdot}c)$, considering the calculation of correlations among all possible feature combinations, although implementations usually include optimizations to reduce the number of required computations by optimizing the subsets search.

$$M_s = \frac{k\,\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{4}$$

In contrast to previous methods, **Wrapper-based Feature Selection (WFS)** methods employ a classifier to determine the adequacy of an attribute subset, returning the most significant attributes for that classifier. The classifier is predetermined, and the generated feature subsets typically exhibit bias toward the base classifier used for evaluation [28]. Some of the most relevant methods are described below.

- **Classifier**: Evaluates the value of an attribute by utilizing a designated classifier (e.g., Naïve Bayes, J48, or Random Forest) instead of evaluating features independently. The classifier is trained and evaluated on the original dataset and is utilized to measure how each attribute contributes to the classifier's performance. Following the evaluation, attributes are ranked based on their importance to the classifier. This FS method relies on evaluating a classifier's performance using a specific subset of features and hence the process involves training and evaluating a classifier for each feature, which can be computationally expensive. The computational complexity of this method (similarly to other WFS methods) primarily depends on the classification algorithm used, as well as the number of features and samples in the data, and therefore there is not a simple expression that precisely describes the complexity of this FS method. In practice, the execution time can significantly increase as the number of features grows since it involves more classifier trainings and evaluations.
- **Classifier Subset**: This method uses a classifier as in previous one, but in this case to estimate the relevance of a subset of attributes. It evaluates different subsets of attributes to determine which ones are more informative in terms of classification based on the performance of a specific classifier. The classifier is trained and evaluated for each generated subset of attributes and its performance measure is used to assess the quality of each subset. After evaluating various subsets, a ranking is generated based on their performance. The computational complexity of the method depends on the classification algorithm used, as well as the number of features, samples in the data and classes. It can be high due to exploring multiple feature combinations and may vary depending on the search strategy employed (e.g., forward selection, backward elimination, random search), the complexity of the base classifier, the performance metric used, etc. The execution time can significantly increase as the number of features grows since it involves evaluating multiple feature subsets using the base classifier and it might be unpractical in large datasets.
- **Wrapper Subset:** Evaluates attribute sets by using a learning scheme with a wrapper approach. Its goal is to select the minimum set of features that provide the best model performance for a specific classifier. It follows a layered learning approach: initially, a group of attributes is randomly selected, and the model is evaluated using a performance metric. A new group of features is then chosen with the aim of improving the previous one, and this process continues until the desired performance criterion is reached. Both ClassifierSubset and WrapperSubset are truly feature evaluators and not feature selection algorithms per se. In ClassifierSubset method, feature selection is carried out using a specific classifier, and each subset of features is evaluated based on the performance of that selected classifier. On the other hand, in WrapperSubset, a classifier is also specified, but the term 'wrapper' implies that feature selection is directly integrated into the classification process. The complexity of this FS method is influenced by various factors, including the classifier used as the wrapper, dataset size, number of features, etc., and might be more computationally intensive than other FS methods.

As it has been explained, feature selection may entail exploring potential combinations of attributes within the dataset to identify the optimal subset of features. To accomplish this, two components should be configured: an attribute evaluator and a search method. The evaluator determines the method employed to assign a value to each single attribute or subset, whereas the search method dictates the type of search to be executed. Therefore, an attribute evaluator is initially employed to assign specific weights to each attribute.

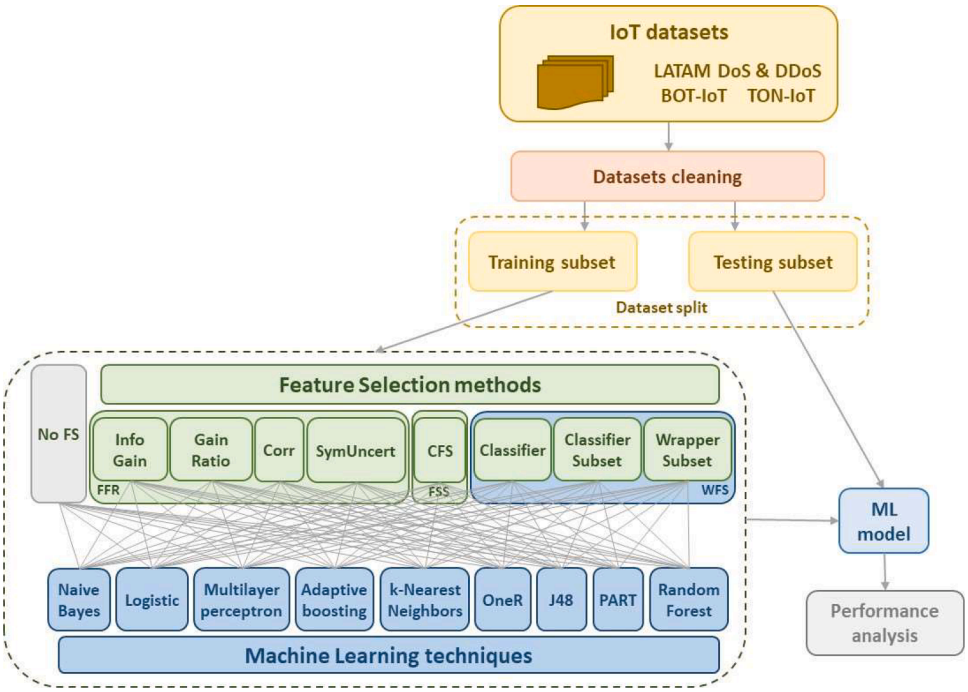


Fig. 1. Benchmark for FS methods and ML techniques evaluation.

Subsequently, the search method is responsible for generating the search space to identify a high-quality subset of attributes. In this work, for individual attribute evaluators, the Ranker method was selected as search method, which ranks attributes by their individual evaluations. For subset attribute evaluators, the Best First method was employed to explore the attribute subset space, which searches the space of attribute subsets by greedy hill-climbing augmented with a backtracking facility.

4. Traffic flow classification testbed

In this section, we outline the methodology designed for empirically evaluating FS methods and ML techniques. The evaluation process is represented in Fig. 1. It starts with an initial dataset cleanup and refinement removing inadequate attributes (IP addresses and ports, sequence number, etc.). Afterward, the training and learning subsets are defined allocating 66% of the data for model training and reserving the remaining 33% for testing purposes. An independent analysis is conducted for both binary classification and multiclass classification during the evaluation process since FS methods can yield different attributes selection. Therefore, an attribute selection is performed for the binary classification of traffic flows (attack vs. normal connection) and another attribute selection to differentiate specific attack types. The complete benchmark included eight FS methods (plus no selection of attributes) which were evaluated in combination with nine ML techniques, both in binary and multiclass classification modes. FFR and FSS methods are applied prior to the execution of the ML technique, thereby deriving sets of attributes independent of the ML technique applied afterward. In contrast, WFS methods require their use in combination with the selected ML technique.

The performance metric selected for traffic flow classification in this work is the F1-Score, which combines precision and recall into a single metric [16], yielding a balanced evaluation of classifier performance (as expressed in (5)). F1 scores are provided for the sake of simplicity and because they are the most widely used in studies of traffic classification. Precision quantifies actual attacks among all flows classified as attacks, recall measures the frequency of actual attacks among all attacks, whereas accuracy measures the ratio of correctly classified traffic flows to the total number of flows. It is important to note that when no attack has been correctly classified and all benign flows have been correctly classified a valid numerical result cannot be obtained for precision and F1 score (the "?" symbol will be indicated for those cases).

$$\mathrm{F1} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{5}$$

In this work WEKA (Waikato Environment for Knowledge Analysis) [30,31] was selected for implementing FS methods and ML techniques, although other environments such as scikit-learn would have been equally feasible. The application of WEKA took place on a system running Windows 11 Operating System, equipped with an Intel(R) Core(TM) i7–1065G7 CPU @ 1.30 GHz (4 cores) and 16 GB RAM.

5. Evaluations and results

In this section, we offer a succinct overview of the diverse experimental results obtained, along with their interpretation. The complete benchmark included a comprehensive empirical analysis of eight FS methods (plus no selection) and nine ML techniques for two different types of classification (binary and multiclass) across the four selected datasets, resulting in a significant amount of results. Therefore, only the key findings are presented below. The Weka parameter settings of the implemented ML methods were the following: Naive Bayes (useKernelEstimator=False), Logistic (ridge=1,0e-8; doNotStandardize Attributes=False), Multilayer Perceptron (hiddenLayers=1; learningRate=0,3; validation Threshold=20; trainingTime=300 epochs), k-Nearest Neighbors (KNN=1; nearestNeighbour Search Algorithm=LinearNNSearch), Adaptive boosting (classifier=Decision Stump; numIterations=10; weight Threshold=100), OneR (minBucketSize=6), PART and J48 (confidenceFactor=0,25; numFolds=3; min-NumObj=2; useMDLcorrection=True), and Random Forest (num Iterations=100; maxDepth=unlimited; bagSize Percent=100; numFeatures=int(log_2(#predictors)+1)).

5.1. Results in LATAM DoS dataset

The distribution of instances among classes in LATAM-DoS dataset is as follows: normal traffic flows (799.187 instances) and attack traffic flows (1.607.915 instances), and the distribution of attacks is: TCP (799.187), UDP (799.187) and HTTP (9.541). In order to work with this dataset, an initial attribute cleaning was performed by removing: StartTime, LastTime, and Seq, as they are related to the times when the attacks were executed. A flow-based IDS that incorporates them could incorrectly associate the timestamp or sequence number with a particular traffic category. For example, if an attribute represents the sequence number of traffic flows (Seq) and is used in the ML model, flows with close sequence number values will belong to the same attack (e.g., in a denial-of-service attack launched during a specific time interval). Thus, some of these attack flows (corresponding to the same attack) may appear in both the training set and the test set, making them characteristic of that dataset. Therefore, we strongly believe that such attributes should be eliminated from any semiautomatic learning approach. Hence, a total of 15 attributes were considered.

The results after applying the feature selection methods to the LATAM-DoS dataset are shown in Table 1. Attributes selected for binary classification, i.e., distinguishing only between normal and attack classes are present at the top of the table. On the other hand, attributes selected for multiclass classification, i.e., distinguishing among the different attack classes over HTTP, TCP, and UDP protocols are shown in the lower part of the table. We can observe that for binary classification, the most relevant features in the

majority of the evaluated FS algorithms were TotPkts and TotBytes (which are highly correlated), followed by SrcPkts and SrcBytes. No significant differences were found among the attribute selections made by the different techniques, except for the Correlation method. We found high similarities in the attribute selection for most of the FS methods comparing binary and multiclass modes, with TotBytes and SrcBytes being the most frequently selected attributes in the multiclass mode. Attributes like Dur, Mean, Max, Sum, and Min appeared to contain similar information.

To provide an initial comparison among the nine ML techniques, Table 2 displays their F1 score results obtained without applying any FS method (binary classification is depicted at the top of the table). The lower part of the table shows the F1 scores using the CFS method for multiclass classification. This table allows to illustrate, through these different combinations of FS method/ML technique (in binary and multiclass classification), the behavior observed overall for all evaluated situations. Better classification performance was achieved by decision tree and/or rule-based techniques against other ML algorithms. The best F1 results were reached using the RF algorithm, closely followed by PART and J48, although the latter two exhibit much greater efficiency in terms of computation time employed for classifying traffic flows. Although KNN generally achieved good results, it came at the expense of very high computation times. In fact, some ML algorithms operating with wrapper-based FS methods may result in impractical computation times, such as in the case of KNN and WrapperSubset, which can require several days of computation time. The rest of the techniques exhibited substantially poorer F1 results, even leading to the misclassification of all traffic flows of a certain type of attack in multiclass classification (see "?" in the table as result of this fact).

In order to summarize the results analysis presented in the paper, only the results with the highest F1 classification rates are displayed henceforth, which were mostly obtained using tree and/or rule-based algorithms. Moreover, the best attribute subsets obtained (considering subsets ranging from a single attribute up to all the available attribute) are shown. Table 3 displays the best F1 scores reached in LATAM-DoS for binary and multiclass classification. This table shows the selected attributes, followed by the FS method used to select them (if any), the ML technique that achieved the best classification results, the number of flows per second that can be classified during the testing phase, and finally, the F1 scores reached. Very high F1 scores (0.999/1) were achieved in binary classification even with only one attribute (using OneR as the classifier). This value seems quite close to that obtained with all attributes (using the RF algorithm). However, when examining the number of incorrectly classified instances in detail, this value increases from 45 when using 15 attributes up to 339 when only one attribute is selected. As expected, the RF technique showed high execution times. Very good results were reached when the J48 algorithm was applied using 4 or 2 attributes selected with WrapperSubset and ClassifierSubset methods, respectively. In these cases, only 69 and 157 instances were incorrectly classified out of 818.415 traffic flows, and J48 presented short execution times during the testing phase. In general, reducing the number of attributes results in an increase of the number of flows per second processed. Nevertheless, this effect was not always proportional to the number of attributes considered. It can be explained in the case of tree-based algorithms due to trees derived with fewer attributes were often more complex in terms of the number of leaves and tree size than trees derived with more attributes. This fact justifies why execution times did not decrease as expected. On the other hand, the time required to select the optimal attribute subset can occasionally be substantial (especially evident in wrapper-based methods, where ML algorithms must be combined with the feature selection method). In those cases, the attribute selection processes could potentially extend over a day of computational time. In general, FFR and FSS methods can operate in real-time whereas WFS methods require much more time. Using a single attribute (TotBytes with J48) for multiclass classification (in the lower part of Table 3) also achieved very good results, with F1 scores close to 1 and only 347 incorrectly classified instances out of 818,415 traffic flows. Unlike the binary case, a more significant improvement was obtained when more attributes are considered, as the multiclass classification problem becomes more challenging than the binary case. The best results were achieved using the RF technique. It is remarkable that quite similar results can be obtained using only 5 attributes (selected using the ClassifierSubset method) with J48 compared to using all 15 (60 vs. 43 incorrectly classified instances, respectively). The worst performance was obtained in classifying HTTP-based attacks due to their lower representation in the dataset.

Table 1
Feature Selection in LATAM-DoS dataset for binary and multiclass classification (attributes selected using rankers are presented in order of selection).

	FS METHOD	SELECTED ATTRIBUTES
Binary classification	**InfoGain**	TotBytes; SrcBytes; TotPkts; Rate; SrcPkts
	GainRatio	TotPkts; SrcPkts; DstPkts; DstBytes; SrcBytes
	Correlation	Max, Mean, Sum, Dur, Min; SrcRate; Rate; DstRate; DstPkts
	SymUncert	TotPkts; SrcPkts; DstPkts; DstBytes; SrcBytes
	CFS	TotPkts: SrcPkts
	Classifier	TotBytes; SrcBytes; TotPkts; Rate; SrcPkts
	ClassifierSubset	TotBytes, Dur
	WrapperSubset	TotBytes; Dur; DstBytes; Rate
Multiclass classification	**InfoGain**	TotBytes; SrcBytes; Rate; Dur, Mean, Max, Sum, Min; SrcRate
	GainRatio	SrcBytes; TotBytes; TotPkts; SrcPkts; DstPkts
	Correlation	Max, Mean, Sum, Dur, Min; SrcRate; Rate; DstRate; DstPkts
	SymUncert	SrcBytes; TotBytes; TotPkts; SrcPkts; DstPkts
	CFS	TotBytes; SrcBytes
	Classifier	TotBytes; SrcBytes; Max, Mean, Sum, Dur, Min; Rate; SrcRate
	ClassifierSubset	TotBytes; SrcPkts; DstBytes; Rate; DstRate
	WrapperSubset	TotPkts; TotBytes; Dur; DstPkts; DstBytes; Rate; SrcRate

Table 2
F1 scores in LATAM-DoS for binary classification (using all the attributes) and multiclass classification (using the attributes selected with CFS method).

Binary classification				F1 Score			
Attributes	FS method	ML technique	Flows/s	Normal			Attack
All (15)	None	RF	70.477	1			1
All (15)	None	PART	1.305.981	1			1
All (15)	None	J48	252.597	1			1
All (15)	None	OneR	1.573.875	0,999			1
All (15)	None	KNN	3,73	0,999			0,999
All (15)	None	Logistic	1.116.020	0,992			0,996
All (15)	None	AB	204.603	0.984			0.992
All (15)	None	MLP	840.837	0,977			0,989
All (15)	None	NB	220.004	0,963			0,982
Multiclass classification				F1 SCORE			
Attributes	FS method	ML technique	Flows/s	Normal	TCP	UDP	HTTP
TotBytes, SrcBytes	CFS	RF	73.864	1	1	1	0,990
TotBytes, SrcBytes	CFS	PART	786.937	1	1	1	0.988
TotBytes, SrcBytes	CFS	J48	405.156	1	1	1	0.988
TotBytes, SrcBytes	CFS	KNN	4,5	1	1	1	0,988
TotBytes, SrcBytes	CFS	OneR	2.455.245	0,999	1	1	0,954
TotBytes, SrcBytes	CFS	AB	767.264	0,960	1	0,968	?
TotBytes, SrcBytes	CFS	Logistic	1.180.406	0,987	0,994	0,996	?
TotBytes, SrcBytes	CFS	NB	626.338	0,111	0	0,666	0,024
TotBytes, SrcBytes	CFS	MLP	959.080	0,499	?	?	?

Table 3
F1 scores in LATAM-DoS for binary and multiclass classification.

Binary classification				F1 Score			
Attributes	FS method	ML technique	Flows/s	Normal			Attack
All (15)	None	RF	70.477	1			1
TotBytes	InfoGain	OneR	256.556	0.999			1
TotBytes, Dur	ClassifierSubset	J48	442.386	1			1
TotBytes, Dur, DstBytes, Rate	WrapperSubset	J48	629.550	1			1
Multiclass classification				F1 SCORE			
Attributes	FS method	ML technique	Flows/s	Normal	TCP	UDP	HTTP
All (15)	None	RF	77.281	1	1	1	0.994
TotBytes	Classifier	J48	345.323	0.999	1	1	0.956
TotBytes, SrcBytes	CFS	J48	405.156	1	1	1	0.988
TotBytes; SrcPkts; DstBytes; Rate; DstRate	ClassifierSubset	J48	528.010	1	1	1	0.994

Table 4
F1 scores in LATAM-DDoS for binary and multiclass classification.

Binary classification				F1 Score			
Attributes	FS method	ML technique	Flows/s	Normal			Attack
All (15)	None	J48	480.636	0.986			0.993
SrcBytes	Classifier	OneR	1.503.080	0.980			0.990
SrcBytes, Rate	WrapperSubset	J48	1.008.163	0.984			0.992
TotBytes, SrcPkts, DstPkts, SrcBytes, DstBytes, Rate	ClassifierSubset	J48	501.027	0.986			0.993
Multiclass classification				F1 SCORE			
Attributes	FS method	ML technique	Flows/s	Normal	TCP	UDP	HTTP
All (15)	None	J48	1.087.755	0.986	1	1	0.485
SrcBytes	Classifier	OneR	888.918	0.980	1	1	0.132
DstPkts, SrcBytes	CFS	J48	547.479	0.983	1	1	0.305
TotBytes, DstPkts, SrcBytes, DstBytes, Rate, DstRate	WrapperSubset	J48	918.549	0.986	1	1	0.493

5.2. Results in LATAM DDoS dataset

The distribution of instances among classes in LATAM-DDoS dataset is as follows: normal traffic flows (799.187 instances) and attack traffic flows (1.632.266 instances), and the distribution of attacks: TCP (799.187), UDP (799.187) and HTTP (33.892). As in LATAM-DoS dataset, to work with LATAM-DDoS dataset, the same attribute cleaning was performed (removing the attributes StartTime, LastTime, and Seq), hence leaving 15 attributes (as in previous dataset).

The optimal classification results achieved in LATAM-DDoS, considering various sizes for the attribute subsets are presented in Table 4, following the same format as in previous Table 3. In this dataset, there was a greater variety of attributes selected by the different FS methods. However, the FS methods that yielded the best results showed agreement in the selection of attributes, such as SrcBytes and Rate. The binary classification results were high again on this dataset but significantly lower than in LATAM-DoS. The J48 algorithm provided the best results with different subsets of attributes, achieving classification levels using only 6 attributes equal to those obtained with 15 attributes. The capacity to classify flows notably increases when reducing the number of attributes to 1 or 2, but not as much with the 6-attribute subset. Regarding execution times (or inversely, flows/s processed), we observe that feature selection results in a reduction, but not proportionally to the number of attributes due to an increase of the tree complexity (number of leaves and size). In multiclass classification the attributes selected largely matched the binary case (see the lower part of Table 4). F1 scores were high for normal traffic flows and for TCP and UDP-based attack classes, but showed very poor performance in classifying HTTP-based attacks (with much lower representation on the dataset). Upon analyzing the confusion matrices, it was discovered that two thirds of the HTTP attack flows were incorrectly classified as normal traffic flows, hence generating a large number of false negatives. Moreover, when using only 1 or 2 attributes, these classification results deteriorated further. However, including 6 attributes (selected using the WrapperSubset method) improved the F1 score, reaching values at least as good as those achieved when considering all 15 attributes. These results show that in some cases, it is better to use a smaller subset of attributes selected through an FS method than including all the available attributes. Similarly to the binary classification case, the capacity to classify (in flows per second) did not improve significantly when reducing the number of attributes due to the complexity of the derived J48 trees.

5.3. Results in BOT-IoT dataset

The distribution of instances among classes in BOT-IoT dataset is as follows: normal traffic flows (477 instances) and attack traffic flows (3.668.045 instances), and the distribution of attacks: TCP (1.593.180), UDP (1.981.230), HTTP (2.474), OS-Fingerprint (17.914), Service-Scan (73.168), and Keylogging (73). A prior attribute cleaning was performed by removing those attributes that are characteristic of the dataset and cannot be generalized to the rest of real-world connections: pkSeqID, stime, flgs_number, proto_number, sadrr, sport, daddr, dport, state_number, ltime, and seq. It is crucial to exclude any data associated with these attributes when training ML techniques. Since all individual attacks present in the dataset originated from the same IPs and ports, traffic flows might exhibit identical values in both the training and test subsets, making them easily recognizable as attacks. Eliminating these attributes helps avoiding this situation. After attribute removal, a total of 32 attributes were considered.

The binary and multiclass classifications results obtained after applying the feature selection methods and machine learning techniques to the BOT-IoT dataset are presented in Table 5. In this dataset, different FS methods lead to varying attribute selections, although attributes like Bytes, State, and those representing traffic flow derived statistics are predominant. In this dataset, results with high F1 values were achieved in binary classification (only 4 instances out of more than one million were incorrectly classified). The best overall performance was obtained using the PART algorithm (although exactly the same results were obtained with the RF technique but with higher execution time). In this case, the normal class was somehow penalized due to being in the minority compared to the attack class. With just 5 out of the 32 attributes (selected using the CFS method) and the RF technique, it was possible to achieve the same classification rate as when using all the features. And using the PART algorithm with the same 5 attributes allowed for reaching an F1 score close to that obtained with RF (only 9 instances were incorrectly classified), while significantly reducing the execution time. The attributes selected as the most relevant in multiclass classification (see the lower part of Table 5) again largely matched the binary case, but the Proto attribute was added with a more significant presence in the selected subsets. F1 scores were very high for all the different traffic classes using all the attributes (53 instances out of more than one million were incorrectly classified) or using 5 or more attributes subsets. F1 scores were lower with up to three attribute subsets in those classes with much lower representation on the dataset (Normal, OS-Fingerprint and Keylogging). Anyway, upon analyzing the confusion matrices, it was found that most of the classification errors occurred when instances from an attack class are misclassified as another attack class, indicating that some attacks may exhibit similar traffic behavior. When using the WrapperSubset method to select the best 5 attributes subset and applying the J48 algorithm, the best classification results were achieved (only 43 instances incorrectly classified). These results were even better than those obtained with PART technique including 12 features selected with ClassifierSubset. As a drawback, the complexity of the resulting J48 tree does not lead to any reduction in execution times.

5.4. Results in TON-IoT dataset

In this dataset, unlike the previous ones, the class of normal traffic flows predominates, with the following distribution of instances: normal traffic flows (300.000 instances) and attack traffic flows (161.041 instances). The attacks are: Scanning (20.000 instances), DoS (20.000 instances), Injection (19.998 instances), DDoS (20.000 instances), Password (20.000 instances), XSS (20.000 instances), Ransomware (20.000 instances), Backdoor (20.000 instances), and MiTM (1.043 instances). Attribute cleaning was performed by removing the following characteristics related to specific information of the attacks in the dataset: Ts, Src_Ip, Src_Port, Dst_Ip, Dst_Port,

Table 5
F1 scores in BOT-IoT for binary and multiclass classification.

Binary classification				F1 Score	
Attributes	FS method	ML technique	Flows/s	Normal	Attack
All (32)	None	PART	437.648	0.987	1
TnBPSrcIP	InfoGain	OneR	959.459	0.859	1
Bytes, State	WrapperSubset	PART	944.922	0.899	1
State, SBytes, AR_P_Proto_P_SrcIP	ClassifierSubset	J48	1.559.121	0.945	1
Dur, SRate, TnP_PerProto, N_IN_Conn_P_SrcIP, N_IN_Conn_P_DstIP	CFS	PART	716.837	0.971	1
Dur, SRate, TnP_PerProto, N_IN_Conn_P_SrcIP, N_IN_Conn_P_DstIP	CFS	RF	128.256	0.987	1

Multiclass classification				F1 SCORE						
Attributes	FS method	ML technique	Flows/s	Normal	HTTP	TCP	UDP	OS-Fingerprint	Service-Scan	Keylogging
All (32)	None	PART	148.135	0.987	0.995	1	1	0.997	0.999	0.846
SBytes	Classifier	OneR	674.214	0.690	0.868	1	1	0.075	0.891	0.222
Flgs, Proto, SBytes	CFS	RF	69.294	0,917	0,917	1	1	0,099	0,894	0,303
Proto, Sbytes, TnBPDstIP, TnP_PerProto, AR_P_Proto_P_DstIP	WrapperSubset	J48	193.080	0.997	0.997	1	1	0.998	0.999	0.957
Proto, SBytes, DBytes, SRate, TnBPDstIP, TnP_PDstIP, TnP_PerProto, TnP_Per_Dport, AR_P_Proto_P_SrcIP, AR_P_Proto_P_Sport, AR_P_Proto_P_Dport, Pkts_P_State_P_Protocol_P_DestIP	ClassifierSubset	PART	474.257	0.991	0.999	1	1	0.997	0,999	0.917

and Dns_Query. After this attribute removal, a total of 38 attributes were considered in our analysis.

The binary and multiclass classifications results obtained after applying the feature selection methods and machine learning techniques to the TON-IoT dataset are presented in Table 6. In this dataset, attributes such as Src_Ip_Bytes and Dst_Ip_Bytes were included in the majority of attribute subsets selections. Classification results present high F1 values in this dataset and the best overall performances were obtained using RF, J48 and PART algorithms. With only 13 out of the 38 attributes, it was possible to achieve an F1 value using the J48 algorithm that is nearly equal to what is obtained using all the attributes, while also reducing significantly the execution times. In multiclass classification (see the lower part of Table 6) some of the attributes selected in the binary study appear again, such as Src_Ip_Bytes, Dst_Ip_Bytes, Conn_State, or Proto. The DNS-related attributes gain slightly more prominence in this study within traffic classes. Similarly, attributes related to HTTP and SSL were not selected, except for Ssl_Resumed. The low presence of MiTM traffic flows in the dataset is evident in their worst classification values in all the tests carried out. Additionally, accurately classifying ransomware attacks proved challenging, with the majority of errors involving attacks misidentified as normal, resulting in false negatives. Again, the results working with a single attribute were not acceptable and the use of 2 and 3 attributes subsets significantly improved F1scores. Using the J48 algorithm with only 13 out of the 38 attributes (selected with WrapperSubsetEval), it was possible to achieve an even slightly better F1 value than that obtained using all the attributes, while also reducing the execution times.

6. Discussion

In this section the main results obtained in the evaluation of FS methods and ML techniques are discussed. As stated in previous section, better classification performance was achieved by decision tree and/or rule-based techniques against other ML algorithms. These specific ML techniques have the following advantages: interpretability (they are easy to understand and their structure resembles a flowchart that shows decisions and outcomes at each stage), handling mixed data (they can work with numerical and categorical features), no data normalization is required, computational efficiency (they often have fast training and testing times), identification of relevant features (helpful in identifying the most important attributes in decision-making). They also have demonstrated good performance in traffic classification in previous works. On the other hand, as main drawbacks decision trees can be prone to overfitting (what can lead to poor performance on new data), they are subject to instability (which can lead to significant changes in the tree structure), and can become very complex with many features. However, after reduction of the number of features they can be an appropriate solution. In this work, the ML technique with the most prominence has been J48, offering a good balance between execution times and classification results, which are challenging to surpass. However, in specific cases, PART or RF yielded the best results. When only one attribute was used, the best results were usually obtained with the OneR algorithm, although F1 scores in these cases were generally lower and may not always be deemed acceptable. In some specific cases, the Random Forest (RF) technique yielded F1 results equal or slightly better to those of simpler algorithms, but its extended execution times precluded its selection as the optimal choice.

Regarding the attributes that have been most relevant in the benchmark study, making a generalization is somewhat complex since each dataset collected data (attributes) in a different way. Moreover, some statistical attributes only present in BOT-IoT dataset were derived from relationships between several raw attributes. Despite this, no significant differences were found among the attribute selections made by the different feature selection (FS) techniques, except for the Correlation method. In other words, the attributes selected using different FS methods on each dataset were quite similar. In most experiments, we can highlight the prevalence of features such as the count of bytes from source to destination and the count of bytes from destination to source (two features with different names in each dataset but present in all of them). This is in agreement with the intrinsic nature of some attacks such e.g., DoS or DDoS launched in the IoT scenarios.

WrapperSubset and ClassifierSubset methods produced the best classifications in many cases, although they tailored the attribute subset selection for a specific ML technique and could be more adapted to them. Although FS is supposed to be a process carried out in offline mode prior to real-time traffic classification, it is necessary to caution about the long execution time of these subset FS algorithms during the feature selection phase and also ML model building. In general, results have shown that FFR and FSS methods can operate in real-time whereas WFS methods require much more time being unpractical in some specific FS/ML combinations. Consequently, it would be advisable to use faster Filter-type methods (since Wrappers are much slower), such as e.g., CFS, which yielded very good results in many of the experiments providing with very good selections of attributes subsets. On the other hand, FS methods that produce an attribute ranking according to a criterion optimization provide an ordered list of features based on their classification capability. However, the attributes selected through this rank mechanism may be highly correlated with each other, so selecting some of them jointly may not be appropriate. For example, TotPkts and TotBytes attributes in LATAM-DoS dataset are top ranked but contain highly correlated information and they should not appear in the same subset of attributes. Therefore, subset-based approaches seem to be more appropriate than attribute ranking-based approaches. In multiple cases evaluated, equal or even better F1 score results were obtained when using a reduced number of attributes, indicating that using a very high number of attributes is not always a guarantee of better classification.

In terms of execution times, all the tests conducted have demonstrated real-time functionality (without taking into account additional factors such as flow collection, attribute estimation, etc.). When the number of attributes is reduced using FS, execution times usually tend to decrease, but it obviously also depends on the ML algorithm considered. Even when using the same tree-based algorithm (e.g., J48), the execution time can increase with a more reduced number of attributes. This is because a very small number of attributes often leads to the construction of a more complex tree model and therefore, the time required to evaluate traffic flows becomes greater than if a simpler tree were used. Anyway, the use of a subset of relevant attributes is encouraged to achieve optimal

Table 6
F1 scores in TON-IoT for binary and multiclass classification.

Binary classification				F1 Score	
Attributes	FS method	ML technique	Flows/s	Normal	Attack
All (38)	None	RF	199.052	0.994	0.988
Src_Ip_Bytes	Classifier	OneR	2.612.567	0.930	0.875
Proto, Src_Ip_Bytes, Dst_Ip_Bytes	CFS	PART	412.510	0.985	0.973
Proto, Conn_State, Src_Ip_Bytes, Dst_Ip_Bytes, Dns_Rejected	Symmetrical Uncert	PART	423.659	0.990	0.981
Proto, Service, Src_Bytes, Dst_Bytes, Conn_State, Missed_Bytes, Src_Ip_Bytes, Dst_Pkts, Dst_Ip_Bytes, Dns_Rcode, Dns_AA, Dns_RD, Dns_Rejected	WrapperSubset	J48	1.959.425	0.993	0.986

Multiclass classification				F1 SCORE									
Attributes	FS method	ML technique	Flows/s	Normal	Scanning	DoS	Injection	DDoS	Password	XSS	Ransomware	Backdoor	MiTM
All (38)	None	J48	340.770	0.993	0.989	0.985	0.969	0.975	0.982	0.921	0.886	0.999	0.686
Src_Ip_Bytes	Classifier	OneR	382.327	0.928	0.967	0.644	0.763	0.780	0.852	0.764	0.537	0.991	0.031
Conn_State, Src_Ip_Bytes, Dst_Ip_Bytes	CFS	J48	783.770	0.987	0.986	0.979	0.956	0.929	0.978	0.909	0.865	0.999	0.596
Proto, Service, Duration, Src_Bytes, Dst_Bytes, Conn_State, Dst_Ip_Bytes, Src_Ip_Bytes, Dns_Qtype, Dns_Rcode, Dns_Rd, Dns_Ra, Ssl_Resumed	WrapperSubset	J48	559.836	0.993	0.989	0.985	0.968	0.978	0.980	0.923	0.888	0.999	0.710

classification rates.

When comparing binary and multiclass classifications, we can conclude that, in LATAM-DoS dataset, there is no significant difference in performance since almost perfect classification was reached in both cases. In LATAM-DDoS dataset, when analyzing the multiclass classification, the ML algorithms confused many HTTP attacks with normal connections. This also occurs in binary analysis, but there is a slight performance difference between binary and multiclass classifications. Similarly, both the binary and the multiclass classifier provided very accurate results in BOT-IoT dataset, with the binary one working much faster. Finally, in the TON-IoT dataset, the binary classifier performed better, as it has higher precision in its selections and does so more quickly. In general, in binary classification, execution times are faster than in multiclass classification (using the same ML technique). This aligns with a more complex classification problem in the latter case, although exceptions may be found due to differences in the complexity of the trees created in each test.

Finally, the results obtained in this work have been compared with those achieved by the creators of the processed datasets (see Table 7). We base the comparison on the F1 weighted scores between classes since these are the results provided in the referenced articles. In LATAM-DoS, the dataset creators did not perform attribute selection, but achieved an F1 score of 0,999 with all attributes using a Decision Tree technique [11], similar to our results using RF, PART or J48. Furthermore, we managed to reduce the attributes from 20 to 4 or 5 (in binary and multiclass classification, respectively) while maintaining the same classification outcomes for J48 algorithm. In LATAM-DDoS [11], attribute selection was also not performed in the dataset creator's study and they achieved an F1 score of 0.991 with the MLP technique and of 0,986 with a Decision Tree, in binary and multiclass classification, respectively. We reached similar F1 results applying the J48 technique (0,990 and 0,988, respectively, in binary and multiclass classification) using all the attributes or with only six features, thus greatly improving the execution speed. In BOT-IoT, the dataset designers conducted binary classification tests with and without feature selection, observing that the FS method worsened the F1 value from 0.999 to 0.938 [10]. They compared the SVM, RNN, and LSTM techniques achieving the best results with SVM. In multiclass classification, it is not clear how the comparative F1 results were obtained, since the tables in [10] suggest that each attack class was classified against normal traffic, in pairs, without simultaneous classification of all attack classes. In our case, quite similar F1 scores (0,999) where obtained using the PART algorithm with all the attributes or considering 5 or 12 attributes subsets for binary and multiclass classification, respectively. Nevertheless, we are surprised by the fact of including the Seq attribute in [10] both in the full feature set and in the 10 attributes subset selection. This could imply the inclusion of information related to the sequencing of traffic flows, enabling the identification of attacks and normal traffic. Finally, in TON-IoT, the dataset creators did not perform attribute selection and achieved their best result with the CART method, obtaining an F1 of 0.880 for the binary study and an F1 of 0.750 for the multiclass study [12]. However, they only used 22 attributes. In [13] using 40 attributes and the RF technique, they reached an F1 of 0,973 in binary classification. We have significantly improved these results using J48, achieving an F1 of 0.990 in the binary study and an F1 of 0.982 in the multiclass study, with only 13 attributes. As a general summary of this comparison, it can be said firstly that the work presented here shares with previous ones the high classification performance achieved by tree-based techniques. Algorithms such as e.g., J48 or PART, may offer a faster alternative solution to other ML techniques. Moreover, it is noteworthy that an appropriate selection of a subset of relevant attributes allows maintaining or even improving classification results with respect to the use of all the attributes present in the datasets.

7. Conclusions

Within this section, the main conclusions derived from the empirical study conducted are summarized.

Table 7
Performance results of related works in the datasets.

	Binary classification			Multiclass classification		
Dataset	N	FS / ML	F1	N	FS / ML	F1
LATAM-DoS						
Results from [11]	15	None / DT	0,999	15	None / DT	0,999
Proposed method	15	None / RF	0,999	15	None / RF	0,999
Proposed method	4	WrapperSubset / J48	0,999	5	ClassifierSubset / J48	0,999
LATAM-DDoS						
Results from [11]	15	None / MLP	0,991	15	None / DT	0,986
Proposed method	15	None / J48	0,990	15	None / J48	0,988
Proposed method	6	ClassifierSubset / J48	0,990	6	WrapperSubset / J48	0,988
BOT-IoT						
Results from [10]	35	None/SVM	0,999	–	–	–
Results from [10]	10	Correlation / SVM	0,938	–	–	–
Proposed method	32	None / PART	0,999	32	None / PART	0,999
Proposed method	5	CFS / PART	0,999	12	WrapperSubset / PART	0,999
TON-IoT						
Results from [12]	22	None / CART	0,880	22	None / CART	0,750
Results from [13]	40	None / RF	0,973	–	–	–
Proposed method	38	None / J48	0,991	38	None / J48	0,982
Proposed method	13	WrapperSubset / J48	0,990	13	WrapperSubset / J48	0,982

- Classification results: tree-based algorithms reached better classification results over other machine learning techniques (such as naive Bayes, multi-layer perceptron, logistic, adaptive boosting or k-Nearest Neighbors). They demonstrated a favorable balance between performance and execution time. J48 stands out due to its balanced performance in execution times and classification rates, often outperforming other techniques. However, in specific cases where J48 misclassified classes, alternative techniques like PART were relevant. RF technique showed high F1 results in some cases but at the expense of longer execution times. The superiority of tree-based techniques could also indicate that these methods have in some way adapted to the characteristics of the attacks present in the datasets, potentially resulting in non-generalizable models. However, they have still performed better than other machine learning techniques, even in the case of multi-class classification.
- Feature selection: the selection of attributes for classifying traffic flows in the examined datasets consistently yielded excellent results both in binary and multiclass classification. In some cases, even outperformed classifications obtained using the complete set of features while enhancing system efficiency. In any case, these excellent results should be taken with caution since they could reveal that the datasets were generated with traffic in which attacks exhibit similar characteristics and, at the same time, very different from normal traffic.
- Relevance of features in FS methods: identifying the most relevant attributes proves complex due to variations in data collection among different datasets. No universally applicable attribute list exists, demanding a case-specific analysis for effective selection. Despite this variability, the features that resulted more relevant across the four different datasets analyzed were: count of bytes from source to destination and vice versa, total number of bytes per source IP, number of inbound connections per destination IP, transport layer protocol of flow connection, and connection state. These attributes emerge as crucial in many experiments due to their association with specific attack types such as e.g., DoS in IoT scenarios. The attribute selection process should be adaptive, as technologies and threats continually evolve.
- Performance of FS methods: WrapperSubset and ClassifierSubset methods excel in various scenarios, tailoring attribute subsets for specific ML techniques. However, caution is necessary due to their lengthy execution times, making Filter-type FS methods like CFS preferable for their efficiency without sacrificing effectiveness in many tests. While FS methods produce attribute rankings based on classification capability, correlated attributes may lead to suboptimal subsets. For instance, attributes like TotPkts and TotBytes in the LATAM-DoS dataset, while highly ranked, contain redundant information and should not coexist in the same attribute subset. This reaffirms that subset-based approaches like CFS could be more suitable than attribute ranking-based approaches. CFS considers feature interactions while simultaneously selecting features that are highly correlated with the target variable but not highly correlated with each other. Moreover, it does not assume linearity in the data, rendering it more flexible and applicable to a variety of datasets and models. However, as a counterpart, CFS may exhibit a bias towards features highly correlated with the target variable. This bias might not be ideal if a more diverse and representative set of features is desired.
- The datasets employed in this study were recorded using experimental setups that may differ from real attack situations, potentially leading to F1 scores challenging to replicate in an actual operational environment. In this sense, the attributes characterizing traffic flows in real communication scenarios could reveal behaviors that are more intricate to distinguish than those shown in the datasets.

Overall, the study highlights the importance of careful attribute selection in enhancing classification outcomes while optimizing computational efficiency, and demonstrating the effectiveness of tree-based algorithms. The conclusions drawn from the empirical study provide valuable insights into the effectiveness and limitations of machine learning techniques for classifying network traffic flows and selecting relevant features. Building upon these findings, several lines for future research emerge. Investigating the augmentation and integration of datasets could facilitate the consolidation of relevant attributes under consideration. Thus, refining the methodology for calculating flow metric characteristics across databases could significantly enhance the comparability of results and contribute to a more robust feature selection process across diverse datasets. Additionally, given the potential limitations of experimental datasets in replicating real attack situations, future research will prioritize validation in real-world operational environments. This may involve conducting field studies using single board computers (such as Raspberry Pi), frequently deployed in IoT scenarios due to their reduced size and relatively high computing capacity.

CRediT authorship contribution statement

José García: Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Jorge Entrena:** Validation, Investigation, Formal analysis, Data curation. **Álvaro Alesanco:** Writing – review & editing, Validation, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jose Garcia reports financial support was provided by Spain Ministry of Science and Innovation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research was supported by Grant PID2022–136476OB-I00 funded by MICIU/AEI/10.13039/501100011033/FEDER EU, ERDF/EU, Gobierno de Aragón (reference group T31_20R), and University of Zaragoza (UZ2021-TEC-01).

Appendix. Attributes description for the datasets

Table 8
Attributes description in LATAM-DoS and LATAM-DDoS (according to [11]).

StartTime: Record start time	**Max:** Maximum duration at records aggregate level
TotPkts: Total number of packets in the transaction	**SrcPkts:** Source to destination packet count
TotBytes: Total number of bytes in the transaction	**DstPkts:** Destination to source packet count
LastTime: Record last time	**SrcBytes:** Source to destination bytes count
Seq: Argus sequence number	**DstBytes:** Destination to source bytes count
Dur: Record total duration	**Rate:** Total packets per second in transaction
Mean: Average duration at records aggregate level	**SrcRate:** Source to destination packets per second
StdDev: Standard deviation of duration at records aggregate level	**DstRate:** Destination to source packets per second
Sum: Total duration at records aggregate level	**Category:** Type of binary traffic, either Normal or Attack
Min: Minimum duration at records aggregate level	**Subcategory:** Type of multiclass traffic, either Normal or Attack (UDP, TCP, HTTP)

Table 9
Attributes description in BOT–IoT (according to [10]).

pkSeqID: Row Identifier	**Dpkts:** Destination-to-source packet count
Stime: Record start time	**Sbytes:** Source-to-destination byte count
Flgs: Flow state flags seen in transactions	**Dbytes:** Destination-to-source byte count
Flgs_number: Numerical representation of feature flags	**Rate:** Total packets per second in transaction
Proto: Textual representation of transaction protocols present in network flow	**Srate:** Source-to-destination packets per second
proto_number: Numerical representation of feature proto	**Drate:** Destination-to-source packets per second
Saddr: Source IP address	**TnBPSrcIP:** Total Number of bytes per source IP
Sport: Source port number	**TnBPDstIP:** Total Number of bytes per Destination IP
Daddr: Destination IP address	**TnP_PSrcIP:** Total Number of packets per source IP
Dport: Destination port number	**TnP_PDstIP:** Total Number of packets per Destination IP
Pkts: Total count of packets in transaction	**TNp_PerProto:** Total Number of packets per protocol
Bytes: Total number of bytes in transaction	**TnP_Per_Dport:** Total Number of packets per dport
State: Transaction state	**AR_P_Proto_P_SrcIP:** Average rate per protocol per Source IP (calculated by pkts/dur)
state_number: Numerical representation of feature state	**AR_P_Proto_P_DstIP:** Average rate per protocol per Destination IP
Ltime: Record last time	**N_IN_Conn_P_DstIP:** Number of inbound connections per destination IP
Seq: Argus sequence number	**N_IN_Conn_P_SrcIP:** Number of inbound connections per source IP
Dur: Record total duration	**AR_P_Proto_P_Sport:** Average rate per protocol per sport
Mean: Average duration of aggregated records	**AR_P_Proto_P_Dport:** Average rate per protocol per dport
Stddev: Standard deviation of aggregated records	**Pkts_P_State_P_Protocol_P_DestIP:** Number of packets grouped by state of flows and protocols per destination IP
Sum: Total duration of aggregated records	**Pkts_P_State_P_Protocol_P_SrctIP:** Number of packets grouped by state of flows and protocols per source IP
Min: Minimum duration of aggregated records	**Attack:** Class label: 0 for Normal traffic, 1 for Attack Traffic
Max: Maximum duration of aggregated records	**Category:** Traffic category
Spkts: Source-to-destination packet count	**Subcategory:** Traffic subcategory

Table 10
Attributes description in TON–IoT (according to [12]).

ts: Timestamp of connection between flow identifiers	**DNS_rejected:** DNS rejection, where the DNS queries are rejected by the server
Src_ip: Source IP addresses which originate endpoints' IP addresses	**Ssl_version:** SSL version which is offered by the server
Src_port: Source ports which Originate endpoint's TCP/UDP ports	**Ssl_cipher:** SSL cipher suite which the server chose
Dst_ip: Destination IP addresses which respond to endpoint's IP addresses	**Ssl_resumed:** SSL flag indicates the session that can be used to initiate new connections, where T refers to the SSL connection is initiated
Dst_port: Destination ports which respond to endpoint's TCP/UDP ports	**Ssl_established:** SSL flag indicates establishing connections between two parties, where T refers to establishing the connection
proto: Transport layer protocols of flow connections	**Ssl_subject:** Subject of the X.509 cert offered by the server

(*continued on next page*)

Table 10 (*continued*)

service: Dynamically detected protocols, such as DNS, HTTP and SSL	**Ssl_issuer:** Trusted owner/originator of SLL and digital certificate (certificate authority)
duration: The time of the packet connections, which is estimated by subtracting 'time of last packet seen' and 'time of first packet seen'	**Http_trans_depth:** Pipelined depth into the HTTP connection
Src_bytes: Source bytes which are originated from payload bytes of TCP sequence numbers	**http_method:** HTTP request methods such as GET, POST and HEAD
Dst_bytes: Destination bytes which are responded payload bytes from TCP sequence numbers	**http_uri:** URIs used in the HTTP request
Con_state: Various connection states, such as S0 (connection without replay), S1 (connection established), and REJ (connection attempt rejected)	**http_version:** The HTTP versions utilised such as V1.1
Missed_bytes: Number of missing bytes in content gaps	**http_request_body_len:** Actual uncompressed content sizes of the data transferred from the HTTP client
Src_pkts: Number of original packets which is estimated from source systems	**http_response_body_len:** Actual uncompressed content sizes of the data transferred from the HTTP server
Src_ip_bytes: Number of original IP bytes which is the total length of IP header field of source systems	**http_status_code:** Status codes returned by the HTTP server
Dst_pkts: Number of destination packets which is estimated from destination systems	**http_user_agent:** Values of the User-Agent header in the HTTP protocol
Dst_ip_bytes: Number of destination IP bytes which is the total length of IP header field of destination systems	**http_orig_mime_types:** Ordered vectors of mime types from source system in the HTTP protocol
Dns_query: Domain name subjects of the DNS queries	**http_resp_mime_types:** Ordered vectors of mime types from destination system in the HTTP protocol
Dns_qclass: Values which specifies the DNS query classes	**Weird_name:** Names of anomalies/violations related to protocols that happened
Dns_qtype: Value which specifies the DNS query types	**Weird_addl:** Additional information is associated to protocol anomalies/ violations
Dns_rcode: Response code values in the DNS responses	**Weird_Notice:** It indicates if the violation/anomaly was turned into a notice
Dns_AA: Authoritative answers of DNS, where T denotes server is authoritative for query	**label:** Tag normal and attack records, where 0 indicates normal and 1 indicates attacks
Dns_RD: Recursion desired of DNS, where T denotes request recursive lookup of query	**type:** Tag attack categories, such as normal, DoS, DDoS and backdoor attacks, and normal records
DNS_RA: Recursion available of DNS, where T denotes server supports recursive queries	

References

[1] A. Rullo, E. Bertino, K. Ren, Guest editorial special issue on intrusion detection for the internet of things, IEEe Internet. Things. J. 10 (10) (2023) 8327–8330, https://doi.org/10.1109/JIOT.2023.3244636, 15 May15,.

[2] H. HaddadPajouh, A. Dehghantanha, R.M. Parizi, M. Aledhari, H. Karimipour, A survey on internet of things security: Requirements, challenges, and solutions, Internet of Things 14 (2021) 100129, https://doi.org/10.1016/j.iot.2019.100129.

[3] Y. Yang, L. Wu, G. Yin, L. Li, H. Zhao, A survey on security and privacy issues in internet-of-things, IEEe Internet. Things. J. 4 (5) (Oct. 2017) 1250–1258, https://doi.org/10.1109/JIOT.2017.2694844.

[4] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, E. Vázquez, Anomaly-based network intrusion detection: Techniques, systems and challenges, Comput. Secur. 28 (2009) 18–28.

[5] B. Bogaz Zarpelão, R. Sanches Miani, C. Toshio Kawakani, S. Carlisto de Alvarenga, A survey of intrusion detection in internet of things, J. Netw. Comput. Appl. 84 (2017) 25–37, https://doi.org/10.1016/j.jnca.2017.02.009.

[6] K.A.P. da Costa, J.P. Papa, C.O. Lisboa, R. Munoz, V.H.C. de Albuquerque, Internet of things: A survey on machine learning-based intrusion detection approaches, Comput. Netw. 151 (2019) 147–157, https://doi.org/10.1016/j.comnet.2019.01.023.

[7] M. Abdullahi, Y. Baashar, H. Alhussian, A. Alwadain, N. Aziz, L.F. Capretz, S.J. Abdulkadir, Detecting cybersecurity attacks in internet of things using artificial intelligence methods: A systematic literature review, Electronics. (Basel) 11 (2022) 198, https://doi.org/10.3390/electronics11020198.

[8] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, A. Hotho, A survey of network-based intrusion detection data sets, Comput. Secur. 86 (2019) 147–167.

[9] A. Thakkar, R.R. Lohiya, A review of the advancement in intrusion detection datasets, Procedia Comput. Sci. 167 (2020) 636–645.

[10] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, Benjamin Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset, Future Gener. Comput. Syst. 100 (2019) 779–796.

[11] J.G. Almaraz-Rivera, J.A. Perez-Diaz, J.A. Cantoral-Ceballos, J.F. Botero, L.A. Trejo, Toward the protection of IoT networks: Introducing the LATAM-DDoS-IoT dataset, IEEe Access. 10 (2022) 106909–106920, https://doi.org/10.1109/ACCESS.2022.3211513.

[12] Abdullah Alsaedi, Nour Moustafa, Zahir Tari, Abdun Mahmood, Adnan Anwar, TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems, IEEe Access. 8 (2020) 165130–165150.

[13] T.M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, F.T.H. d. Hartog, ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets, IEEe Internet. Things. J. 9 (1) (2022) 485–496, https://doi.org/10.1109/JIOT.2021.3085194, 1 Jan.1.

[14] S. Dadkhah, H. Mahdikhani, P.K. Danso, A. Zohourian, K.A. Truong, A.A. Ghorbani, Towards the development of a realistic multidimensional IoT profiling dataset, in: 19th Annual International Conference on Privacy, Security & Trust (PST), Fredericton, NB, Canada 2022, 2022, pp. 1–11, https://doi.org/10.1109/PST55820.2022.9851966.

[15] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, E. Cambiaso, MQTTset, A new dataset for machine learning techniques on MQTT, Sensors 20 (2020) 6578, https://doi.org/10.3390/s20226578.

[16] M. Kubat, An Introduction to Machine Learning, Springer International Publishing, Berlin/Heidelberg, Germany, 2021, https://doi.org/10.1007/978-3-030-81935-4.

[17] J. Ross Quinlan, Programs For Machine Learning, Kaufmann Publishers, Burlington, MA, USA, 1994.

[18] E. Frank, I.H. Witten, Generating accurate rule sets without global optimization, in: ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, 24–27 July.

[19] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[20] R.C. Holte, Very simple classification rules perform well on most commonly used datasets, Mach. Learn. 11 (1993) 63–90.

[21] I. Guyon, S. Gunn, M. Nikravesh, L.A. Zadeh, Feature Extraction: Foundations and Applications. Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer, Berlin/Heidelberg, Germany, 2006.
[22] S. Khalid, T. Khalil, S. Nasreen, A Survey of Feature Selection and Feature Extraction Techniques in Machine Learning, in: Proceedings of the 2014 Science and Information Conference (SAI), IEEE: Piscataway, NJ, USA, London, UK, 2014, pp. 372–378, 27–29 August.
[23] Y.B. Wah, N. Ibrahim, H.A. Hamid, S. Abdul-Rahman, S. Fong, Feature selection methods: Case of filter and wrapper approaches for maximising classification accuracy, Pertanika J. Sci. Technol. 26 (2018) 329–340.
[24] M.U. Ilyas, S.A. Alharbi, Machine learning approaches to network intrusion detection for contemporary internet traffic, Computing 104 (2022) 1061–1076, https://doi.org/10.1007/s00607-021-01050-5.
[25] M. Rodríguez, Á. Alesanco, L. Mehavilla, J. García, Evaluation of machine learning techniques for traffic flow-based intrusion detection, Sensors 22 (2022) 9326.
[26] S. Ullah, Z. Mahmood, N. Ali, T. Ahmad, A. Buriro, Machine learning-based dynamic attribute selection technique for DDoS attack classification in IoT networks, Computers 12 (2023) 115, https://doi.org/10.3390/computers12060115.
[27] A.O. Balogun, S. Basri, S.J. Abdulkadir, A.S. Hashim, Performance analysis of feature selection methods in software defect prediction: A search method approach, Appl. Sci. 9 (2019) 2764, https://doi.org/10.3390/app9132764.
[28] A.O. Balogun, S. Basri, S. Mahamad, S.J. Abdulkadir, M.A. Almomani, V.E. Adeyemo, Q. Al-Tashi, H.A. Mojeed, A.A. Imam, A.O. Bajeh, Impact of feature selection methods on the predictive performance of software defect prediction models: An extensive empirical study, Symmetry. (Basel) 12 (2020) 1147, https://doi.org/10.3390/sym12071147.
[29] M.A. Hall, Correlation-Based Feature Selection For Machine Learning. Doctoral Dissertation, University of Waikato, Ham-ilton, New Zealand, 1999.
[30] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed., Morgan Kaufmann, San Francisco, CA, USA, 2005.
[31] E. Frank, M.A. Hall, I.H. Witten, WEKA Workbench Online Appendix For "Data mining: Practical Machine Learning Tools and Techniques, 4th Edition, Morgan Kaufmann, 2016.