Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Which algorithm can detect unknown attacks? Comparison of supervised, unsupervised and meta-learning algorithms for intrusion detection

Tommaso Zoppi*, Andrea Ceccarelli, Tommaso Puccetti, Andrea Bondavalli

Department of Mathematics and Informatics, University of Florence, Viale Morgagni 65, Florence 50142, Italy

A R T I C L E I N F O

Article history:
Received 4 August 2022
Revised 16 November 2022
Accepted 14 January 2023
Available online 18 January 2023

Keywords:
Intrusion detection
Machine learning
Unknown attacks
Unsupervised
Meta-Learning
Zero-Day attacks

A B S T R A C T

There is an astounding growth in the adoption of machine learners (MLs) to craft intrusion detection systems (IDSs). These IDSs model the behavior of a target system during a training phase, making them able to detect attacks at runtime. Particularly, they can detect known attacks, whose information is available during training, at the cost of a very small number of false alarms, i.e., the detector suspects attacks but no attack is actually threatening the system. However, the attacks experienced at runtime will likely differ from those learned during training and thus will be unknown to the IDS. Consequently, the ability to detect unknown attacks becomes a relevant distinguishing factor for an IDS. This study aims to evaluate and quantify such ability by exercising multiple ML algorithms for IDSs. We apply 47 supervised, unsupervised, deep learning, and meta-learning algorithms in an experimental campaign embracing 11 attack datasets, and with a methodology that simulates the occurrence of unknown attacks. Detecting unknown attacks is not trivial: however, we show how unsupervised meta-learning algorithms have better detection capabilities of unknowns and may even outperform classification performance of other ML algorithms when dealing with unknown attacks.

1. Introduction

It is widely acknowledged that modern ICT systems such as industrial control systems (Cruz et al., 2015), medical support systems (Dey et al., 2018), virtual environments (Cotroneoet al., 2017), and the Internet of Things (Akyildiz and Kak, 2019) can be the target of attackers (ABC n.d.; Chou and Jiang, 2021; Buczak and Guven, 2015). There is significant evidence on the risk of cyberattacks, both in terms of the likelihood of being targeted and the cost and impact of a successful attack.

The number of computer security incidents has been steadily growing over the past few years: in 2021, SonicWall (Connell, 2022) reported an average of 28 million cyberattacks detected daily, with 140 000 of them being novel malware samples. Starting from 2020, the European Union Agency for Cybersecurity (ENISA) observed a spike in nonmalicious incidents, most likely because the COVID-19 pandemic became a multiplier for human errors and system misconfigurations, and attributed them as the root cause for the majority of security breaches (Ardagna et al., 2021).

The consequence of a successful cyberattack (simply termed an *attack* from now on) may range (Avizienis et al., 2004) from confidentiality issues to availability reduction or the loss of sensitive data and thus integrity concerns. Importantly, security threats may also have a safety impact; for example, an attack that aims at making the automatic braking system of a vehicle unavailable may also have severe impacts on the health of the driver, the infrastructures and the surrounding environment. Consequently, systems must be conceptualized, designed, and implemented to ensure that appropriate security requirements are met. Among the possible countermeasures, the detection of ongoing attacks is typically included in the mandatory security requirements (Chou and Jiang, 2021; Buczak and Guven, 2015).

Intrusion detection systems (IDSs) are well-known means to promptly detect attacks. Given a target system to protect, an IDS monitors its performance indicators: examples include memory usage (Cotroneoet al., 2017), throughput of buses (Cruz et al., 2015), active sessions (T. Zoppi et al., 2019), and system calls (Al and Dener, 2021). The set of monitored values (i.e., features) gathered by an IDS at a given instant is called a *data point*: collections of data points are typically collected in the form of tabular datasets.

* Corresponding author.
E-mail address: tommaso.zoppi@unifi.it (T. Zoppi).

An IDS contains a machine learning (ML) algorithm that performs binary classification (Zhang et al., 2022; Erickson et al., 2020) (i.e., it is a binary classifier) discerning between data points corresponding to an attack and data points corresponding to the normal behavior of a system. The ML algorithm undergoes a training phase in which it processes a training dataset: it learns a model, which at a later stage will be deployed in the production environment to detect attacks occurring at runtime.

Unfortunately, data points collected in the production environment may differ from the data points in the training dataset. This is a very frequent scenario for two reasons. First, systems are becoming increasingly more complex and dynamic, committing updates and reconfigurations. Consequently, training may quickly become obsolete (Casas et al., 2012) and reduce the effectiveness of the model learned. Second, during its operational life, a system may be targeted by attacks that were not known at training time, which we call *unknown attacks*. Unknown attacks are significant threats, with the same effect as zero days (ABC n.d.; T. Zoppi et al., 2021), i.e., new attacks or variations of existing attacks that are specifically created to exploit new vulnerabilities. It is credible that during its life, a system will be the target of unknown attacks (T. Zoppi et al., 2021; Ardagna et al., 2021; Connell, 2022; Chou and Jiang, 2021); therefore, IDSs must be prepared to deal with them to avoid major security issues.

This paper reviews classifiers for intrusion detection, evaluating their capability to detect unknown attacks. We organize classifiers into five categories: unsupervised (UNS), supervised (SUP), deep learning (DEEP), supervised meta-learning (META-SUP), and unsupervised meta-learning (META-UNS). UNS classifiers do not use labels during training, i.e., they are not aware of whether a data point is an attack. In contrast, SUP and DEEP classifiers need labeled data points during training. The SUP and DEEP classifiers are both supervised classifiers, and the latter uses deep neural networks. We apply this distinction because the literature on ML for tabular data shows different classification performance between deep neural networks and other supervised classifiers. Often, DEEP classifiers are considered to perform worse than SUP on tabular data (Shwartz-Ziv and Armon, 2022; Gorishniy et al., 2021). Last, META-SUP and META-UNS are supervised and unsupervised classifiers, respectively, that employ meta-learning: meta-learning uses knowledge acquired during base-learning episodes, i.e., meta-knowledge, to improve classification capabilities at the meta-level (Brazdil et al., 2009). META-SUP classifiers require labeled data for training, while META-UNS classifiers do not.

We exercise a total of 47 classifiers on 11 public attack datasets, which we manipulate to simulate the occurrence of unknown attacks. Very briefly, the procedure is as follows. Some attacks are removed from the training datasets and used only for testing, which makes them unknown attacks. The whole procedure is repeated for all the attack categories and all the datasets. We analyze and compare the detection performance when the number of unknown attacks increases, and we explain which classifiers are more suited to detect unknown attacks. Our analysis reveals the following:

- Classifiers suffer the introduction of unknown attacks, either because unknowns are undetected (especially when using DEEP and SUP classifiers) or because many false alarms are raised (this mostly occurs with UNS).
- SUP, META-SUP, and, to a lesser extent, DEEP classifiers are effective in detecting known attacks, but their detection performance drops significantly when unknown attacks occur.
- Instead, UNS classifiers have better detection performance of unknown attacks but are clearly outperformed by DEEP, SUP, and META-SUP when dealing with known attacks.
- Meta-learning enhances the classification performance of both SUP and UNS classifiers. Most noticeably and contrary to common knowledge, META-UNS classifiers based on bagging (Breiman, 2001) and boosting (Rätsch et al., 2001) ensembles improve detection performance to a point at which they are slightly worse than SUP, META-SUP, and DEEP classifiers against known attacks but have superior ability to detect unknown attacks.

The rest of the paper is structured as follows. Section 2 reports background notions that are used in the subsequent sections. Section 3 reviews the related literature, describing the differences from our work. Section 4 illustrates the methodology we applied. Section 5 presents our experimental results. Section 6 discusses threats to validity. Section 7 concludes the paper.

2. Intrusion detection

Recent trends (Zhang et al., 2022; He et al., 2017; Chou and Jiang, 2021) show how data-driven detection may provide a means to detect intrusions, as opposed to traditional rule- and signature-based mechanisms. Data-driven detectors are usually implemented as binary classifiers (simply called *classifiers* in the rest of the paper) that assign either a positive or a negative class to each data point. Typically, they act as *anomaly detectors*: they identify patterns that do not conform to a well-defined notion of normal behavior (Chandola et al., 2009), and these patterns are the symptom of intrusions or upcoming failures. The negative class is mapped to the normal class (no attack), whereas the positive class is known as the anomaly (or attack) class.

2.1. Machine learning and intrusion detection

Traditionally, the vast majority of anomaly detectors for tabular data are implemented through SUP and DEEP classifiers (Liao and Vemuri, 2002; Zhao et al., 2019). They require training data for which the label is known and build a model that is usually accurate, i.e., only a few misclassifications occur. SUP are usually partitioned into 4 families: i) tree-based, mostly decision trees, ii) statistical techniques (Srivastava et al., 2007), iii) distance-based learners (Liao and Vemuri, 2002), and iv) support vector machines (Hearst et al., 1998).

DEEPs are neural network classifiers structured with many hidden layers and are referred to as deep neural networks or deep learners (Zhang et al., 2022; Li et al., 2021; LeCun et al., 2015). Deep learners are the de facto standard for classifying unstructured data such as images, audio, lidar point clouds, and videos; however, they often struggle when classifying tabular data. For instance, recent works (Zhang et al., 2022; Shwartz-Ziv and Armon, 2022; Li et al., 2021; LeCun et al., 2015) advocate that deep neural network classifiers applied to tabular data have worse classification performance than other supervised classifiers. There are also studies that convert tabular data into images to fully exploit the potential of deep learners in processing images (Zhu et al., 2021), but classification performance does not benefit much. Research efforts have escalated in proposing libraries such as AutoGluon (Erickson et al., 2020), FastAI (Howard and Gugger, 2020), and TabNet (Nishida et al., 2017), which provide implementations of deep learners to process tabular data. However, a benchmark of DEEP classifiers for tabular data against other intrusion detectors has yet to be published.

Intrusion detectors can also be implemented using UNS classifiers, which do not require labels in the training data: they build their model under the assumption that ongoing attacks manifest as observable deviations from the nominal behavior. The implications are twofold: on the positive side, it makes UNS classifiers more capable of detecting unknown attacks than supervised classifiers, either SUP or DEEP. On the downside, they usually generate a higher number of misclassifications – especially false positives – than supervised classifiers (Sathya and Abraham, 2013; Lee et al., 2005). Clustering (Hamerly and Elkan, 2004; Amer and Goldstein, 2012) is probably the most widespread among all families of UNS classifiers, although statistical (Goldstein and Dengel, 2012), angle (Kriegel and Zimek, n.d.), density (Vázquez et al., 2018), unsupervised variants of neural networks (Kohonen, 1997), neighbor-based (Hautamaki et al., 2004), and other (Liu et al., 2008) classifiers have been proven to be valid alternatives.

In addition, *meta-learning* is the study of methods that exploit knowledge acquired during base-level learning to build a strong meta-level learner (T. Zoppi et al., 2021). A base-learning process feeds dataset features into many classifiers to be used for classification at the first stage: those are called base learners. Outputs of base learners are orchestrated together to compute the classification result of the whole meta-learner. On the downside, the complexity of the classification problem usually increases when adopting meta-learning because it requires training and running multiple classifiers.

Various meta-learners, namely, bagging, boosting, stacking, cascading, delegating, voting, and arbitrating, are summarized in (T. Zoppi et al., 2021; van Rijn et al., 2015). Bagging creates base-learners as instances of the same classifier that are trained using different bootstrap replicas of the training set (Breiman, 2001). The unified result of the ensemble is derived by majority voting the individual results of base learners. Instead, boosting builds ensembles of weak learners, which are instances of the same classifier that undergo a quick training process using a small subset of the training set. The detection performance of a single weak learner is poor; however, the proper composition of several weak learners builds a strong meta-learner (Rätsch et al., 2001; Chen and Guestrin, 2016). Similar to bagging, the meta-level output is obtained through majority voting of the outputs of individual weak learners. Bagging and boosting meta-learners are widely used for supervised learning in the form of random forests (Breiman, 2001) (bagging) and ADABoost (Rätsch et al., 2001), gradient boosting, and XGBoost (Chen and Guestrin, 2016) (boosting), respectively. They also show promising applications to unsupervised learning (T. Zoppi et al., 2021). Other meta-learners, i.e., voting, cascading, stacking, delegating, and arbitrating, employ a heterogeneous set of classifiers and have many possible configurations. Consequently, we are not considering them further in this study.

Overall, we consider two groups of meta-learning classifiers for intrusion detection: META-SUP classifiers, which are bagging and boosting meta-learners whose base learners are supervised, and META-UNS classifiers, which instead use unsupervised base learners.

2.2. Features and monitored performance indicators

Features provide the knowledge that makes ML algorithms learn models and perform classification. For intrusion detection, features are usually obtained by monitoring the performance indicators at the hardware or low level (do Nascimento et al., 2021), system level (T. Zoppi et al., 2019; Al and Dener, 2021), input/sensor (Robles-Velasco et al., 2020), environment (Cotroneoet al., 2017), application level (e.g., SCADA (Cruz et al., 2015)) or even coding level (Li et al., 2021). Features can be textual or numeric: textual features (e.g., the name of a protocol) are always categorical, while numeric features may either be categorical = (e.g., the ID of a system call) or continuous, describing a continuous ordinal range of values, e.g., the percentage of memory used, or the number of packets received from the network interface in a time frame.

Categorical features usually require preprocessing before being fed to a classifier. In fact, classifiers may compute algebraic calculus such as Euclidean distance (Liao and Vemuri, 2002) or the estimation of angles in a multidimensional space (Kriegel and Zimek, n.d.), which delivers misleading results when processing categorical features. This can be easily understood considering our previous examples: there is no meaning in computing the distance between the IDs of system calls or between the names of network protocols. Commonly, categorical features are preprocessed using either one-hot encoding (Rodríguez et al., 2018) or entity embedding (Guo and Berkhahn, 2016). One-hot encoding replaces a categorical feature with new features, one for each possible value of the categorical feature. Given a data point, each of the new features is set to 0, except for the feature that corresponds to the value in the data point, which is set to 1 (Rodríguez et al., 2018). In other words, each categorical feature is converted into an array of features: only one of the new features has a value of 1, leaving all the others at 0. Entity embedding represents categorical values in a continuous way, aiming to retain the relationship between different data values. Each value (entity) of a categorical feature is represented by a vector of floating-point numbers. This avoids creating sparse matrices, which are the likely result of one-hot encoding.

2.3. Metrics to evaluate intrusion detectors

The classification performance of intrusion detectors is typically expressed using metrics (Chicco and Jurman, 2020) that compute correct classifications, True Positives (TPs) and True Negatives (TNs), and misclassifications, False-Positives (FPs) and False Negatives (FNs). These metrics can be aggregated into a wide variety of compound metrics (Boughorbel et al., 2017). In this paper, we will use the following.

Accuracy (ACC) calculates the percentage of correct classifications (TPs and TNs) over all classifications. Importantly, 1 - ACC is usually referred to as the misclassification rate.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

$$1 - ACC = \frac{FP + FN}{TP + TN + FP + FN}$$

The *Matthews correlation coefficient* (MCC) (Boughorbel et al., 2017) is particularly suited when the dataset is unbalanced (Chicco and Jurman, 2020), i.e., the numbers of normal data points and anomalous data points are significantly different. This situation occurs quite frequently in the security domain, where normal data points are many and easy to collect, but only a few anomalies are present.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Recall (REC) (Campos et al., 2016) is an FN-oriented metric that shows the fraction of detected attacks out of all attacks in a given dataset. It is also called coverage.

$$REC = \frac{TP}{TP + FN}$$

3. Related works

The ever-growing demand for security mechanisms has led to the design of many intrusion detectors that embed ML-based classifiers. This is repeatedly confirmed in the literature; for example, the reader may refer to recent surveys on intrusion detection systems that employ ML (Khraisat et al., 2019; Chou and Jiang, 2021; Buczak and Guven, 2015).

In this paper, we carefully review the related works based on the following three subjects.

- *Classifiers for intrusion detection*: We seek the most relevant applications of supervised (SUP, DEEP) and unsupervised UNS classifiers for intrusion detection.
- *Performance in the case of unknown attacks*: We review the most relevant strategies to evaluate intrusion detectors in the presence of unknown attacks and the results.
- *Meta-learning for detecting intrusions*: We discuss the most recent applications of META-SUP and META-UNS to intrusion detection and the results achieved.

3.1. Classifiers for intrusion detection

Several works argue for the successful usage of SUP classifiers for intrusion detection. The study in (Chkirbene et al., 2020) compares 6 tree-based supervised classifiers for detecting intrusions on the UNSW-NB15 (Moustafa and Slay, 2015) dataset, concluding that decision trees with pruning often outperform other approaches (random forests and J48, among others). The authors of (Taher et al., 2019) compare support vector machines (SVMs) and a deep convolutional neural network (CNN). They conclude that selecting features helps increase classification accuracy on the NSL-KDD dataset and that the deep CNN outperforms SVM by a fair amount. Additionally, the work in (Vinayakumar et al., 2019) evaluates the performance of deep learning models on six datasets.

At the same time, recent studies (Zhang et al., 2022; Shwartz-Ziv and Armon, 2022; Li et al., 2021; LeCun et al., 2015) question the efficiency of DEEP classifiers in processing tabular data, showing how they often underperform with respect to SUP classifiers. Particularly, the work in (Shwartz-Ziv and Armon, 2022) compares 4 state-of-the-art deep learners for tabular data against a nonneural network classifier and shows that the latter has better classification performance than the former. In our study, we separate SUP from DEEP, and we evaluate both on a wide variety of datasets for intrusion detection, unraveling doubts about their detection performance.

3.2. Addressing unknown attacks

The papers reviewed in Section 3.1 analyze the behavior of IDSs under the condition that all attacks are known at training time, and the IDSs are evaluated against the same attack categories that have been used for training. As acknowledged in many works, (Khraisat et al., 2019; T. Zoppi et al., 2021; Liao and Vemuri, 2002; T. Zoppi et al., 2021; Catillo et al., 2022), a system may be targeted by unknown attacks during its operational life. Detecting those attacks is fundamental for the security of the target system; therefore, it is of utmost importance to quantify to what extent an IDS can detect unknown attacks.

In (Khraisat et al., 2019), the authors review existing datasets and classifiers for attack detection. Among other findings, they conclude that classifiers may have the problem of generating and updating the information about new attacks, and they may raise many false alarms or have poor accuracy. Similarly, the study (Liao and Vemuri, 2002) applies the k-th nearest neighbor (kNN) classifier to a dataset of system calls and measures a clear degradation in the detection performance of kNN, whose Recall drops from 100% to 75% in the presence of unknown attacks. The studies (T. Zoppi et al., 2021; T. Zoppi et al., 2021) measure the impact that zero-day attacks have on UNS classifiers and conclude that zero-day attacks have a limited impact on the detection performance. Last, the work in (Catillo et al., 2022) trains different ML algorithms with CICIDS17 (Sharafaldin et al., 2018) and evaluates them using an unseen dataset, measuring a clear drop in accuracy.

To summarize, the occurrence of unknown attacks has a noticeable impact on the detection performance. The performance degradation is less evident in the case of UNS rather than SUP. However, the extent and performance of SUP and UNS are still not unanimous. One of the goals of our study is to clearly state the performance differences between UNS and SUP in the case of both known and unknown attacks.

3.3. Meta-Learning for detecting intrusions

In general, bagging and boosting meta-learners are commonly and successfully used in supervised learning (Breiman, 2001; Rätsch et al., 2001). Especially in the case of bagging, a whole survey (Resende and Drummond, 2018) recaps the last 20 years of research works that used random forests to detect intrusions.

Most recently, it has been proven that meta-learning has the potential to reduce misclassifications of UNS classifiers (T. Zoppi et al., 2021). The study (T. Zoppi et al., 2021) applies a total of 8 different meta-learners that use unsupervised base learners, showing how bagging and boosting ensembles of UNS classifiers significantly reduce misclassifications. However, the work focuses only on unsupervised classifiers and does not specifically challenge the detection of unknown attacks.

No other works are identified that exploit meta-learning to build intrusion detectors. In the adjacent domain of error detection, the work in (Medico et al., 2018) evaluates the application of meta-learning. The authors investigated whether a kNN and a gradient boosting supervised meta-learner can detect anomalies in transient susceptibility tests. Lighting anomalies define dangerous operation conditions for machinery, and gradient boosting ended up detecting a significant number of anomalies with few missed detections, with a very high recall and low false-positive rate. From a different perspective, the work in (Zhao et al., 2019) pairs gradient boosting with feature preprocessing to build detectors that are robust against unreliable labels due to careless annotations or malicious data transformation.

To the best of our knowledge, our study is the first to compare supervised and unsupervised classifiers with and without meta-learning (i.e., SUP, DEEP, UNS, META-SUP, META-UNS) to detect known and unknown attacks. In this way, we provide a solid evaluation of the detection performance of all the major families of classifiers according to the same experimental methodology.

4. Experimental plan

This section details the experimental setup to compare the detection performance of supervised and unsupervised classifiers, with and without meta-learning, addressing known and unknown attacks.

4.1. Methodology to execute experiments

We designed and performed a quantitative evaluation organized into steps M1 to M5.

M1 We collect public datasets containing data about intrusion detection. These datasets contain features collected by monitoring real or simulated systems during their normal operation and when they are under attack. In total, we identified 11 datasets, which we describe in Section 4.2 and Table 1.

M2 We preprocess each dataset to obtain tabular CSV files (see Section 4.3). Each row of the CSV file represents a data point, and each column represents a feature, except for the last column, which is the binary label (normal/attack) and describes whether a row corresponds to an attack or the normal operation.

M3 Preprocessed datasets are used to generate training variants according to the procedure in Section 4.4. Very briefly, given a dataset split into a training set and a test set and one attack category, we remove all the attacks of such categories

Table 1
Selected datasets: name, release year, size, number of ordinal and categorical features, number and percentage of attacks, training variants.

Name	Year	# Data Points	Features		Attacks		TrainingVariants
			Ord.	Cat.	#	%	
ADFANet	2015	132 002	5	6(0)	3	11.3	3
AndMal17	2017	100 000	77	5(0)	4	15.5	4
CICIDS17	2017	500 000	77	5(1)	5	79.7	5
CICIDS18	2018	200 000	77	5(1)	8	26.2	8
CIDDS	2015	400 000	5	7(2)	4	14.4	4
IoT-IDS	2019	210 425	8	1(1)	8	42.3	8
ISCX12	2012	600 000	4	10(3)	4	43.5	4
NSLKDD	2009	148 516	37	5(3)	4	40.7	4
SDN20	2020	205 167	63	5(1)	5	66.6	5
UGR16	2016	207 256	4	6(2)	5	3.3	5
UNSW-NB15	2015	165 461	38	6(5)	8	6.5	8

from the training set; this way, we obtain a *training variant*. We repeat this procedure for all 11 training sets and all attack categories contained in each dataset. In total, we obtain 58 training variants. Classifiers are trained on the 11 training sets and the 58 training variants. The 11 training sets include all the attacks that are in the test sets, meaning that all attacks are known by the classifier. The training variants miss one attack category each, which instead appears in the test sets; this is equivalent to having unknown attacks.

M4 Afterward, we select classifiers from the categories SUP, DEEP, UNS, META-SUP, and META-UNS. In total, we select 6 SUP, 4 DEEP, 11 UNS, 4 META-UNS, and 22 META-UNS classifiers. We train each of the 47 classifiers on each of the 11 training sets and each of the 58 training variants. The classifiers and their parameters are explained in Section 4.5.

M5 Finally, we collect the metric scores of all classifiers, and we create tables and plots to drive discussions and analyses. Our experimental setup, metrics, and tools used in the paper are reported in Section 4.6.

4.2. M1 - dataset collection

There is a wide variety of tabular datasets related to intrusion detection, ranging from device data in Internet of Things (IoT) systems to network data for intrusion detection. Among many alternatives, we seek datasets that include more than one attack in addition to normal data. This is necessary to apply step M3.

We select 11 datasets of network intrusion detection. The datasets are identified by consulting recent surveys on datasets (Khraisat et al., 2019; Ring et al., 2019) and querying online portals. Our selection process resulted in the following datasets: NSLKDD (Tavallaee et al., 2009), ISCX12 (Shiravi et al., 2012), UNSW-NB15 (Moustafa and Slay, 2015), UGR16 (Maciá-Fernández et al., 2018), ADFA-Net (Haider et al., 2017), AndMal17 (Lashkari et al., 2018), CIDDS001 (Ring et al., 2017), CICIDS17 (Sharafaldin et al., 2018), CICIDS18 (Sharafaldin et al., 2018), IoT-IDS (Kang et al., 2019) and SDN20 (Elsayed et al., 2020). Table 1 summarizes the datasets considered in this study, reporting domain, name, publication year, number of data points, ordinal and categorical features, types, and percentages of attacks. The Training Variants column indicates the number of variants that are created in step M3. The datasets are distributed in the period from 2009 to 2020, and they are well-known reference datasets in the domain.

4.3. M2 - dataset processing

The selected datasets have heterogeneous structures; for instance, ISCX12, IoT-IDS, and UNSW-NB15 are available only as a collection of network packets stored as PCAP files. Consequently, we first converted the PCAP files into CSV files: this way, all 11 datasets had the same tabular structure.

Then, we analyzed all datasets to remove the following features when present: IP address, port number, timestamp, and ID number (T. Zoppi et al., 2021). In the majority of cases, these features can mislead the analysis. For example, let us consider the IP address. The intrusion detector learns from the training set that a specific IP is malicious, and it classifies all the data points that contain such IPs in the test set as attacks. This would result in a high attack detection rate, but it is unrealistic because the intrusion detector cannot assume to know from which IP address attackers will attack. Consequently, it should not use the information on the IP for training: it describes a very specific setup of the system used to generate the dataset and does not provide useful information on the system behavior.

Last, we removed duplicated label columns (if any), zero-filled all blank values resulting from the conversion from PCAP to CSV files, and split each dataset: 50% for training and 50% for testing.

4.4. M3 - creation of training variants

We create training variants of each dataset as follows. First, we remind that an attack category that does not appear in the training set is unknown to the classifier.

For this reason, we remove specific attack categories from the training set of each dataset: this way, we create as many training variants as the attack categories (# Attacks in Table 1) contained in each dataset. We label the training variants of each dataset as *dn_NO(att),* where i) *dn* is the dataset name and ii) *att* is one of the attack categories (contained in the *dn* dataset), which is removed from the training set. For example (see Fig. 1), ISCX12_NO(DoS) indicates a training variant of the ISCX12 dataset where i) the training set contains normal data and data related to all attacks but DoS, i.e., Brute-Force, DDoS, and Infiltration attacks, and ii) the test set contains normal data and data related to all the attacks DoS, DDoS, Infiltration, and Brute-Force. Notably, the test set is unaltered: classifiers trained on different training sets or training variants of a specific dataset will be validated using the same test set. We iterate this process for all 11 datasets, obtaining a total of 58 training variants.

4.5. M4 - selection of classifiers and categories

Selecting a heterogeneous and relevant set of classifiers is of utmost importance for our study. As such, we select 6 SUP, 4 DEEP, 4 META-SUP, 11 UNS, and 22 META-UNS classifiers as described below and grouped in Table 2.

Table 2
Classifiers used in this study.

	Uses Meta-Learning *No*		*Yes*
Supervised	**DEEP** AutoGluon, FastAI, Py-Custom, TabNet	**SUP** kNN, LDA, Naïve Bayes, Logistic Regression, SVM	**META-SUP** *Bagging*: Random Forest *Boosting*: ADABoost, Gradient Boosting, XGBoost
Unsupervised	**UNS** COF, FastABOD, G-Means, K-Means, LOF, ODIN, One-Class SVM, HBOS, LDCOF, SDO, SOM, iForest		**META-UNS** *Bagging*: ensembles of each UNS *Boosting*: ensembles of each UNS

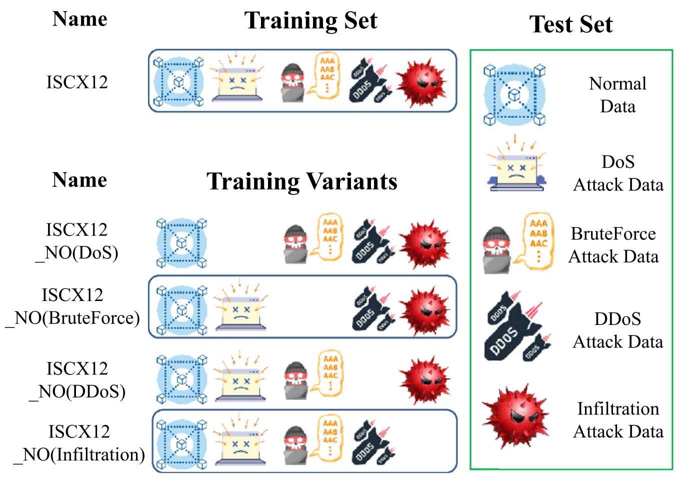


Fig. 1. Creation of 4 training variants from the ISCX12 training set. The same approach is applied to the 11 datasets. The normal data (no attacks) in the training set and in the training variants are the same. Each training variant has one attack category less than the training set. The test set is the same for the training set and the training variants.

4.5.1. Supervised classifiers (SUP)

We select 6 SUP classifiers that are widely used in the literature: decision trees (tree-based family), k-nearest neighbors (kNN, (Liao and Vemuri, 2002), neighbor family), support vector machines (SVMs, (Hearst et al., 1998), support vector family), logistic regression (Robles-Velasco et al., 2020), linear discriminant analysis (Srivastava et al., 2007), and naïve Bayes (Srivastava et al., 2007) (statistical family). The configuration parameters, selected through grid searches, are described below.

- Decision Tree: We adopt different depth limits in the range {5, 10, no limit} to build the tree.
- K-NN was use with values of $k \in \{1, 3, 5, 9, 19, 49, 99\}$ and Euclidean distance as a reference distance function. The usage of odd k values avoids ties in binary classification.
- SVM: We individually instantiate three different SVMs using linear, RBF, or polynomial (quadratic) kernels.
- Logistic regression, LDA and naïve Bayes: We use default parameters from Scikit-Learn (Scikit-Learn Library 2022).

4.5.2. Deep learners (DEEP)

Additionally, we select 4 deep learning approaches specific for the analysis of tabular data. We choose three approaches that are currently deemed the most promising in the state of the art, and for comparison, we also implement a by-the-book approach.

AutoGluon (AutoGluon Repository 2022) implements a deep neural network whose core structure is composed of densely connected layers of neurons (Erickson et al., 2020). TabNet (TabNet GitHub online) selects the most relevant features at each training epoch through a process that the authors call *sequential attention*. FastAI (Howard and Gugger, 2020) provides a built-in Tabular Learner that implements entity embedding of categorical features. Py-Custom is a custom model that we build using PyTorch and is available at (ABC 2022). We implement a deep neural network with two hidden linear layers of 200 and 100 neurons, and we apply one-hot encoding to process categorical features.

Autogluon, FastAI, and TabNet perform automatic parameter tuning. Py-Custom is trained with different weight decays of the Adam optimizer from the set {0.01, 0.02, 0.1, 0.2}. Notably, FastAI provides the *lr_finder* callback that trains the model over a few iterations and automatically finds the starting learning rate, which we also use in Py-Custom.

4.5.3. Supervised meta-learners (META-SUP)

We applied meta-learning to supervised classifiers as follows: i) bagging through random forests (Breiman, 2001) and ii) boosting using ADABoost (Rätsch et al., 2001) and extreme gradient boosting (XGBoost) (Chen and Guestrin, 2016). The parameters of classifiers, whose best configuration is selected through grid searches, are described below.

- Random Forest: 6 parameter combinations by using {10, 30, 100} trees and depth limit set to {10, *no limit*}.
- AdaBoostM2 with {10, 30, 100, 200} trees.
- Gradient boosting with {10, 30, 100} trees, learning rate of {0.5, 1.0}, and depth limit set to {10, *no limit*}, i.e., $3 * 2 * 2 = 12$ combinations of parameters.
- eXtreme gradient boosting (XGBoost) with different functions to estimate residual error, namely, {*default(logistic), squared-error, hinge,* Poisson}.

4.5.4. Unsupervised classifiers (UNS)

We select a pool of UNS classifiers that are as heterogeneous as possible and belong to the 7 main families (Goldstein and Uchida, 2016; T. Zoppi et al., 2021). We disregard heavy classifiers (e.g., ABOD (Kriegel and Zimek, n.d.) has cubic time complexity for training), which require huge resources to achieve adequate time performance. We select one classifier for each family: *K-Means* (clustering, (Hartigan and Wong, 1979)), *HBOS* (statistical, (Goldstein and Dengel, 2012)), *SOM* (neural-network, (Kohonen, 1997)), *FastABOD* (angle-based, (Kriegel and Zimek, n.d.)), *ODIN* (neighbor, (Hautamaki et al., 2004)), *LOF* (density, (Breunig et al., 2000)) and *Isolation Forests* (iForest (Liu et al., 2008), other family). Moreover, we select well-known classifiers such as *COF* (Tang et al., 2002), *LDCOF* (Amer and Goldstein, 2012), *one-class SVM, G-means* (Hamerly and Elkan, 2004), and *sparse density observers* (SDO, (Vázquez et al., 2018)) to complete the selection of UNS classifiers.

We use the following combinations of parameters for training those classifiers. Values k of neighbor-based classifiers, samples s and trees t of iForest, number *hist* of histograms in HBOS, and observers *obs* of SDO are chosen in the set {1, 2, 3, 5, 10, 20, 50, 100}.

Other classifiers have specific parameters as follows:

- One-class SVM may be created either with {linear, quadratic, cubic, radial basis function} kernels and $\nu \in \{0.01, 0.02, 0.05,$

0.1, 0.2}, where ν contributes to the definition of bounds on the number of support vectors to create.
- In addition to *obs*, SDO also needs the $q \in$ {0.05, 0.1, 0.2, 0.5} threshold that the classifier uses to derive the closest observers during the training phase.

4.5.5. Unsupervised meta-learners (META-UNS)

Additionally, we create bagging and boosting meta-learners of each UNS classifier: we create ensembles of {10, 20, 50} boosting and bagging instances of the same unsupervised classifier. For boosting, we vary the learning rate to update the sampling weights of train data points in the {0.5, 1, 2} range. This results in 22 META-UNS classifiers, i.e., a bagging and a boosting classifier for each of the 11 UNS classifiers.

4.6. M5 - Execute and compute scores

Experiments are executed on a Dell Precision 5820 Tower with an Intel I9–9920X, GPU NVIDIA Quadro RTX6000 with 24 GB VRAM, 192 GB RAM, and Ubuntu 18.04, and they required approximately 6 weeks of 24 h execution. We provide GPU support to exercise DEEP classifiers.

The *Scikit-Learn* (Scikit-Learn Library 2022) and *xgboost* (XGboost package 2022) *Python* packages contain all the code needed to exercise SUP and META-SUP classifiers, including mechanisms for grid searches. Instead, we exercised UNS and META-UNS classifiers through RELOAD (T. Zoppi et al., 2019), a Java open-source tool that includes many implementations of unsupervised classifiers and supports the creation of meta-learners. These frameworks allow the easy calculation of ACC, MCC, and REC metric values. Classifiers were trained using the combination of parameters that resulted in the highest MCC after grid searches, embracing all possible configurations from Section 4.5. Overall, we trained each of the 47 classifiers on each of the 11 training sets and the 58 training variants. Models obtained at the end of this process are used to evaluate detection performance.

Notably, ACC, MCC, and REC metrics do not quantify detection capabilities with respect to unknown attacks. Therefore, we define two new quantities TU and FU similar to TP and FN but related to the occurrence of unknown attacks.

TU (True Unknown): an unknown attack is detected.

FU (False Unknown): an unknown attack is not detected.

Then, we combine TU and FU into the Recall-Unknown (Rec-Unk) metric as follows:

$$Rec - Unk = \frac{TU}{TU + FU}$$

Rec-Unk shows the fraction of unknown attacks detected by the classifier out of all the unknown attacks. The higher the Rec-Unk is, the better coverage a classifier has in detecting unknown attacks. Computing Rec-Unk required writing a simple Python function that filters out normal data and known attacks from the test set. This way, it becomes easy to compute TU and FU.

The code to reproduce steps M1-M5 and the detailed result report are available in the repository (ABC 2022) and are anonymous for double-blind submission.

5. Results and discussion

Section 5.1 investigates which classifiers show better detection performance across all datasets and training variants and illustrates the difference between supervised (i.e., SUP, DEEP, META-SUP) and unsupervised (i.e., UNS, META-UNS) approaches. Section 5.2 discusses the ability to detect unknowns and their impact on detection performance. Then, we compare the performance of the best supervised classifier (XGBoost) against the best unsupervised classifier (boosting of FastABOD) in Section 5.3, and we discuss the conditions upon which unsupervised classifiers are preferable to supervised classifiers for intrusion detection (and vice versa).

5.1. On the detection performance of classifiers

Table 3 reports the ACC, MCC, Recall, and Rec-Unk achieved by all classifiers used in this study, averaged across datasets and training variants. The table shows the metric scores for each SUP, META-SUP, DEEP, UNS, and META-UNS classifier. We put in bold the lines corresponding to classifiers that resulted in the highest average MCC of each category. Underlined items point to classifiers (and metric scores), which we will extensively compare in Section 5.3, because they achieve either the highest MCC or the highest Rec-Unk.

Classifiers of the same category usually show slight variations. In particular, the ACC, MCC, and REC of the META-SUP classifiers (random forests, ADABoost, gradient boosting, and XGBoost) do not fluctuate much. The same applies to UNS classifiers, whose metric values are in the ranges ACC $\in$ [0.838; 0.876], MCC $\in$ [0.511; 0.591], and REC $\in$ [0.58; 0.65]. The largest variability of scores between classifiers of the same category occurs with SUP classifiers on the top left of Table 3. Scores achieved by Decision Trees and kNN are clearly better than those of LDA, Logistic Regression, Naïve Bayes, and SVM. A possible explanation is that LDA, naïve Bayes, and logistic regression heavily rely on a correct characterization of a statistical distribution that describes training data. A poor fitting of this distribution to the training data creates a model that outputs many misclassifications. Specifically, for supervised classifiers in the top half of Table 3, we observe that the META-SUP XGBoost and random forest have better ACC and MCC scores than the SUP and DEEP classifiers. In particular, DEEP classifiers are outperformed by META-SUP classifiers and by some SUPs. This confirms the known difficulties of deep learners in operating with tabular data (Shwartz-Ziv and Armon, 2022).

The ACC and MCC scores of the UNS classifiers are lower than those of the DEEP and META-SUP classifiers. In particular, the ACC and MCC of the UNS classifiers never exceed 0.88 and 0.60, respectively. The difference with respect to META-SUP is remarkable: the best UNS classifier generates on average more than 12 misclassifications out of 100 classifications (ACC of ODIN, the best UNS, is below 0.88), while with META-SUP, we have only half of those misclassifications (5.2 out of 100, since XGBoost has ACC=0.948). These results are not surprising: supervised classifiers are known to be more accurate than unsupervised classifiers when detecting known attacks. However, Table 3 also shows that META-UNS improves all metric scores with respect to UNS classifiers. Particularly, the Boosting ensemble of FastABOD (on the right of Table 3, bold, underlined) generates ACC scores that outperform DEEP classifiers; this halves the accuracy gap between META-SUP and UNS classifiers. This is important because UNS classifiers are traditionally considered to perform much more poorly than SUP, DEEP, and META-SUP classifiers and much less used in practice.

5.2. Detection of unknown attacks

We evaluate the ability to detect unknown attacks of DEEP, META-SUP, and META-UNS. We avoid considering SUP and UNS classifiers because their meta-learning counterparts, META-SUP and META-UNS, respectively, are definitely better.

The capability of detecting unknown attacks is quantified by the Rec-Unk metric in Table 3, which represents the percentage of unknown attacks being detected by the classifier, averaged across all datasets. DEEP classifiers do not exceed 36% of Rec-Unk (on average, at least 3 out of 5 unknown attacks remain undetected). Instead, META-UNS classifiers are intrinsically more likely to de-

Table 3
Average ACC, MCC, Recall, Rec-Unk obtained by classifiers when trained on the 11 training sets and the 58 training variants.

Type	Classifier		no Meta-Learning					Bagging				Boosting			
			ACC	MCC	Recall	Rec-Unk		ACC	MCC	Recall	Rec-Unk	ACC	MCC	Recall	Rec-Unk
Supervised	**Decision Tree**	**SUP**	**0.935**	**0.745**	**0.73**	**0.40**									
	kNN		0.936	0.742	0.76	0.28									
	LDA		0.862	0.473	0.46	0.26									
	Logistic Regression		0.808	0.296	0.30	0.16									
	Naïve Bayes		0.803	0.296	0.31	0.32									
	SVM		0.820	0.539	0.63	0.35									
	Random Forest						**META-SUP**	**0.945**	**0.781**	**0.74**	**0.35**				
	ADABoost											0.921	0.700	0.65	0.30
	Gradient Boosting											0.935	0.743	0.71	0.38
	<u>**XGBoost**</u>											<u>**0.948**</u>	<u>**0.796**</u>	<u>**0.82**</u>	<u>**0.38**</u>
	AutoGluon	**DEEP**	**0.901**	**0.691**	**0.70**	**0.36**									
	FastAI		0.883	0.587	0.56	0.27									
	Py-Custom		0.895	0.614	0.62	0.30									
	TabNet		0.879	0.451	0.45	0.26									
Unsupervised	COF	**UNS**	0.874	0.589	0.63	0.57	**META-UNS**	0.889	0.638	0.70	0.58	0.890	0.632	0.73	0.59
	<u>**FastABOD**</u>		0.853	0.544	0.65	0.64		0.870	0.588	0.67	0.60	<u>**0.908**</u>	<u>**0.674**</u>	<u>**0.76**</u>	<u>**0.65**</u>
	G-Means		0.859	0.532	0.61	0.53		0.884	0.596	0.64	0.57	0.907	0.652	0.68	0.56
	K-Means		0.871	0.551	0.60	0.54		0.887	0.601	0.66	0.55	0.904	0.643	0.69	0.57
	LOF		0.838	0.514	0.58	0.56		0.844	0.500	0.54	0.51	0.847	0.530	0.66	0.52
	ODIN		**0.876**	**0.591**	**0.65**	**0.58**		0.898	0.625	0.68	0.61	0.905	0.657	0.72	0.60
	One-Class SVM		0.871	0.555	0.59	0.54		0.874	0.574	0.62	0.57	0.887	0.616	0.67	0.57
	HBOS		0.875	0.565	0.63	0.62		0.855	0.537	0.63	0.55	0.899	0.637	0.68	0.56
	LDCOF		0.854	0.511	0.58	0.54		0.872	0.580	0.64	0.55	0.901	0.643	0.68	0.54
	SDO		0.867	0.549	0.59	0.50		**0.890**	**0.643**	**0.64**	**0.56**	0.895	0.645	0.71	0.58
	SOM		0.853	0.512	0.58	0.55		0.885	0.624	0.64	0.55	0.889	0.609	0.64	0.54
	iForest		0.876	0.587	0.63	0.55		0.881	0.594	0.64	0.53	0.889	0.640	0.62	0.58

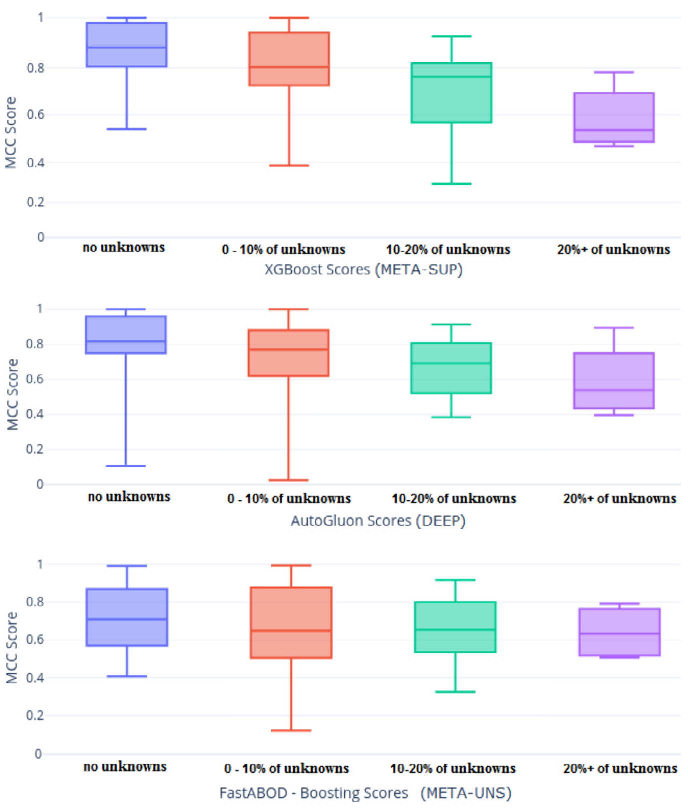


Fig. 2. Box-plots showing the MCC scores of META-SUP XGBoost, DEEP AutoGluon and META-UNS FastABOD - Boosting classifiers when i) all attacks are known, ii) at most 10% are unknown attacks, iii) between 10 and 20% are unknown attacks, and iv) more than 20% are unknown attacks.

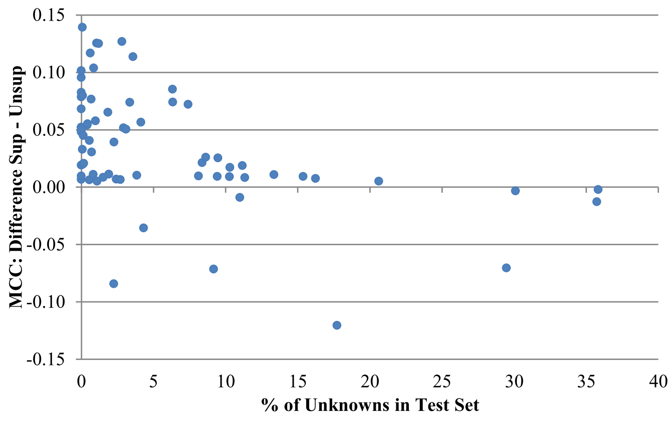
tect unknown attacks due to the way they learn the model during training.

Furthermore, Fig. 2 shows box plots for the META-SUP XGBoost, DEEP AutoGluon, and META-UNS FastABOD - Boosting classifiers with varying numbers of unknown attacks. These three classifiers have the best ACC and MCC scores in their groups. The blue box on the left of the figure draws MCC scores when no unknowns occur: XGBoost and AutoGluon scores are higher than those of FastABOD. However, increasing the percentage of unknowns in the test set makes the MCC of supervised and deep classifiers drop by a noticeable amount, whereas the MCC of FastABOD suffers only a minor degradation (see red, green, and purple boxes).

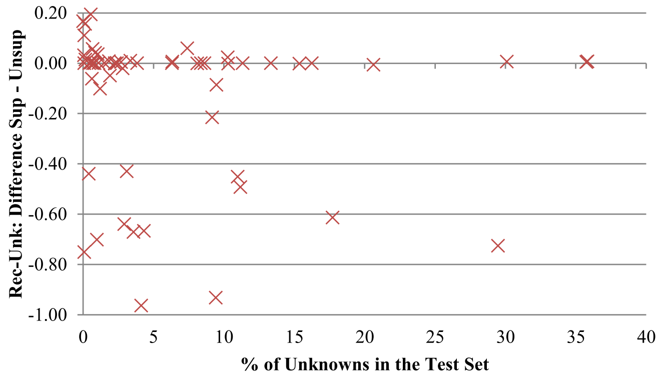
We elaborate on this with the aid of Fig. 3, which we built according to the following procedure. We train all classifiers using a training set, evaluate them on the test set and select the supervised (SUP, META-Sup, or DEEP) and unsupervised (UNSUP or META-UNS) classifiers with the highest MCC. These two classifiers are considered the best supervised and unsupervised approaches for a given training set. We repeat this process by training all classifiers using training variants, evaluating them on the test set, and selecting the supervised and unsupervised classifiers that have the highest MCC. Last, we repeat the procedure but measure Rec-Unk instead of MCC. In total, this produces 69 points in the figure, obtained from the 11 training sets and the 58 training variants.

For each training set and the training variants, we compute the difference in MCC (Fig. 3a) and Rec-Unk (Fig. 3b) between the two best classifiers previously selected. These differences are ultimately depicted in a scatterplot against the percentage of unknowns in the test set. As previously discussed, there are no unknowns when training on the training sets: the corresponding results are on the $x = 0$ axes. In the other cases, the ratio of unknowns differs depending on the training variant, from 0.5% to almost 40%. Both scatterplots contain 69 items, i.e., one item for each training (on the 11 datasets and 58 variants). Items above the x-axis point to datasets or training variants where a supervised classifier is better than an unsupervised classifier.

Clearly, different classifiers may be selected when varying the training sets and training variants. Particularly, the META-SUP XGBoost is selected in 36 out of 69 cases, the META-SUP Random Forests in 18 out of 69, and the DEEP AutoGluon in 10 out of 69. For unsupervised classifiers, FastABOD outperforms others in 26 out of 69 cases, SDO in 12 out of 69, HBOS in 7 out of 69, and ODIN in 6 out of 69.



a) MCC differences between supervised and unsuperised classifiers



b) RecUnk differences between supervised and unsuperised classifiers

Fig. 3. Differences in MCC (Fig. 3a) and Rec-Unk (Fig. 3b) of the best supervised classifier versus the best unsupervised classifier when trained on the 11 training sets and the 58 training variants. Differences are plotted against the% of unknowns in the test set.

Fig. 3a highlights that SUP, DEEP and META-SUP classifiers usually result in higher MCC scores, with fewer misclassifications – both FPs and FNs – than UNS and META-UNS classifiers. This trend becomes progressively less evident as the number of unknowns in the test set increases: on the right of the plot, the difference in MCC scores becomes almost negligible or even negative, meaning that there is a turning point at which unsupervised classifiers become better overall.

Fig. 3b shows the superior capabilities of UNS and META-UNS in detecting unknown attacks. The difference in Rec-Unk is almost always negative: unsupervised classifiers are better than supervised classifiers in identifying unknown attacks.

5.3. In-Depth comparison of boosting meta-learners

Here, we directly compare the two classifiers shown in Table 3 that have the highest ACC and MCC scores for supervised and unsupervised intrusion detection. These are the META-SUP XGBoost and the META-UNS FastABOD - Boosting, respectively.

XGBoost (Chen and Guestrin, 2016) is an optimized distributed classifier that builds ensembles of decision trees that cooperate to perform classification. Notably, XGBoost was proven to outperform many DEEP classifiers in a recent study (Shwartz-Ziv and Armon, 2022). Instead, FastABOD is an unsupervised classifier that decides on anomalies by calculating the variance of the amplitude of the angles where a given data point is the vertex, and the other two external items are any couple of neighboring data points. FastABOD was primarily meant to be used as a regular UNS classifier, but it can be adapted to build META-UNS boosting meta-learners that improve classification performance.

As shown in Table 3, XGBoost outperforms FastABOD - Boosting when looking at average ACC, MCC, and Recall scores. Instead, FastABOD - Boosting almost doubles the Rec-Unk score of XGBoost. Fig. 4 elaborates more on this comparison. The approach is the same we followed for Fig. 3, but we compute the difference in MCC and Rec-Unk between XGBoost and FastABOD - Boosting. Fig. 4a shows how FastABOD - Boosting generates fewer misclassifications than XGBoost when unknowns in the test set exceed 10%. Items in the figure are either close to the $y = 0$ value or below. This is especially evident when the unknowns exceed 10% of attacks: points on the right of the dashed vertical line in Fig. 4b mostly fall below the x-axis.

To summarize, our results show that when the percentage of unknowns in the test set exceeds 10%, the META-UNS FastABOD - Boosting is likely to be more accurate than the META-SUP XGBoost. More generally, META-UNS classifiers have better detection capabilities of intrusions when the percentage of unknown attacks is above a given threshold. With few unknown attacks, the results are more unpredictable, but there is still evidence of the superior ability to detect unknowns of unsupervised classifiers: the differences in Fig. 4b are mostly negative, highlighting how Rec-Unk scores achieved by FastABOD - Boosting are higher than those of XGBoost.

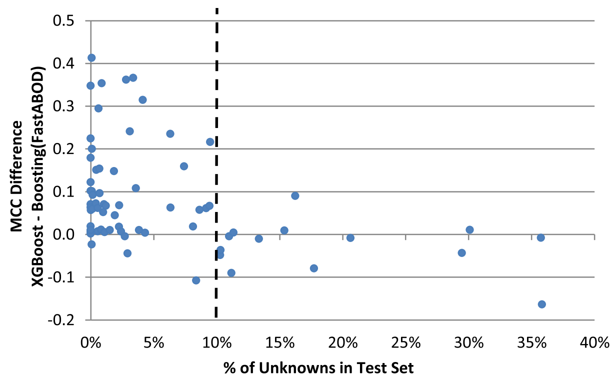
6. Threats to validity

Here, we report possible limitations to the validity and applicability of our study. These are not to be intended as showstoppers when considering the conclusions of this paper. Instead, they should be interpreted as boundaries that may affect the validity of this study.

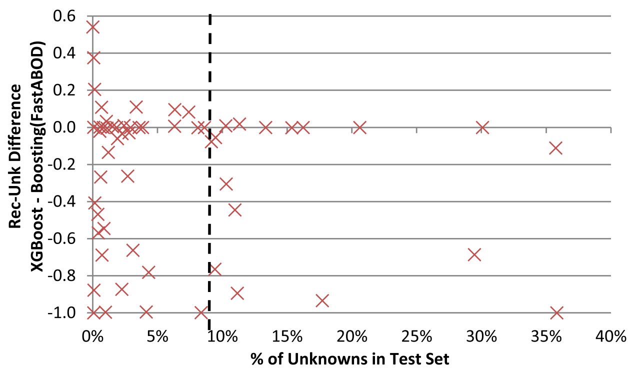
6.1. Internal validity

Internal validity is concerned with factors that may have influenced the results, but they have not been thoroughly considered in the study.

- First, public datasets i) are often collected from heterogeneous systems, ii) may have been documented poorly, limiting the understandability of data, and iii) are not under our control (Catillo et al., 2021); therefore, actions such as considering more features, rebalancing classes or improving the quality of data to improve detection scores are out of consideration. We believe that the usage of many datasets, the creation of variants and the employment of a wide variety of different classifiers allowed us to reduce the adverse effect of this threat on this study as much as possible.
- Second, almost all 47 classifiers have hyperparameters whose optimal values have to be derived for each dataset as a result of sensitivity analyses. When applying DEEP, SUP, META-SUP, UNS, and META-UNS classifiers to different datasets, sensitivity analyses may not always find the optimal combinations of parameter values: as a result, metric values could be slightly inferior with respect to using a classifier with optimal parameter values. We tried our best to minimize this last aspect by exercising sensitivity analyses for the main parameters of all 47 classifiers considered in this study.
- Third, each classifier may encounter a wide variety of problems when learning a model for each dataset during training (e.g., under/overfitting, poor quality of features, feature selection to leave out noisy features). It was not possible to examine each single case to detect and eventually solve these problems, but we believe that these events are mostly situational and do not



a) MCC differences between XGBoost and FastABOD - Boosting



b) Rec-Unk differences between XGBoost and FastABOD - Boosting

Fig. 4. Differences in MCC (Fig. 4a) and Rec-Unk (Fig. 4b) of XGBoost against the Boosting ensemble of FastABOD (FastABOD - Boosting) when trained on the 11 training sets and the 58 training variants. Differences are plotted against the% of unknowns in the test set.

have a noticeable impact when looking at the detection performance of the same classifier over a span of many datasets.

6.2. External validity

External validity is concerned with to what extent the results of the study can be generalized.

We conclude that META-UNS classifiers have better detection capabilities than others when the likelihood of unknown attacks exceeds 10%. This conclusion was obtained at the end of an experimental study embracing many public datasets and many ML algorithms for binary classification. It was not feasible to exercise all possible classifiers using all available attack datasets; however, we believe that the subset of classifiers includes algorithms belonging to all the most widespread families and, as such, does not have a detrimental impact on the generalization capabilities of this study.

6.3. Reproducibility of the study

Reproducibility is concerned with to what extent the study is dependent on the researcher(s), i.e., if other researchers conducted the exact same study, the result should be almost the same.

The usage of public data and public tools to run algorithms was a prerequisite of our analysis to allow reproducibility and to rely on proven-in-use data. We publicly shared scripts, methodologies and all metric scores, allowing any researcher or practitioner to repeat the experiments.

7. Conclusions and future works

Supervised classifiers and, to a lesser extent, deep learners are usually accurate in detecting known attacks, but they cannot effectively detect unknown attacks (either brand new attacks, variants of existing exploits, or threats against which the intrusion detector is unprepared). Conversely, unsupervised classifiers are usually less accurate than supervised classifiers, because their learning process does not rely on labelled train data. However, the accuracy of unsupervised classifiers does not suffer major degradation in case of unknown attacks.

This paper conducted an experimental analysis to compare supervised and unsupervised classifiers, with and without the adoption of meta-learning, to quantitatively analyze their ability in detecting known and unknown attacks. Results showed that unsupervised meta-learners are the best solution to detect unknown attacks, and can detect known attacks similarly to several supervised classifiers. Summarizing, unsupervised meta-learning is a promising approach to implement IDSs: it offers satisfactory detection accuracy in case of both known and unknown attacks.

However, the detection of unknown attacks is still an open challenge, and more research is needed. Among the many alternatives, our current and future works are focusing on crafting intrusion detectors that can detect known and unknown attacks even if minimal to no information on attacks is provided in the training data. This happens frequently in many applications, when creating an attack dataset to train the intrusion detector is not possible or exceedingly expensive. Consequently, we foresee the application of statistical methods (Moller et al., 2021) and generative adversarial networks (Ashrapov, 2020) to generate out-of-distribution data, which we will use to enrich the training set and train accurate intrusion detectors even if only normal data is provided. We will compare the detection performance of this approach with intrusion detectors crafted using zero-shot learning, experimenting with approaches similar to (Zhang et al., 2020), which show promising results in setups where the availability of data for training intrusion detectors is scarce.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Tommaso Zoppi: Conceptualization, Data curation, Methodology, Writing – original draft, Software, Validation. **Andrea Ceccarelli:** Methodology, Investigation, Writing – review & editing, Validation. **Tommaso Puccetti:** Investigation, Software, Validation. **Andrea Bondavalli:** Supervision, Writing – review & editing.

Data availability

The paper uses public tools and datasets. Additional files are shared in the paper at the ZIP file linked as reference [79] in the paper.

Acknowledgments

This work has been partially supported by the Regione Toscana POR FESR 2014–2020 SPaCe and by the NextGenerationEU program, Italian DM737 – CUP B15F21005410003.

References

Akyildiz, I.F., Kak, A., 2019. The Internet of Space Things/CubeSats: a ubiquitous cyber-physical system for the connected world. Comput. Networks Chem. Lab., Symp. 150, 134–149.

Dey, N., Ashour, A.S., Shi, F., Fong, S.J., Tavares, J.M.R., 2018. Medical cyber-physical systems: a survey. J. Med. Syst. 42 (4), 1–13.

ABC, A zero-day guide for 2020: recent attacks and advanced preventive techniques (online), https://blog.malwarebytes.com/exploits-and-vulnerabilities/2020/06/a-zero-day-guide-for-2020/

Goldstein, M., Uchida, S., 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PLoS One 11 (4), e0152173.

Breiman, L. "Random forests." Mach Learn 45.1 (2001): 5–32.

Rätsch, G, Onoda, T, Müller, K-R., 2001. Soft margins for AdaBoost. Mach Learn 42 (3), 287–320.

Hearst, MA., et al., 1998. Support vector machines. IEEE Intell. Syst. 13 (4), 18–28.

Kriegel H.-.P., Zimek A. “Angle-based outlier detection in high-dimensional data”. Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge discovery data mining; ‘08. p. 444–452.

Zoppi, T., Ceccarelli, A., Bondavalli, A., 2019a. Evaluation of anomaly detection algorithms made easy with RELOAD. In: Proceedings of the 30th Int. Symposium on Software Reliability Engineering (ISSRE). IEEE, pp. 446–455.

Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., 2019. Survey of intrusion detection systems: techniques, datasets, and challenges. Cyber Secur 2 (1), 20.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A., 2019. A survey of network-based intrusion detection data sets. Comput. Secur..

Zoppi, T., Gharib, M., Atif, M., Bondavalli, A., 2021a. Meta-Learning to improve unsupervised intrusion detection in cyber-physical systems. ACM Trans. Cyber-Physical Syst. 5 (4), 1–27.

Zoppi, T., Ceccarelli, A., Capecchi, T., Bondavalli, A., 2021b. Unsupervised anomaly detectors to detect intrusions in the current threat landscape. ACM/IMS Trans. Data Sci. 2 (2), 1–26.

Elsayed, M.S., Le-Khac, N.A., Jurcut, A.D., 2020. InSDN: a novel SDN intrusion dataset. IEEE Access 8, 165263–165284.

Kang, H, Ahn, DH, Lee, GM, Yoo, JDo, Park, KHo, Kim, HK, 2019. IoT network intrusion dataset. IEEE Dataport doi:10.21227/q70p-q449.

Liao, Y, Vemuri, V.R, 2002. Use of k-nearest neighbor classifier for intrusion detection. Comput. Secur. 21 (5), 439–448.

Srivastava, S., Gupta, M.R., Frigyik, B.A., 2007. Bayesian quadratic discriminant analysis. J. Mach. Learn Res. 8 (Jun), 1277–1305.

Chen, T., Guestrin, C., 2016. Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM sigkdd international conference on knowledge discovery and data mining, pp. 785–794.

Chicco, D, Jurman, G, 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. Biomed. Chromatogr. 21 (1), 6.

Boughorbel, S, Jarray, F, El-Anbari, M, 2017. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. PLoS One 12 (6), e0177678.

Campos, G.O., Zimek, A., Sander, J., Campello, R.J., Micenko-va, B., Schubert, E., Assent, I., Houle, M.E., 2016. On the evaluation of outlier detection: measures, datasets, and an empirical study. In: Proceedings of the Lernen, Wissen, Daten, Analysen 2016. CEUR workshop proceedings.

Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: a survey. In: Proceedings of the ACM computing surveys (CSUR), 41, p. 15.

Zoppi, T., Ceccarelli, A., Bondavalli, A., 2021c. Unsupervised classifiers to detect zero-day attacks: strategy and application. IEEE Access 9, 90603–90615.

do Nascimento, P.P., Pereira, P., Mialaret, J.M., Ferreira, I., Maciel, P., 2021. A methodology for selecting hardware performance counters for supporting non-intrusive diagnostic of flood DDoS attacks on web servers. Comput. Secur. 110, 102434.

Rodríguez, P., Bautista, M.A., Gonzalez, J., Escalera, S., 2018. Beyond one-hot encoding: lower dimensional target embedding. Vis. Comput. 75, 21–31.

Sathya, R., Abraham, A., 2013. Comparison of supervised and unsupervised learning algorithms for pattern classification. Int. J. Adv. Res. Artif. Intell. 2 (2), 34–38.

Lee, K., Booth, D., Alam, P., 2005. A comparison of supervised and unsupervised neural networks in predicting bankruptcy of Korean firms. Expert Syst. Appl. 29 (1), 1–16.

Medico, R., Lambrecht, N., Pues, H., Ginste, D.V., Deschrijver, D., Dhaene, T., Spina, D., 2018. Machine learning based error detection in transient susceptibility tests. IEEE Trans. Electromagn. Compat. 61 (2), 352–360.

Nishida, K., Sadamitsu, K., Higashinaka, R., Matsuo, Y., 2017. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence.

Guo, C., & Berkhahn, F. (2016). Entity embeddings of categorical variables. arXiv preprint arXiv:1604.06737.

Zhang, C., Jia, D., Wang, L., Wang, W., Liu, F., Yang, A., 2022. Comparative research on network intrusion detection methods based on machine learning. Comput. Secur., 102861.

Zhu, Y., Brettin, T., Xia, F., Partin, A., Shukla, M., Yoo, H., Stevens, R.L., 2021. Converting tabular data into images for deep learning with convolutional neural networks. Sci. Rep. 11 (1), 1–11.

Shwartz-Ziv, R., Armon, A., 2022. Tabular data: deep learning is not all you need. Aktuel. Aspekte Kernfusionsforsch., Informationstag. 81, 84–90.

Howard, J, Gugger, S, 2020. Fastai: a layered API for deep learning. Information 11 (2), 108.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2020). Autogluon-tabular: robust and accurate automl for structured data. arXiv preprint arXiv:2003.06505.

Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y., Chen, Z., 2021. SySeVR: a framework for using deep learning to detect software vulnerabilities. IEEE Trans. Dependable Secure Comput. early access article.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

Goldstein, M, Dengel, A, 2012. Histogram-based outlier score (HBOS): a fast unsupervised anomaly detection algorithm. In: Proceedings of the KI-2012: Poster and Demo Track, pp. 59–63.

Hautamaki, V., Karkkainen, I., Franti, P., 2004. Outlier detection using k-nearest neighbour graph. In: Proceedings of the Pattern Recognition. ICPR 2004. Proceedings of the 17th International Conference on, 3. IEEE, pp. 430–433.

Kohonen, T., 1997. Exploration of very large databases by self-organizing maps. In: Proceedings of International Conference on Neural Networks (ICNN’97), 1. IEEE, pp. PL1–PL6.

Liu, F.T., Ting, K.M., Zhou, Z.H., 2008. Isolation forest. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. IEEE, pp. 413–422.

Hartigan, J.A., Wong, M.A., 1979. Algorithm AS 136: a k-means clustering algorithm. J. R. Stat. Soc. Ser. C Appl. Stat. 28 (1), 100–108.

Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J., 2000. LOF: identifying density-based local outliers. In: Proceedings of the ACM sigmod record, 29. ACM, pp. 93–104.

Zhao, Z, Cerf, S, Birke, R, Robu, B, Bouchenak, S, Mokhtar, SB, Chen, LY., 2019. Robust anomaly detection on unreliable data. In: Proceedings of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, pp. 630–637.

van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J., 2015. Having a blast: meta-learning and heterogeneous ensembles for data streams. In: Proceedings of the 2015 IEEE international conference on data mining. IEEE, pp. 1003–1008.

Robles-Velasco, A., Cortés, P., Muñuzuri, J., Onieva, L., 2020. Prediction of pipe failures in water supply networks using logistic regression and support vector classification. Reliab. Eng. Syst. Saf. 196, 106754.

Vázquez, FI, Zseby, T, Zimek, A, 2018. Outlier detection based on low density models. In: Proceedings of the 2018 IEEE Int. Conference on Data Mining Workshops (ICDMW). IEEE.

Hamerly, G., & Elkan, C. (2004). Learning the k in k-means. In Advances in neural information processing systems (pp. 281–288).

Amer, M, Goldstein, M, 2012. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In: Proceedings of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012).

Tang, J, Chen, Z, Fu, AW-C, Cheung, DW, 2002. Enhancing effctiveness of outlier detections for low density patterns. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp. 535–548.

Scikit-Learn Library (online), Scikit-Learn Library, https://scikit-learn.org/stable/user_guide.html (last accessed: 4th August 2022)

AutoGluon Repository (online), https://auto.gluon.ai/stable/index.html (last accessed: 4th August 2022)

TabNet GitHub (online), https://github.com/dreamquark-ai/tabnet (last accessed: 4th August 2022)

Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A., 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput. Secur. 31 (3), 357–374.

Tavallaee, M, Bagheri, E, Lu, W, Ghorbani, AA, 2009. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on. IEEE, pp. 1–6.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D., Hotho, A., 2017. Flow-based benchmark data sets for intrusion detection. In: Proceedings of the 16th European Conference on Cyber Warfare and Security. ACPI, pp. 361–369.

Moustafa, N, Slay, J, 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Proceedings of the Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, pp. 1–6.

Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the ICISSP, pp. 108–116.

Haider, W., Hu, J., Slay, J., Turnbull, B.P., Xie, Y., 2017. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. J. Netw. Comput. Appl. 87, 185–192.

Lashkari, A.H., Kadir, A.F.A., Taheri, L., Ghorbani, A.A., 2018. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In: Proceedings of the 2018 International Carnahan Conference on Security Technology (ICCST). IEEE, pp. 1–7.

Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., Theron, R., 2018. UGR ‘16: a new dataset for the evaluation of cyclostationarity-based network IDSs. Comput. Secur. 73, 411–424.

Resende, P.A.A., Drummond, A.C., 2018. A survey of random forest based methods for intrusion detection systems. ACM Computing Surveys (CSUR) 51 (3), 1–36.

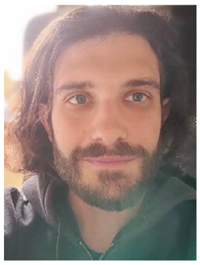
Chkirbene, Z., Eltanbouly, S., Bashendy, M., AlNaimi, N., Erbad, A., 2020. Hybrid machine learning for network anomaly intrusion detection. In: Proceedings of the 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), pp. 163–170.

Taher, K.A., Mohammed Yasin Jisan, B., Rahman, M.M., 2019. Network intrusion detection using supervised machine learning technique with feature selection. In: Proceedings of the 2019 Int. Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), pp. 643–646.

Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S., 2019. Deep learning approach for intelligent intrusion detection system. IEEE Access 7, 41525–41550.
Catillo, M, et al., 2022. Transferability of machine learning models learned from public intrusion detection datasets: the CICIDS2017 case study. Software Quality J. 1–27.
Ardagna, C., Corbiaux, S., Sfakianakis, A., Douliger, C., ENISA Threat Landscape 2021 (online), https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends (last accessed: 4th August 2022)
Connell, B., “2022 SonicWall Threat Report” (online), https://www.sonicwall.com/2022-cyber-threat-report/(last accessed: 4th August 2022)
Cotroneo, D, Natella, R, Rosiello, S, 2017. A fault correlation approach to detect performance anomalies in Virtual Network Function chains. In: Proceedings of the Software Reliability Engineering (ISSRE), 2017 IEEE 28th Int. Symposium on. IEEE, pp. 90–100.
Cruz, T., Barrigas, J., Proença, J., Graziano, A., Panzieri, S., Lev, L., Simões, P., 2015. Improving network security monitoring for industrial control systems. In: Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, pp. 878–881.
Zoppi, T., Ceccarelli, A., Bondavalli, A., 2019b. MADneSs: a multi-layer anomaly detection framework for complex dynamic systems. IEEE Trans. Dependable Secure Comput. 18 (2), 796–809.
Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R., 2009. Metalearning: Applications to Data Mining. Springer, Berlin.
He, P., Zhu, J., He, S., Li, J., Lyu, M.R., 2017. Towards automated log parsing for large-scale log data analysis. IEEE Trans. Dependable Secure Comput. 15 (6), 931–944.
Chou, D., Jiang, M., 2021. A survey on data-driven network intrusion detection. ACM Computing Surveys (CSUR) 54 (9), 1–36.
Buczak, AL., Guven, E, 2015. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surveys Tutorials 18 (2), 1153–1176.
Al, S., Dener, M., 2021. STL-HDL: a new hybrid network intrusion detection system for imbalanced dataset on big data environment. Comput. Secur. 110, 102435.
XGboost package (online) https://xgboost.readthedocs.io/en/stable/python/python_intro.html (last accessed: 4th August 2022)
ABC, Additional files for Submission (online ZIP file) https://github.com/tommyippoz/Miscellaneous-Files/blob/master/COSE22_Zoppi_SupportingMaterial.zip (last accessed: 4th August 2022)
Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A., 2021. Revisiting deep learning models for tabular data. Adv. Neural Inf. Process. Syst. 34.
Casas, P, Mazel, J, Owezarski, P, 2012. Unsupervised network intrusion detection systems: detecting the unknown without knowledge. Comput. Commun. 35 (7), 772–783.
Catillo, M., Pecchia, A., Rak, M., Villano, U., 2021. Demystifying the role of public intrusion datasets: a replication study of DoS network traffic data. Comput. Secur. 108, 102341.
Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C., 2004. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. 1 (1), 11–33.
Zhang, Z., Liu, Q., Qiu, S., Zhou, S., Zhang, C., 2020. Unknown attack detection based on zero-shot learning. IEEE Access 8, 193981–193991.
Moller, F., Botache, D., Huseljic, D., Heidecker, F., Bieshaar, M., Sick, B., 2021. Out-of-distribution detection and generation using soft brownian offset sampling and autoencoders. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 46–55.
Ashrapov, I. (2020). Tabular GANs for uneven distribution. arXiv preprint arXiv:2010.00638.

Tommaso Zoppi received his Ph.D. in Computer Science from the University of Firenze (Italy) in 2018. He is currently a Research associate at the same university, mainly working on error detection, intrusion detection and failure prediction through anomaly detection. Often, his research crosses the domains of safety, security, and system engineering with participation to projects about safety critical architectures for railway systems, applying different Verification&Validation activities as required by applicable standards. He serves as TPC member in various international conferences and as a reviewer of top-rated journals. His-scientific activities originated more than 30 papers to date.

Andrea Ceccarelli received the PhD in Informatics and Automation Engineering from the University of Florence, Florence, Italy, in 2012. He is currently a Research Associate of Computer Science at the same University. His-primary research interests are in the design, monitoring and experimental evaluation of dependable and secure systems, and systems-of-systems. His-scientific activities originated more than 100 papers which appeared in international conferences, workshops, and journals.


Tommaso Puccetti received the master’s degree in Computer Science (curriculum: “Resilient and Secure Cyber Physical Systems”) from the University Florence, Italy, in 2021. Currently, he is the holder of a research grant issued by the University of Florence on the topic of "Design and evaluation of secure systems and applications for the railway domain”. He is mainly active in the study of safety critical systems and their verification and validation processes guided by the standards. He is also interested in the aspects related to the security and safety of ML components that operate within these systems and in the domain of the related defensive mechanisms, such as intrusion detectors.

Andrea Bondavalli is a Full Professor of Computer Science at the University of Florence. His-research interest is focused on Dependability and Resilience of critical systems and infrastructures. In particular, he has been working on designing resiliency, safety, security, and on evaluating attributes such as reliability, availability and performability. He led various national and European projects and has been chairing the PC in several International Conferences in the field, Andrea Bondavalli is a member of the IEEE, and of the IFIP W.G. 10.4 Working Group on "Dependable Computing and Fault-Tolerance.