Meta-Learning to Improve Unsupervised Intrusion Detection in Cyber-Physical Systems

TOMMASO ZOPPI, MOHAMAD GHARIB, MUHAMMAD ATIF, and
ANDREA BONDAVALLI, Dept. of Mathematics and Informatics, University of Florence, Florence - Italy

Artificial Intelligence (AI)-based classifiers rely on **Machine Learning (ML)** algorithms to provide functionalities that system architects are often willing to integrate into critical **Cyber-Physical Systems (CPSs)**. However, such algorithms may misclassify observations, with potential detrimental effects on the system itself or on the health of people and of the environment. In addition, CPSs may be subject to threats that were not previously known, motivating the need for building **Intrusion Detectors (IDs)** that can effectively deal with zero-day attacks. Different studies were directed to compare misclassifications of various algorithms to identify the most suitable one for a given system. Unfortunately, even the most suitable algorithm may still show an unsatisfactory number of misclassifications when system requirements are strict. A possible solution may rely on the adoption of meta-learners, which build ensembles of base-learners to reduce misclassifications and that are widely used for supervised learning. Meta-learners have the potential to reduce misclassifications with respect to non-meta learners: however, misleading base-learners may let the meta-learner leaning towards misclassifications and therefore their behavior needs to be carefully assessed through empirical evaluation. To such extent, in this paper we investigate, expand, empirically evaluate, and discuss meta-learning approaches that rely on ensembles of unsupervised algorithms to detect (zero-day) intrusions in CPSs. Our experimental comparison is conducted by means of public datasets belonging to network intrusion detection and biometric authentication systems, which are common IDSs for CPSs. Overall, we selected 21 datasets, 15 unsupervised algorithms and 9 different meta-learning approaches. Results allow discussing the applicability and suitability of meta-learning for unsupervised anomaly detection, comparing metric scores achieved by base algorithms and meta-learners. Analyses and discussion end up showing how the adoption of meta-learners significantly reduces misclassifications when detecting (zero-day) intrusions in CPSs.

CCS Concepts: • **Security and privacy → Intrusion/anomaly detection and malware mitigation**; • **Computer systems organization → Dependable and fault-tolerant systems and networks**; *Embedded and cyber-physical systems*;

Additional Key Words and Phrases: Critical systems, intrusion detection, machine learning, meta-learning, security, reliability

This work has been partially supported by the REGIONE TOSCANA POR FESR 2014-2020 SISTER and by the H2020 programme under the Marie Sklodowska-Curie grant agreement 823788 (ADVANCE) projects. Portions of the research in this paper use the CASIA-FingerprintV5 collected by the Chinese Academy of Sciences' Institute of Automation (CASIA).
Authors' Address: T. Zoppi, M. Gharib, M. Atif, and A. Bondavalli, Dept. of Mathematics and Informatics, University of Florence, Viale Morgagni 65, 50142 - Florence - Italy; emails: {tommaso.zoppi, mohamad.gharib, muhammad.atif, bondavalli}@unifi.it.

2378-962X/2021/09-ART42 $15.00
https://doi.org/10.1145/3467470

ACM Reference format:
Tommaso Zoppi, Mohamad Gharib, Muhammad Atif, and Andrea Bondavalli. 2021. Meta-Learning to Improve Unsupervised Intrusion Detection in Cyber-Physical Systems. *ACM Trans. Cyber-Phys. Syst.* 5, 4, Article 42 (September 2021), 27 pages.
https://doi.org/10.1145/3467470

1 INTRODUCTION

Nowadays the paradigm of **Cyber-Physical Systems (CPSs)** guides the definition and design of hardware-software systems whose functionalities are partially controlled or monitored by computer-based sub-systems. According to [52], "*in cyber-physical systems, physical and software components are deeply intertwined, able to operate on different spatial and temporal scales, exhibit multiple and distinct behavioural modalities, and interact with each other in ways that change with context*". In particular, CPSs might deliver critical functionalities, whose malfunction may lead to fatalities, severe injuries, or major damages to the environment: As a result, they must be conceptualized, designed, and implemented to ensure that appropriate safety and/or security requirements are met [13].

Intrusion Detection in CPSs. Amongst those requirements, CPSs may be subject to (cyber)attacks. The U.S.A. Committee on National Security Systems Glossary [12] defines *cybersecurity* as "*prevention of damage to, protection of, and restoration of computers, electronic communications systems, electronic communications services, wire communication, and electronic communication, including information contained therein, to ensure its availability, integrity, authentication, confidentiality, and nonrepudiation*". In the last decade, cyber-threats had a constantly growing impact as pointed out by technical reports [28, 29]. Consequently, ***Intrusion Detection Systems* (IDSs)** [6, 7] are becoming common building blocks when designing CPSs, to detect potential threats and trigger modules that are able to block or mitigate the adverse effects of cyber-threats. IDSs collect and analyze data from networks and systems indicators to detect malicious or unauthorized activities, based on the hypothesis that an ongoing attack has distinguishable effects on such indicators.

To detect intrusions, IDSs may adopt **Artificial Intelligence (AI)** mechanisms as signature-based algorithms [8], which search for predefined patterns (or *signatures*) in the monitored data in order to detect an ongoing attack that matches one or more signatures. Signature-based approaches are ideal when detecting known attacks [5, 6, 7]; on the other hand, they exhibit weaknesses in detecting slight variations of known attacks or brand new zero-day attacks [10], calling for a prompt update to add the signature of the novel threat. Unfortunately, this is a major weakness as CPSs may evolve during their operational life, exposing their interfaces to multiple threats that cannot be entirely defined at design time.

Dealing with Unknowns: Anomaly Detectors. This intrinsic aspect of CPSs calls for mechanisms that efficiently deal with unknown threats *(zero-day attacks* [10, 11]) as anomaly detectors. Differently from signature-based approaches, which rely on the knowledge of threats, anomaly detectors characterize the normal (expected) behavior of the system. Then, they use this knowledge to *find patterns in data that do not conform to the expected behavior of a system* (or a network) [1]: these patterns are called anomalies. Anomaly-based IDS are built on the assumption that ongoing attacks will generate observable anomalies in the features we gather from the system and the network [7] through monitoring activities. Supervised anomaly detectors which may be based on neural networks [89], decision trees [8], or gradient boosting [90] show excellent detection capabilities when dealing with known attacks. Instead, unsupervised algorithms also fit the detection of unknown attacks [5, 3] as they do not rely on labels in training data. However, detection

efficacy of anomaly-based detectors depends on their ability in precisely characterizing the system expected behavior [9]: as a consequence, they usually generate a higher number of false alarms than signature-based methods [5, 11].

The Role of Meta-Learning. Different anomaly detection algorithms usually exhibit [11] different rates of missed (**False Negatives, FNs**) and wrong (**False Positives, FPs**) detections and consequently result in different detection capabilities. Although most of such algorithms have a generic, context-independent formulation, they are often more effective to detect specific attacks on specific systems or applications [84]. Therefore, studies such as [2, 3, 4] focus on the comparison of different (unsupervised) algorithms for anomaly detection in different CPSs. In most of the cases, even the most promising algorithm still shows a number of misclassifications (either FPs or FNs, or both), which might not satisfy the requirements of a critical system. Indeed, several studies concluded that meta-learners such as Bagging (e.g., Random Forests [14], Isolation Forests [76]), or Boosting [15] may result in a lower number of misclassifications, especially within supervised learning. However, *a combination of learners does not always result in improved capabilities*: some misleading learners may let the meta-learner lean towards a misclassification, with cascading effects with respect to the whole system.

Paper Aim. *This paper systematically instantiates various meta-learning approaches through ensembles of unsupervised base-learners, discussing how the adoption of a specific meta-learning approach may help in significantly reducing the number of misclassifications* with respect to non-meta unsupervised algorithms. We first expand on specific meta-learners and on their suitability to detect (zero-day) attacks in CPSs and then we proceed to an experimental campaign to compare different approaches. Baseline data is selected among publicly available datasets that report on network intrusion detectors and biometric authentication systems, which are usually employed to enhance security in CPSs. Unsupervised algorithms selected for this study will be used both as non-meta learners and as base-level learners of meta-learning approaches to deal with the detection of unknown threats or zero-day attacks. Overall, we selected 21 datasets, 15 unsupervised algorithms and 9 different meta-learning approaches that we instantiate by considering the unsupervised algorithms above as base-learners. Results allow comparing metric scores achieved by both meta and non-meta learners on each dataset. We observe how meta-learning reduces misclassifications, consequently improving metric scores, in 20 out of the 21 datasets we used in this study. In addition, we discuss the impact of the choice of base-learners for those meta-learners which rely on multiple algorithms such as Stacking (Generalization), Cascading, Delegating, (Weighted) Voting, and Cascade Generalization. Finally, we provide conclusive statements about the advantages of adopting meta-learners to improve intrusion detection in CPSs.

Paper Structure. The paper continues as follows. Section 2 describes the state-of-the-art and the terminology to explain meta-learners. Section 3 expands on the suitability and the potential improvements following the adoption of meta-learners to build IDSs. A methodology for empirical evaluation of meta-learners is described in Section 4, and generates results that are presented and then discussed in Section 5. Section 6 concludes the paper.

2 META-LEARNERS TO COMBINE CLASSIFIERS

2.1 Meta-Learning: Definition and Purpose

Several definitions of meta-learning have been proposed in the literature [24, 25]; in this paper, we adopt the following definition [25]: "*Meta-learning is the study of principled methods that exploit meta-knowledge to obtain efficient models and solutions by adapting machine learning and data mining processes*". Consequently, a meta-learner is a learning module that uses knowledge acquired during base-learning episodes, i.e., *meta-knowledge*, to improve classification capabilities. More specifically [26], a *base-learning* process starts feeding dataset (given) features into one or more

learning algorithms to derive one or more models to be used for classification at a first stage. Results of base-learners partially build *meta-data* that is provided alongside with other features to the *meta-layer*, which provides the classification result of the whole meta-learner.

2.2 Meta-Features

As a result, the potential improvement of a meta-learner with respect to a base-learner is related to the quality of meta-data, or rather meta-features that describe attributes of both dataset and base-learners [25, 26, 24]. Meta-features have been classified into five separate groups [24] as follows:

- *Simple* or *given* features: dataset features that can be normalized to provide data that suits the learner.
- *Statistical* features: they describe the statistical properties of a dataset e.g., the number of data points or features, the Kurtosis and Skewness indices, and correlation between independent features.
- *Information-theoretic* features, which explicitly calculate the entropy of simple features, to obtain quantitative estimations of their relevance in terms of information gain.
- *Landmark features* define regions of the dataset where a base-learner fits the best. These regions can be used to partition the input space mapping regions to the learner that is the most adequate to process such data.
- Finally, *model-based* meta-features are derived from the models produced after training by the learning algorithms e.g., the number of vectors in SVMs, nodes and leaves when a decision tree is induced, or the number of rules when a rule-based learner is employed.

2.3 Meta-Learning Approaches

Regardless of the possible ways to generate meta-features, different features may be more suited to describe specific domains. Amongst all these domains, *we focus on techniques for model combination* [25], where the meta-learner can either play as (i) an adjudicator, with a meta-layer that combines individual results from different classifiers at base-layer (e.g., Stacking), (ii) averaging results of base-learners that are created according to specific criteria as Bagging or (Weighted) Voting, or (iii) relying on more sophisticated instantiations of both base and meta-layer e.g., Cascading, Boosting. With the aid of Figure 1, the remainder of this section highlights *bagging, boosting, Stacking (Generalization), cascade* (*generalization*), *delegating, arbitrating* and *(weighted) voting*, which build the pool of possible meta-learners for model combination according to our literature review.

2.3.1 Bagging. Bagging combines base-learners of the same type by submitting bootstrap replicas of the training set [16]. The unified result of the ensemble is derived by majority voting the individual results of base-learners. Individual learners execute the same algorithm, but are fed with different training subsets created through random sampling with replacement i.e., *bootstrap sampling*. Consequently, Bagging embeds pseudo-independent learners: since subsets are taken from the same training set, classifiers built on these training sets might not give independent outputs.

2.3.2 Boosting. Boosting builds ensembles of *weak learners*. Overall capabilities of a weak learner are only slightly better than random guessing: the underlying idea of boosting is to *orchestrate several weak learners to build a strong meta-learner* [17]. Each weak learner is trained *hierarchically* to discriminate more complex regions in the feature space. Initially, Boosting trains a single (weak) classifier and assigns the same *weight* to all data. The weight of a data point in the training set defines its probability of being selected for training: intuitively, the trickier the classification of a data point, the higher the weight. As a result, subsequent weak learners tend to be trained by using hard-to-classify data points, which have high weight. The process ends either

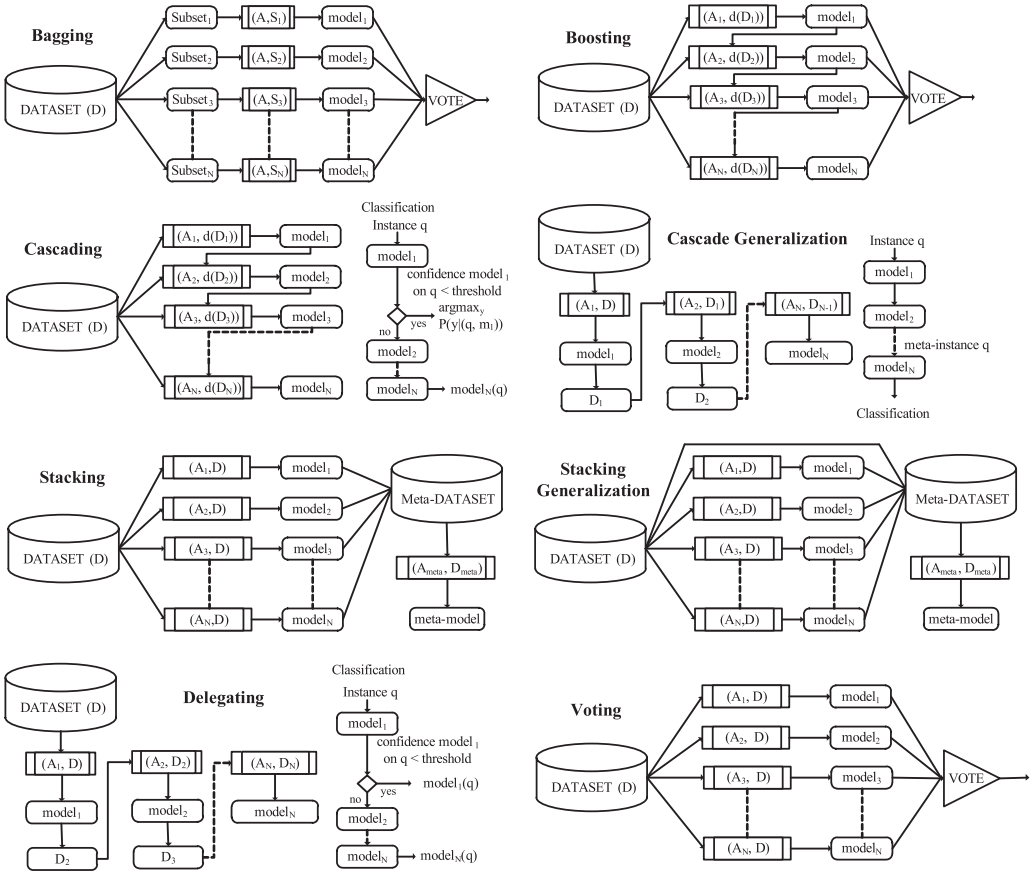

Fig. 1. Meta-Learning schemes, showing source dataset (eventually partitioned into subsets S_i or derived datasets D_i), algorithms Ai, and the models M_i they learn, for Bagging, Boosting, Stacking (Generalization), Cascade (Generalization), Delegating and Voting.

if (i) a fixed number of iterations is achieved, or (ii) the total weight of "hard" data points exceeds a threshold [25]. Similarly to bagging, the meta-level output is obtained through majority voting of the outputs of individual weak learners.

2.3.3 Stacking (Generalization). Stacking [18] relies upon different algorithms, trained with the exact same training set, as base-learners. Their outputs become model-based meta-features, which are fed to another independent classifier, the meta-layer classifier, to deliver a unified result. In this paper, we differentiate between (i) stacking, where the meta-dataset is only composed of model-based features, and (ii) stacking generalization, where also simple (dataset) features are added to the meta-dataset. Differently from bagging and boosting, the final output is not obtained through majority voting: the independent classifier combines individual results according to a general (and possibly non-linear) function. A new unknown data point is first processed by base-learners in parallel to create the meta-data, which is then provided to the meta-layer classifier to produce the stacking result.

2.3.4 Cascading. The Cascading schema, proposed in [19], stems from boosting by employing heterogeneous learners in order to increase the system complexity. Initially, all instances have

the same weight, and therefore the initial dataset distribution is uniform. A first model is created through the first base-level algorithm. According to the *confidence* of the first learner, all instances are reweighted. Then, other base-level (weak) learners are sequentially trained from the same dataset, with the distribution of the weights determined by the previous learner, until a threshold is reached. During validation, a data point is sequentially sent through the sequence of classifiers. The first classifier who is confident enough produces the output of the cascading meta-learner.

2.3.5 Cascade Generalization. Cascading generalization [20] performs a sequential composition of heterogeneous classifiers. Instead of a parallel use of classifiers as in Stacking, Cascade Generalization sequentially applies classifiers such that the *i-th* classifier is a meta-level classifier for the previous learner. The inputs of the *i-th* classifier consist of simple features and *i-1* model-based features that represent the outputs of the previous learners. In other words, each classifier of the sequence is both a base-learner and a meta-learner. A new data point is progressively extended as meta-instance as it collects all meta-data produced in each level of the sequence. The unified meta-result of cascade generalization is the output of the last learner.

2.3.6 Delegating. Delegating [21] is similar to Cascade Generalization as it employs a sequence of heterogeneous classifiers which are meant to be exercised sequentially for each data point. However, Delegating uses a more cautious approach: if a classifier is not confident enough about its result, the final decision is delegated to the next classifier. The process of Delegating ends when a fixed number of delegation steps is reached, or if a classifier is confident enough about its result.

2.3.7 (Weighted) Voting. Voters count opinions coming from different independent sources, and provide the final decision based on a linear aggregation (sum) of individual responses. It has been widely used in *N-version programming* [30] to aggregate results that are independently generated by multiple functionally equivalent programs that are built upon the same initial specifications. Voters decide on a ***k out of N* (kooN)** rule as follows: when at least k out of N replicas ($k \leq N$) agree on a result, the result is chosen as the result of the ensemble. Common configurations are majority voting ($k \geq N/2$), and overall agreement, where $k = N$. For binary decisions, voting has been extended to a *weighted formulation* that gives a positive outcome of the ensemble if and only if the cumulative weight of all individuals that give a positive result reaches or exceeds a given *threshold of cumulative weights* [31].

2.3.8 Arbitrating. The underlying idea of Arbitrating is that different learners map different areas of expertise in which they perform better classifications i.e., with a higher confidence [22]. The area of confidence of each classifier is described by a *referee*, typically a decision tree. Differently from delegating, all classifiers are trained in parallel on the same, initial dataset. When a data point needs to be classified, all learners are exercised to provide their output and related confidence. Different referees may suit this approach. For example, in [22] a referee is induced for each individual component classifier; then, each referee assigns a confidence value to its corresponding classifier, letting the learner with the highest confidence to make the final classification. In [23], a binary tree of arbiters, called arbiter tree, is generated with the classifiers at the leaves level. When an unknown instance is classified by the arbiter tree, predictions propagate upwards from the leaves (base-learners) to the root, with arbitration taking place at each intermediate level.

2.4 Weak Learners as Base-Learners

Overall, we can notice that different meta-learners orchestrate ensembles of base-learners in many different ways, and often rely (i.e., Boosting, Cascading) on weak learners [17] to build a strong meta-learner that aims at improving detection capabilities of non-meta learners. Therefore, we

should be able – when needed – to configure algorithms that act as base-learners as weak classifiers through adequate parameters combinations. This allows to instantiate the same base-learner as weak learner to suit specific meta-learners, while others may prefer relying on strong base-learners. For example, Decision Trees build strong learners without constraints on the depth of the tree and the number of branches, but may become weak learners when limiting their maximum depth with a small value. The same goes for many unsupervised algorithms as clustering [70] or neighbor-based [66] algorithms, which may act as weak learners when accounting for a few clusters / neighbors and using small training datasets.

Unfortunately, not all the unsupervised algorithms that have been proposed in the literature can be instantiated as weak learners. For instance, deep learners are considered very strong individual learners: they use **Convolutional Neural Networks (CNN)** in combination with different representation learning techniques, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks [93], Auto-Encoders [92] and deep Boltzmann machines [91]. Training phase may be either built from scratch, requiring massive computational power and many data, or adopt *transfer learning* [94], which allows to customize an existing general-purpose network to the specific domain or system. In both cases, the resulting deep neural network is by definition "deep", with many hidden layers and interconnections, and does not fit at all the purpose of weak learners. Therefore, those algorithms cannot be always instantiated as base-learners, limiting their usage as building blocks of meta-learners.

2.5 Related Works on Meta-Learning for IDSs

Various meta-learners were occasionally used in the literature for Intrusion Detection. In [53], the authors used both a combination of output coming from different Random Trees and Voting as meta-learning algorithms. Their results showed that the ensemble of random trees had lower accuracy than the Bayesian network used as a comparison, while voting had comparable scores, i.e., Bayesian reached 99.32% of accuracy, while voting reached 99.82%. Instead, Tama et al. [54] describe the construction of two-layer classifiers as rotation forest and bagging; the results of the base-level were aggregated through a majority voting scheme. By using the NSL-KDD and UNSW-NB15 datasets, they achieved an accuracy of 85.8%, a precision of 88%, a sensitivity of 86.8%, and a False positive rate of 11.7%. These scores show a lower overall number of misclassification with respect to algorithms such as **Support Vector Machines (SVM)** or Decision Trees.

The Stacking generalization ensemble has been used in [55] to perform anomaly detection in the NSL-KDD dataset, achieving noticeably high accuracy of 97%, and F1 score of 98% outperforming the performance of any base classifiers. Similarly, a stacking system using kNN, Logistic Regression and Random Forests as base learners, and SVM as meta-learner was proven [58] to be useful in UNSW-NB15 and UGR16 datasets, achieving very competitive scores concerning related works on the same datasets.

Multiple meta-learning techniques were investigated in [56], where the authors instantiated Stacking, Voting, Boosting and Bagging and compared them to a Multi-layer perceptron, kNN and Decision Tree by using data from UCI repository. Results show that Bagging achieved the highest metric scores compared to other meta-learners, e.g., an accuracy of 99.97%, a False Positive rate of 0.00018%, while other meta-learners resulted in scores comparable with respect to non-meta algorithms, without highlighting any noticeable improvements.

Lastly, Alaba et al. [57] considered the old KDDCup99 and NSL-KDD dataset and proposed a stacking ensemble-based strategy relying on Naïve Bayes, Random Forest and C.45 as base classifiers, with SVM managing the meta-layer. The training of the SVM classifier performed on the

output probabilities of the base classifiers. The study concludes that the scores achieved by stacking, with an accuracy and a precision reaching 99.5% constituted a significant improvement with respect to other meta-learners as Bagging and Boosting, which achieved an accuracy below 84%.

3 SUITABILITY OF META-LEARNING FOR INTRUSION DETECTION

We discuss here the main characteristics of meta-learning approaches in the previous section to understand if they could be successfully applied as IDSs. We start discussing their individual aspects, starting with a categorization based on heterogeneity, and proceeding with meta-features, usage of confidence and applicability to data streams, ending up summarizing the suitability of meta-learning systems to intrusion detectors.

3.1 Categorization of Model-Combiners

To summarize and wrap-up the digression on the existing meta-learning techniques for combining models, we partition such methods into three categories depending on the base-level learners they rely upon.

- **Single Classifier (SC):** *Bagging, Boosting*. These meta-learners build ensembles of base-learners that rely on the same homogeneous algorithm, but are trained with different portions or feature sets that are extracted from the training dataset.
- **Multiple Classifiers (MC):** *Stacking (Generalization), Voting (Weighted)*. These meta-learners use heterogeneous classifiers to build base-level learners. The way they aggregate individual results of base-learners into the meta-result does not depend on the order of base-level learners.
- **Multiple Classifiers with Ordering (MCO):** *Cascading (Generalization), Delegating*. These meta-learners produce the final result depending on subsequent operations that involve heterogeneous base-level classifiers. Results are collected sequentially from each base-level learner, and therefore the ordering may impact the final outcome of the technique.

With respect to Arbitrating, it is worth noting that *the way the referee is implemented heavily impacts* the realization of this meta-learner. If the referee is a counter, this approach degenerates into voting, while using a separate independent classifier may let arbitrating to overlap with stacking. To such extent, we disregard considering it further, as the usage of a sub-optimal function may lead this meta-learner to perform poorly, limiting their role in our comparison study.

3.2 Usage of Meta-Features

Meta-features are usually associated with typical issues that may appear and impact their applicability or usability [25]. Main concerns and limitations to the usage of meta-features are: (i) the discriminative power of meta-features, (ii) their number, which should be controlled as meta-learning is very sensitive to the curse of dimensionality [26], and, finally, (iii) the computation complexity to calculate meta-features. For intrusion detection, this translates into two practical implications. First, the base-level should not be composed by a very wide range of base-learners, to limit the number of meta-features that are being generated. Approaches such as stacking that may consider as meta-features both base-learners results and dataset features (i.e., stacking generalization), may escalate into a number of meta-features that the meta-learner could not process efficiently. A similar constraint should be put on cascade generalization, to limit the generation of additional meta-features i.e., one for each step of the process.

Nevertheless, base-learners *should employ solid algorithms* to raise the discriminative power of the meta-features they generate, and *should not require too much time for training:* algorithms as

ABOD [65], which show cubic complexity for training, should not take part to the base-level of a meta-learner.

3.3 Heterogeneity and Diversity of Base-Learners

Differently from SC meta-learners as Bagging and Boosting, MC(O) meta-learners relying on heterogeneous classifiers to be used as base-learners heavily depend on how these classifiers interact together. *Intuitively, if an intrusion is not detected by any of the base-learners, no matter how we combine their individual results it would not be possible to identify it by just ensembling independent classifiers.* More in detail, it would not be possible at all with (weighted) voting, delegating and cascading (Generalization), while stacking offers a few – albeit very limited – opportunities. In particular, Stacking relies on the numeric scores that are provided by classifiers, before a decision function is applied to convert the numeric score into boolean. Small fluctuations of these scores may not be enough to let individual base-level classifiers to detect the data point as anomalous, but the meta-level classifier used in stacking may have room to identify the intrusion.

Nevertheless, it is acknowledged that base-level classifiers should be diverse [31]. This helps avoid common-mode failures [13], which translate in avoiding misclassifications of the same data point by multiple and potentially independent classifiers. In our context, it is very easy to select a group of two or more different classifiers. However, many classifiers – despite being different – are hardly diverse as they rely on the same intrinsic mechanisms e.g., k-nearest neighbor search is used by many algorithms either as a main mechanism or to lower computational complexity. Therefore, instantiating MC(O) meta-learners require choosing a set of base-learners that are diverse enough.

3.4 Applicability to Data Streams

Another important dimension of our problem is the suitability of meta-learners to analyse streaming data. Usually, IDSs monitor network and system features e.g., bytes sent/received, packets, at runtime, and they are meant to provide answers promptly. As a result, meta-learners that sequentially execute base-learners, or rely on algorithms with high computational complexity required to decide on a single data point, should be used cautiously when dealing with data streams. It turns out evident how meta-learners such as Cascade Generalization, Delegating and Arbitrating may take too much time to decide on a data point at runtime, given the sequential execution of the base-learners. However, this negative aspect may be mitigated by using a reduced set of learners.

3.5 Confidence of Classification

Binary classifiers are meant to classify a data point either as expected or anomalous, i.e., pointing to an attack. However, in some cases they may not be confident enough whether a data point belongs to one of the two classes. Forcing algorithms to answer also if they are not sufficiently confident may let the critical system incur major problems. Therefore, recent studies such as [27] point to mechanisms and metrics that wrap the binary decision, converting it into a ternary {yes, no, not sure}, and re-modulating the confusion matrix accordingly. In a nutshell, classifiers should answer either “yes” or “no” if and only if they are confident enough. Otherwise, the system may activate back-up strategies, i.e., calling the administrator or switching off some interfaces to temporary move to a safe state.

Meta-learners as Delegating and, to a lesser extent, Arbitrating (through the referee), intrinsically rely on the “confidence” of classifiers, and therefore are meant to answer only when they are confident enough. Moreover, Boosting and Cascading employ the idea of confidence during training, to trigger new iterations that generate additional weak learners.

Table 1. Suitability of Meta-Learners to Intrusion Detectors

Meta-Learner	Category	(Meta)Features	Classifiers Diversity	Suitability to Data Streams	Confidence	Usage
Bagging	SC	Simple		✓		Widespread
Boosting	SC	Simple				Widespread
Stacking	MC	Model-Based	✓	✓		Uncommon
Stacking Generalization	MC	Simple, Model-Based	✓	✓		Uncommon
Cascading	MCO		✓		✓	Uncommon
Cascade Generalization	MCO		✓		✓	Rare
Delegating	MCO	Simple	✓		✓	Rare
Voting	MC	Model-Based	✓	✓		Common
Weighted Voting	MC	Model-Based, Statistical	✓	✓		Uncommon

The tick mark denotes the perfect capability of a meta-learner. Other cases point to sub-optimal capabilities.

3.6 Meta-Learners for IDSs

Summarizing, most of the meta-learners in this paper have the potential to improve the detection capabilities of IDSs, under adequate constraints. We pointed out relevant characteristics in Table 1, which wraps up on the discussion we carried out in this section and puts the basis for experimental evaluation in the rest of the paper. Bagging in its pure formulation does not allow a diversity of classifiers used as base-learners, but relies on sampling to create different learners. Cascade Generalization, Delegating,and Arbitrating are able to provide results that intrinsically revolve around their confidence. Boosting and Cascading are promising despite time-consuming training phases, while Stacking has the most flexibility out of the possible approaches, and can find applicability when implementations of heterogeneous algorithms are available. All meta-learners in the table could find usability, given that the number of base-learners does not exceed a (reasonably low) limit, for performance constraints.

4 METHODOLOGY TO EXERCISE EXPERIMENTS

To substantiate and elaborate on how and when different meta-learning approaches may improve intrusion detection capabilities, we planned and executed an experimental campaign as follows:

We collect public datasets that contain data concerning attacks related to security aspects that embrace critical systems. In particular, we refer to data related to network intrusion detection and biometric authentication.

Then, we review the literature to identify unsupervised algorithms that are suitable for anomaly detection. The chosen algorithms will be used both individually and as base-learners of meta-learning strategies.

After, we apply each (meta-)algorithm to each dataset, collecting metric scores. These metrics describe detection capabilities of (meta-)algorithms on each dataset, accounting for misclassification and allowing for discussion of results.

Once the experiments have been executed, experimental data should allow exploring (i) detection capabilities of (meta-)algorithms across all datasets, (ii) impact of the choice of base-learners on detection capabilities of MC(O) meta-learners, and (iii) comparison of meta-learners with respect to unsupervised (non-meta) algorithms, plus an overall comparison of all algorithms considered in this study.

Different inputs are needed to execute our experimental campaign. First, Section 4.1 reports on publicly available attack datasets that contain data relevant to CPSs. Section 4.2 describes the

Table 2. Datasets Used in this Study

Domain	Dataset Name	Year	# Attacks	Simple (Given) Features			% Attacks (Evaluation)	Ref
				Initial	Numeric	Ordinal		
IDS	Netflow_IDS	2015	3	11	5	5	11.3	[39]
IDS	AndMal17	2017	4	85	80	77	15.5	[41]
IDS	CICIDS17	2017	5	85	80	77	79.7	[38]
IDS	CICIDS18	2018	6	85	80	77	26.2	[38]
IDS	CIDDS	2015	4	16	7	5	14.4	[36]
IDS	CTU13	2013	1	16	8	6	0.4	[40]
IDS	ISCX12	2012	4	16	6	4	43.5	[34]
IDS	NGDIS-DS	2015	7	9	4	3	5.3	[39]
IDS	NSLKDD	2009	4	42	37	37	40.7	[35]
IDS	UGR16	2016	5	13	7	4	3.3	[42]
IDS	UNSW-NB15	2015	8	45	39	38	6.5	[37]
Bio	Fingerprint	n.a.	4	images	15	12	9.2	[43]
Bio	Face	2017	4	30	30	30	8.7	[51]
Bio	Keystroke Tappy	2017	4	8	4	3	9.5	[45]
Bio	HRV(SWELL)	2014	4	66	65	64	10	[46]
Bio	HRV(WESAD)	2018	4	62	62	62	9.9	[47]
Bio	Human Gait	2019	4	70	42	37	8.8	[49]
Bio	EDA(SWELL)	2014	4	54	52	52	10.1	[46]
Bio	EDA(WESAD)	2018	4	95	49	44	7.8	[47]
Bio	Voice	2018	4	21	20	20	10.3	[50]
Bio	Hand Gesture Kinect	2015	4	95	95	94	8.5	[48]

We report on the domain, name and release year. We then report on the categories of attacks they contain, with details on features and % of attacks in the portion we used for evaluation (training goes unsupervised – no label needed albeit it is often provided).

metrics that will be used to evaluate detection capabilities of algorithms on the datasets above. Then, we briefly introduce unsupervised anomaly detection algorithms in Section 4.3, leaving Section 4.4 to expand on meta-learners. Further details on the implementation of the experimental campaign are summarized in Section 4.5, which completes the digression on the experimental campaign and makes room for discussions in the rest of the paper.

4.1 Publicly Available Datasets

We report below on the public datasets that we used in our study. Note that we focused on two different intrusion detection domains: *Network attacks* and threats to *Biometric authentication* processes: both domains are related to security, albeit they have (slightly) different characteristics. Table 2 summarizes the 21 datasets involved in this study, reporting domain, name, year, number of attacks, % of attacks and reference, as well as an insight on the number of available features (both textual and numeric), showing also the subset of ordinal features (i.e., numeric, not categorical) that can be used by algorithms.

Network Intrusion Detection. Starting from recent surveys and taxonomies that expand on datasets for intrusion detection such as Hindy et al. [85], Khraisat et al. [86], Ring et al. [33], and by querying online portals[1,2] we selected datasets with the following characteristics: (i) published

[1]Intelligence and Security Informatics Datasets, https://www.azsecure-data.org/other-data.html.
[2]UNB – Canadian Institute for CyberSecurity, https://www.unb.ca/cic/datasets/index.html.

recently, (ii) labeled (at least partially), and (iii) contain attacks in the ENISA [32] top 10, to ensure they report on relevant attack data. Our selection process resulted in the following datasets NSL-KDD (2009) [35], CTU-13 (2011) [40], ISCX12 (2012) [34], UNSW-NB15 (2015) [37], UGR16 (2016) [42], NGIDS-DS (2017), Netflow-IDS (2017) [39], AndMal17 (2017) [41], CIDDS001 (2017) [36], CICIDS17 (2017) [38], and CICIDS18 (2018) [38].

Biometric Authentication. We focused on datasets containing textual information rather than images or audio tracks as they require a custom transformation into textual features. The only exception has been the fingerprints dataset [43], where we processed the images by using state-of-the-art feature extractors [44]. This research process, conducted by two people independently, identified 10 datasets related to 8 different biometric characteristics. The datasets pertain to different biometric characteristics, namely: Fingerprint (CASIA dataset [43]), Voice [50], Face [51], Heart Rate Variability (SWELL [46] and WESAD [47] datasets), Electro Dermal Activity (SWELL [46], WESAD [47]), Human Gait (activity recognition [49]), Keystroke (Tappy, [45]), and Hand Gesture (Kinect LeapMotion [48]).

While most of these datasets report on relevant numberss of features and data about each biometric system, none of the datasets above contains data collected when the system was under attack. To such extent, we injected attacks by mutating or inserting some data points of datasets according to four categories of attacks we derived starting from surveys [79], [80], [81]. The attack model and the injection process are described in Annex A.

4.2 Metrics for Evaluation

The effectiveness of anomaly detectors is usually assessed using correct classifications (**true positives TP, true negatives TN**) and misclassifications (**false negatives FN, false positives FP**), which build the so-called *confusion matrix*. As in [59], [60], aggregated metrics as **Precision, Recall (or Coverage), False Positive Rate (FPR), Accuracy (ACC), FScore-β (Fβ), F-Measure (F1)**, **Area Under ROC Curve (AUC)** and **Matthews Coefficient (MCC)** are used in different studies, depending on the domain. FP-inclined metrics such as Precision, FPR and F-Score (with $\beta < 1$), are relevant when the number of false alarms needs to be as low as possible e.g., to increase usability, while Recall and F-Score (with $\beta > 1$) are more relevant in those systems where FNs may constitute severe threats e.g., safety-critical systems, which also need domain-specific metrics [27].

For the sake of generality and ease of comparison with other studies, in this study we mainly report on widely used metrics that weight FPs and FNs as equally undesired, i.e., F-Measure, Accuracy and MCC. However, with unbalanced datasets, i.e., if a file reports on many attacks data and a few normal data, some of the metrics above can be misleading [61], since they either (i) do not consider all the four classes of the confusion matrix, i.e., F1, FScore-β, or (ii) consider all the classes without weighting the size of trues and falses, i.e., Accuracy. To such an extent, this paper primarily focuses on *MCC*, which does not suffer from the weaknesses mentioned above [87], [88].

4.3 Unsupervised Algorithms and Base-Learners

We chose a set of unsupervised anomaly detection algorithms to perform intrusion detection on the selected datasets. To provide a broad picture about detecting attacks in different datasets, we selected a pool of algorithms that are as heterogeneous as possible. To such an extent, we refer to algorithms' families from [1], which were used also in other studies [2–4]: clustering, neural networks, density-based, neighbor-based, statistical, and classification. Then, we selected algorithms belonging to as many families as we could, disregarding algorithms with high computational complexity, e.g., ABOD [65], as this study already builds on meta-learning, which naturally requires more computing and memory resources. Out of available algorithms, we selected a mixture of 15 as follows:

- *One algorithm per Family*: One-Class SVM (classification family) [67], K-Means (clustering) [78], kNN (neighbor, unsupervised variant) [75], HBOS (statistical) [64], SOM (neural-network) [69].
- *Algorithms that belong to many families*: Neighbors identification is employed to reduce noise and computational complexity in the angle-based FastABOD [65], and in the density-based LOF [74] and COF [72]. Interesting mixtures of clustering and density-based families allow devising DBSCAN [73] and LDCOF [71], which builds a density-based anomaly detector on top of a clustering procedure.
- Others such as Isolation Forests [76], Stochastic Outlier Selection (SOS) [77], G-Means [70], ODIN [66], Sparse Density Observers (SDO) [68] to complete this selection of unsupervised algorithms for binary classification.

In addition, we looked for public frameworks that allow running unsupervised algorithms on datasets. After examining different options, we chose RELOAD [62], an open-source tool that wraps unsupervised algorithms from ELKI[3] and WEKA,[4] and adds more implementations of unsupervised algorithms. The tool also implements grid searches to find adequate values to assign to algorithms' parameters.

4.4 Meta-Learners

Our study embraces all nine meta-learning strategies in Table 1 to have a broader picture of the capabilities of meta-learners when dealing with intrusions. Different setups are needed, depending on their characteristics.

Before going through setups for SC, MC and MCO categories, we remark here that we would need a study to test all the possible sets of base-level classifiers to identify the optimal set of base-learners. However, *finding the optimal or more diverse set of classifiers, or the set where they have a good synergy, is a separate task that takes an amount of time and resources that is potentially unlimited*. For example, the reader can think about all the possible combinations (or even permutations, with MCO) of size in the range [2, 15] that can be created with our 15 unsupervised classifiers. To such extent, in this study we arbitrarily sampled four different groups of classifiers to be used as base-level learners as described in the MC subsection below.

SC Meta-Learners. Bagging and **Boosting** meta-learners are built upon each of the 15 algorithms by using different parameters combinations. More in detail, they are instantiated with {10, 20, 50} learners and exercised independently. More specifically, for each algorithm *alg*, we will instantiate *Bagging(alg, 10), Bagging(alg, 20), Bagging(alg, 50)* and *Boosting(alg, 10), Boosting(alg, 20), Boosting(alg, 50)*.

MC Meta-Learners. (Weighted) Voting and Stacking (Generalization) rely on multiple heterogeneous base-learners to build meta-features for the meta-level, which is realized either as counter for (Weighted) Voting or as another learner for Stacking (Generalization). The choice of base-learners has for sure a relevant impact on the behavior of the learner. For this study, we selected four unordered sets of base-learners as follows. We chose the 3 unsupervised algorithms which show the higher average MCC (G1 - first group), the higher average Accuracy (G2 - second group) and the higher average Recall (G3 - third group) in our experiments. Moreover, we chose an additional fourth group G4 which includes 7 algorithms - one algorithm for each family – namely {SVM, HBOS, G-Means, SDO, ODIN, SOM, FastABOD}. G4 was selected to guarantee diversity of families of classifiers, while groups G1 – G3 were selected depending on the results

[3]ELKI Project - https://elki-project.github.io/.
[4]Weka 3: Data Mining Software in Java - www.cs.waikato.ac.nz/~ml/weka/.

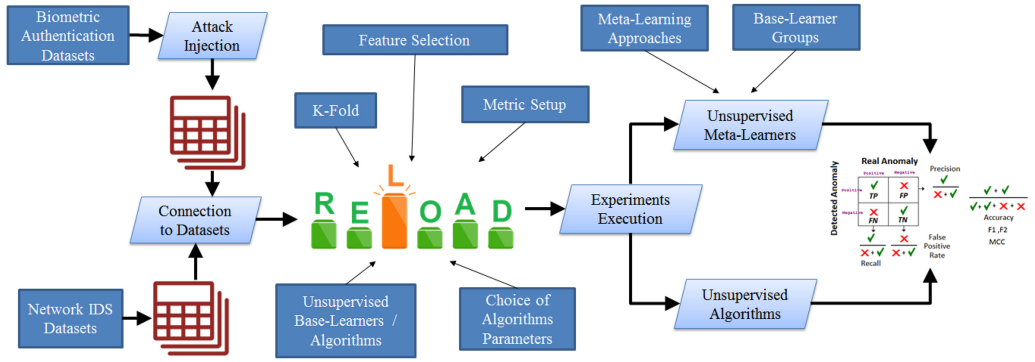

Fig. 2. Building blocks of the experimental setup and methodology used in this paper.

of the experimental campaign as "good on average" algorithms that potentially could build a set of base-learners with good synergy for a good meta-learning process. Discussion on the impact of the choice of the groups will be expanded later in the paper.

MCO Meta-Learners. Delegating, Cascading, and **Cascade Generalization** differ from MC learners as ordering of base-learners matter. As a result, we will generate different permutations starting from the sets of algorithms G1 – G4. When choosing 3 bae learners {alg1, alg2, alg3} as in G1, G2, G3, we will instantiate each meta-learner 3 times by sorting algorithms as follows: *<alg1, alg2, alg3>*, *< alg2, alg3, alg1>*, *<alg3, alg1, alg2>*. When using 7 algorithms (G4), we will instantiate each meta-learner 7 times by rotating the algorithms similarly to the example above: *<alg1, alg2, alg3, alg4, alg5, alg6, alg7>*, *<alg2, alg3, alg4, alg5, alg6, alg7, alg1>*... In addition to instantiating each MCO learner with each of the permutations of base-learners above, each algorithm has its own parameters as the confidence threshold to build the final score. As RELOAD outputs algorithms confidence as a number between 0 (no confidence) to 1 (maximum confidence), we alternatively use {0.90, 0.95, 0.99} as minimum confidence to stop the process. For example, when instantiating delegating with a set of three algorithms, 9 instances are created as follows: *Delegating (<alg1, alg2, alg3>, 0.90), Delegating (<alg1, alg2, alg3>, 0.95), Delegating (<alg1, alg2, alg3>, 0.99), Delegating (<alg2, alg3, alg1>, 0.90), Delegating (<alg2, alg3, alg1>, 0.95), Delegating (<alg2, alg3, alg1>, 0.99), Delegating (<alg3, alg1, alg2>, 0.90), Delegating (<alg3, alg1, alg2>, 0.95), Delegating (<alg3, alg1, alg2>, 0.99).*

4.5 Experiments Setup and Execution

We describe here the experimental setup for our study with the aid of Figure 2.

Datasets/Tool Download. We downloaded the datasets in Section 4.1 from their repositories shaping them as CSV files, performing attack injection for datasets related to biometric authentication. Then, we downloaded the latest release of RELOAD, setting up *connectors to datasets*.

Metric Setup. We adopted MCC as the target metric: while this metric is used by RELOAD to find optimal parameters values of algorithms, metrics other than MCC (see Section 4.2) are still reported as output and will be discussed in the rest of the paper.

Feature Selection. According to literature studies, out of the feature selection strategies made available by RELOAD, we chose Information Gain [63], to extract the n most relevant features out of each dataset. Since several datasets have just a few ordinal features (see *NGDIS-DS* and *Keystroke Tappy* in Table 2), we set $n = 3$.

Cross-Validation (K-Fold). We proceeded with a 10-fold sampling of the training set as widely suggested in the literature.

Choice of Algorithms Parameters. Besides G-Means, which does not rely on parameters, we tried different combinations of parameters for each algorithm through grid searches. Grid searches were automatically managed by RELOAD, which selects the combination of parameters that allows obtaining the best MCC score in a small portion of dataset (not overlapping with the evaluation set), which was used for testing. More in detail, k for kNN-based algorithms, samples s to and trees t building iForest, number *hist* of histograms in HBOS and observers *obs* of SDO were chosen in the set {1, 2, 3, 5, 10, 20, 50, 100, 200}. Other algorithms have specific parameters: one-class SVM may be created either with {linear, quadratic, cubic, radial basis function} kernels and nu – which impacts the number of support vectors to be created – in {0.01, 0.02, 0.05, 0.1, 0.2}. In addition to *obs*, SDO needs also a q ϵ {0.05, 0.1, 0.2, 0.5} thresholds that the algorithm uses to derive "closest" observers. Lastly, DBSCAN clustering uses a combination of the minimum number of data points in a cluster *pts* ϵ {1, 2, 3, 5, 10} and *eps* ϵ {100, 200, 500, 1000}, which defines the radius of the cluster around each data point.

Unsupervised Base-Learners. We completed the setup of RELOAD by submitting a list of unsupervised algorithms that may also be used as base-learners in meta-learning experiments. As described in Section 4.3, we employed the following 15 algorithms: One-Class SVM, K-Means, kNN, HBOS, SOM, FastABOD, LOF, COF, DBSCAN, LDCOF, Isolation Forests, SOS, G-Means, ODIN, and SDO. Grid searches to select optimal values of parameters will use a subset of the parameters in the previous paragraph to instantiate weak base-learners, as motivated in Section 2.4. For example, k for kNN-based algorithms, samples s to and trees t building iForest, number *hist* of histograms in HBOS and observers *obs* of SDO were chosen in the set {1, 2, 3, 5, 10} for (weak) base-learning.

Experiments Execution. Once all the parameters mentioned above were set, we ran the experimental campaigns including all the datasets and (meta-)algorithms considered in this study. The experiments were executed on a server equipped with Intel Core i7-6700 with four 3.40GHz cores, 24GB of RAM and 1TB of user storage. Overall, executing the experiments required approximately 30 days of 24H execution.

Experiments Execution – Unsupervised Algorithms. We executed all the 15 algorithms on all the 21 datasets, obtaining a total of 315 triples <*algorithm, dataset, metric_values_on_validation*> for unsupervised algorithms.

Experiments Execution – Unsupervised Meta-Learners. Moreover, we repeated meta-learning analyses on the same twenty-one datasets, with additional 900 Bagging and Boosting experiments, 80 for each MC, and 960 each for MCO meta-learner, obtaining a grand total of 5,585 confusion matrixes (and sets of metric scores) to be presented and discussed in the next sections. As described in Section 4.4, we conducted experiments using nine different meta-learning approaches by using (when necessary) four groups of base-learners.

All the metric scores and files that we used to collect and summarize values are publicly available at [82].

5 RESULTS AND DISCUSSION

This section is devoted to the presentation, discussion and analysis of the results. We start in Section 5.1 by commenting on metric scores achieved by non-meta algorithms, which also helped choosing groups of base-learners G1 – G4 for MC and MCO meta-learners as described in Section 4.4. The impact of the choice of base-learners on the detection capabilities of MC and MCO meta-learners is debated in Section 5.2. Ultimately, Section 5.3 steps into a deep comparison of the results of meta-learners with respect to non-meta algorithms, commenting on metric scores and highlighting relevant differences.

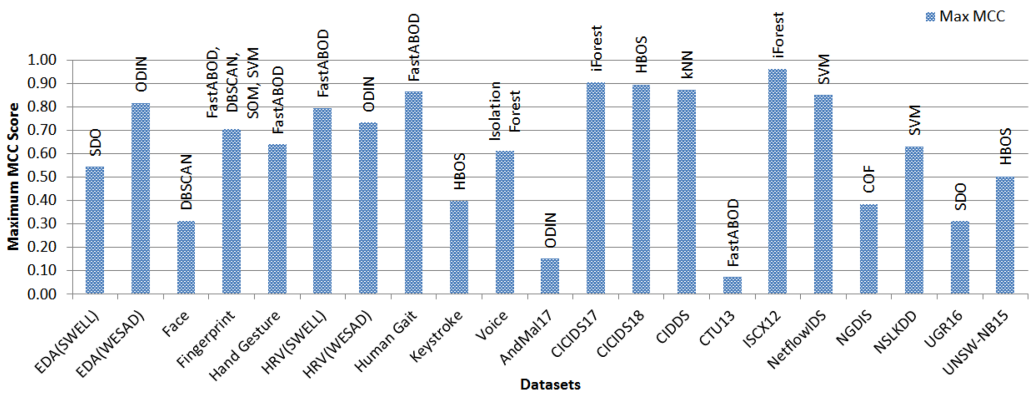

Fig. 3 Bar chart showing the highest MCC for each dataset. Algorithm(s) that reach the highest score are shown as labels on top of the chart.

Table 3. Unsupervised (Non-meta) Algorithms that Reach Maximum MCC for Each Dataset in this Study

Domain	Dataset	Algorithm(s)	Parameter(s)	FPR	P	R	ACC	F1	F2	MCC
Bio	EDA(SWELL)	SDO	obs = 50, q = 0.1	0.009	0.827	0.401	0.93	0.54	0.45	**0.55**
Bio	EDA(WESAD)	ODIN	k = 100	0.014	0.833	0.833	0.97	0.83	0.83	**0.82**
Bio	Face	DBSCAN	pts = 1, eps = 100	0.004	0.765	0.149	0.92	0.25	0.18	**0.32**
Bio	Fingerprint	DBSCAN	pts = 1, eps = 100	0.000	1.000	0.508	0.99	0.67	0.56	**0.71**
Bio	Hand_Gesture	FastABOD	k = 5	0.004	0.917	0.478	0.95	0.63	0.53	**0.64**
Bio	HRV(SWELL)	FastABOD	k = 5	0.004	0.946	0.700	0.97	0.80	0.74	**0.80**
Bio	HRV(WESAD)	ODIN	k = 50	0.004	0.938	0.606	0.96	0.74	0.65	**0.73**
Bio	Human Gait	SVM	ker. RBF, nu = 0.02	0.000	1.000	0.756	0.99	0.86	0.79	**0.87**
Bio	Keystroke	HBOS	hist = 50	0.007	0.786	0.232	0.92	0.36	0.27	**0.40**
Bio	Voice	iForest	t = 20, s = 50	0.011	0.839	0.495	0.94	0.62	0.54	**0.62**
IDS	AndMal17	ODIN	k = 5	0.796	0.181	0.958	0.32	0.30	0.51	**0.15**
IDS	CICIDS17	iForest	t = 10, s = 20	0.010	0.997	0.962	0.97	0.98	0.97	**0.91**
IDS	CICIDS18	HBOS	hist = 100	0.060	0.855	1.000	0.96	0.92	0.97	**0.90**
IDS	CIDDS	kNN / SVM	k = 5 / ker. RBF, nu = 0.02	0.041	0.802	0.997	0.96	0.89	0.95	**0.88**
IDS	CTU13	FastABOD	k = 5	0.269	0.012	0.793	0.73	0.02	0.06	**0.08**
IDS	ISCX12	iForest	t = 5, s = 50	0.010	0.987	0.974	0.98	0.98	0.98	**0.97**
IDS	Netflow-IDS	SVM	ker. RBF, nu = 0.1	0.015	0.876	0.863	0.97	0.87	0.87	**0.85**
IDS	NGDIS-DS	COF	k = 20	0.127	0.253	0.759	0.87	0.38	0.54	**0.39**
IDS	NSLKDD	SVM	ker. Cubic, nu = 0.02	0.055	0.890	0.642	0.82	0.75	0.68	**0.63**
IDS	UGR16	SDO	obs = 50, q = 0.1	0.033	0.290	0.398	0.95	0.34	0.37	**0.31**
IDS	UNSW-NB15	HBOS	hist = 50	0.006	0.807	0.340	0.95	0.48	0.38	**0.51**

5.1 Detection Capabilities of Unsupervised Algorithms

We begin the analysis with a discussion of the unsupervised algorithms that show the best MCC score for each dataset, commenting also on metric scores other than MCC. We show metric scores for each dataset in Figure 3 and Table 3: while Figure 3 shows bars for MCC scores as well as the algorithm(s) that reach that maximum score, Table 3 reports on FPR, P, R, ACC, F1, F2 and MCC achieved by the best algorithm on each dataset, alongside with their parameters values.

At a first glance, Figure 3 shows how the number of misclassifications – measured through MCC score – may vary a lot when considering different datasets. Even the best algorithm in AndMal17 and CTU13 datasets achieved very bad MCC scores, meaning that the number of misclassifications was really high and that unsupervised algorithms were not able to provide actionable support to intrusion detection. Instead, in 8 out of the 21 datasets, it was possible to find an algorithm that reached or exceeded an MCC of 0.8. While these scores are still far from the optimum (absolute MCC peaks at 1), they point to a reasonably low number of misclassifications, considering that such algorithms learn the normal behavior of the system without considering any label in training data. In particular, Isolation Forests in CICIDS17 and HBOS in CICIDS18 datasets reached an MCC of 0.9. With the aid of Table 3 (12^{th} and 13^{th} rows) we can see how these MCC values correspond to Accuracy values of 0.97 and 0.96, meaning that respectively 3% and 4% of data points are being misclassified by the two algorithms above. For these two datasets, it is also interesting to note how those misclassifications are distributed: HBOS on CICIDS18 dataset shows only FPs (see perfect Recall = 1 in Table 3), while the vast majority of misclassifications of Isolation Forests in CICIDS17 are FNs: Precision and FPR values are very low and close to the optimum (1 for Precision, 0 for FPR).

A similar analysis can be conducted for all the datasets considered in this study. However we want to highlight here the following general observations.

- *It is not possible to find an algorithm that always classifies correctly each data point for any of the 21 datasets.* In fact, the MCC score of the best algorithm never reaches the desirable 1 value. As a result, for each dataset there is room for improvement as no perfect intrusion detection strategy was found by using unsupervised algorithms.
- Only SVM on Human Gait and DBSCAN on Fingerprint datasets do not raise FPs (see Precision = 1 and FPR = 0 in Table 3), while zero FNs were achieved only by HBOS in the CICIDS18 dataset. It is possible to conclude that unsupervised algorithms do not lean towards minimizing either FPs or FNs. However, it is possible to note that *for datasets in the domain of biometric authentication FNs are usually higher than FPs* (Precision is more frequently higher than Recall in Table 3 for *Bio* datasets), while for network IDs datasets the trend is swapped, with higher Recall values than Precision counterparts.
- *There is no algorithm that is the optimal choice in the majority of the datasets*, and that could have been chosen as the best candidate when performing unsupervised intrusion detection. Some algorithms as FastABOD and SVM appear quite frequently (respectively in 24% and 19% of the datasets), but there is no clear evidence of algorithms being the preferred choice for a relevant number of datasets. Instead, we can observe how LDCOF, LOF, K-Means and G-Means never appear as preferred choices. Lastly, it is worth noticing that for some datasets there is more than one algorithm that allows reaching the best MCC. This may happen as some algorithms produce the same number of misclassifications even if they rely on different actions to decide on anomalies.

The three aspects above highlight how unsupervised algorithms – as expected – are able to derive a normal behavior from an unlabelled dataset at a cost of a quite high number of misclassifications. Therefore, we continue our analysis to see if meta-learners can help in reducing their gap with respect to perfect (i.e., no misclassifications at all) detection capabilities.

5.2 On the Choice of Base-Learners in MC(O) Meta-Learners

Before starting the discussion on the potential improvements of meta-learners with respect to unsupervised algorithms, we debate here on the impact of the choice of base-learners for MC and MCO meta-learners. As already described in Section 4.4, within our experimental campaign

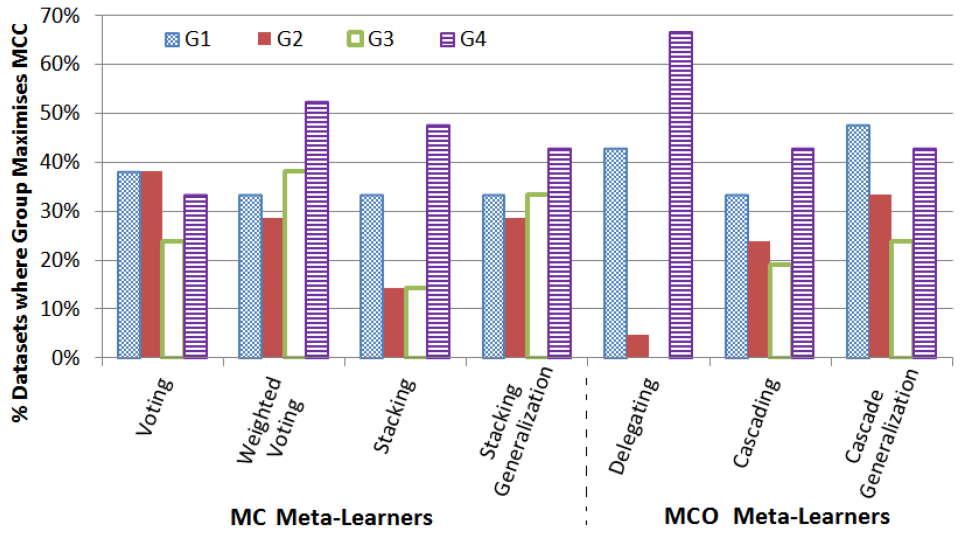

Fig. 4. Impact on the choice of groups G1-G4 on detection capabilities of algorithms. Bars are higher the higher the % of datasets in which a given group maximises the MCC achieved by a meta-learner.

we repeated the experiments involving MC and MCO meta-learners by using G1 – G4 groups of base-learners that were selected according to different rules.

- G1. Three Algorithms that resulted in a higher average MCC in Table 3: SVM (Classification family), ODIN (Neighbor-based), FastABOD (Angle/Neighbor-based).
- G2. Three Algorithms that resulted in a higher average Accuracy in our experiments: SDO (Density family), DBSCAN (Clustering), SVM (Classification).
- G3. Three Algorithms that resulted in a higher average Recall in our experiments: G-Means (Clustering family), FastABOD (Angle/Neighbor-based), COF (Density/Neighbor-based).
- G4. Seven Algorithms, one for each family: SVM (Classification family), ODIN (Neighbor-based), FastABOD (Angle/Neighbor-based), SDO (Density), DBSCAN (Clustering), SOM (Neural Networks), HBOS (Statistical). When more classifiers for a given family were available, we chose the ones that had fewer semantic overlaps with other families. For example, we selected SDO as density-based representative instead of LOF or COF, which pair a nearest-neighbor search with density measures.

We expect a different number of misclassifications by meta-learners when G1-G4 groups are alternatively used to build base-level learners. To show the differences, Figure 4 reports a bar chart with 4 data series (one for each group G1 – G4) and seven partitions, one for each MC(O) meta-learner. The higher the bars, the higher the number of datasets in which that group let a specific meta-learner result in the highest MCC out of the four. Note that the sum of a set of 4 bars for a meta-learner usually sums up to more than 100% as it happens that more than one group maximizes the MCC reached by a meta-learner on a dataset; they result in the same score, and are both considered as groups that maximize MCC.

The graph shows quite clearly that G3 (GMeans, FastABOD, COF) did not help as much as other groups in raising metric scores achieved by meta-learners. FastABOD and COF partially belong to the neighbor-based family: as a result, they may incur in the same misclassifications once the intrusion is not detected by their engine, which relies on Euclidean distance from neighbors. The same goes for G1, which includes two neighbor-based algorithms; however, G1 leads to

higher MCC than G3 for all meta-learners but weighted voting. In fact, pattern-filled blue bars in Figure 4 are usually higher than green-bordered bars that represent G3. *It turns out that looking only at different families to select base-learners is not enough.* Instead, it is more likely that SVM (G1) has a better synergy with both or either ODIN and FastABOD than what GMeans (G3) has with both COF and FastABOD. As an additional remark, we observe how G2 algorithms belong to completely separate families with no overlaps, but indeed this group is never the preferred choice over G1.

The last observation is dedicated to G4, which is the only group that includes seven algorithms belonging to different families. It is worth mentioning that this group allowed reaching the highest MCC scores (or highest %, as reported in Figure 4 – see purple-striped bars) for five out of the seven MC(O) meta-learners. Therefore, in the next section, we will consider results obtained by using group G4 as base-learners for MC(O) meta-learners, as they seem to benefit the most from this group of base-learners.

Wrapping up this analysis, we can conclude that - as expected - the choice of base-learners impacts the detection capabilities of MC(O) meta-learners. However, our results show that selecting classifiers belonging to different families does not guarantee that the resulting set will provide base-level learners with good synergy. Indeed, we foresee that *selecting classifiers belonging the same family leads to less-diverse base learners*, which does not help the meta-learner in reducing misclassifications. However, to derive conclusive results, we will prepare a separate work that we will be describing at the end of the paper as a possible follow-up of this paper.

5.3 Comparing Meta-Learners and Unsupervised Algorithms

To complete our analysis, in Table 4 we report the metric scores achieved by both unsupervised (non-meta) algorithms and meta learners that reach the highest MCC on each dataset. This allows understanding if adopting meta-learners may help in reducing misclassifications and consequently improving metric scores. The table reports on the same metrics as Table 3: instead, here we report two rows for each dataset, with metric scores related to unsupervised (non-meta) algorithms, and meta-learners. Albeit data for non-meta algorithms is repeated from Table 3, we duplicated them for ease of comparison with meta-learners.

For each dataset, we can observe that *the MCC are obtained by the optimal meta-learner(s) is higher than the MCC of the optimal unsupervised algorithm for all datasets but AndMal17*, which still represents a challenging situation where neither unsupervised algorithms nor meta-learners were able to satisfactorily detect anomalies due to intrusions. More in detail, meta-learners allow perfect classification in the Fingerprint dataset, with almost perfect MCC of 0.999 also for the CICIDS18 dataset. These improvements are remarkable as they significantly reduce – if not avoid all – misclassifications in these two datasets, providing a solid answer to the detection of intrusions that rely on unsupervised base-learners.

In addition, more than one meta-learner reaches perfect classification scores in the Fingerprint dataset, namely Cascading, and Boosting with the following setups: (i) kNN algorithm and 10 weak learners, (ii) ODIN with 10, (iii) SDO with 10 and (iv) K-Means with 10. While the tie between kNN and ODIN is somewhat expected as ODIN builds upon kNN, it is surprising to observe how building Boosting with algorithms belonging to different families as kNN (neighbor), SDO (density) and K-Means (clustering) allows reaching the same –perfect – scores.

Moreover, Table 4 allows elaborating on FPs and FNs, which are indirectly measured through FPR, P, and R (but still directly available at [82], though not shown here for brevity). *Zero FPs were achieved by meta-learners in 9 out of the 21 datasets, while only in Fingerprint and Human Gait datasets meta-learners were able to completely avoid FNs* **.** This marks a significant improvement with respect to unsupervised algorithms, which we can expand in detail through Table 5. This last

Table 4. Non-meta and Meta Learners that Reach the Maximum MCC for Each Dataset

Dataset	Meta / Unsupervised (non-meta) Algorithm(s)	FPR	P	R	ACC	F1	F2	MCC
EDA(SWELL)	SDO	0.009	0.827	0.401	0.93	0.54	0.45	**0.547**
EDA(SWELL)	Boosting(FastABOD, 20)	0.000	1.000	0.569	0.96	0.73	0.62	**0.737**
EDA(WESAD)	ODIN	0.014	0.833	0.833	0.97	0.83	0.83	**0.819**
EDA(WESAD)	Voting, Weighted Voting, Bagging(FastABOD, 10)	0.000	1.000	0.833	0.99	0.91	0.86	**0.907**
Face	DBSCAN	0.004	0.765	0.149	0.92	0.25	0.18	**0.316**
Face	Stacking, Stacking Generalization	0.000	1.000	0.218	0.93	0.36	0.26	**0.451**
Fingerprint	DBSCAN	0.000	1.000	0.508	0.99	0.67	0.56	**0.709**
Fingerprint	Boosting*, Cascading	0.000	1.000	1.000	1.00	1.00	1.00	**1.000**
Hand_Gesture	FastABOD	0.004	0.917	0.478	0.95	0.63	0.53	**0.643**
Hand_Gesture	Boosting(COF, 20)	0.008	0.857	0.522	0.95	0.65	0.57	**0.647**
HRV(SWELL)	FastABOD	0.004	0.946	0.700	0.97	0.80	0.74	**0.797**
HRV(SWELL)	Voting, Weighted Voting	0.000	1.000	0.670	0.97	0.80	0.72	**0.804**
HRV(WESAD)	ODIN	0.004	0.938	0.606	0.96	0.74	0.65	**0.734**
HRV(WESAD)	Boosting(COF, 50)	0.000	1.000	0.606	0.96	0.75	0.66	**0.762**
Human Gait	SVM	0.000	1.000	0.756	0.99	0.86	0.79	**0.867**
Human Gait	Boosting(FastABOD, 10)	0.003	0.882	1.000	1.00	0.94	0.97	**0.938**
Keystroke	HBOS	0.007	0.786	0.232	0.92	0.36	0.27	**0.400**
Keystroke	Bagging(HBOS), Boosting*, Cascading, Cascade Generalization	0.000	1.000	0.495	0.95	0.66	0.55	**0.685**
Voice	Isolation Forest	0.011	0.839	0.495	0.94	0.62	0.54	**0.616**
Voice	Cascade Generalization	0.017	0.816	0.653	0.95	0.73	0.68	**0.703**
AndMal17	ODIN	0.796	0.181	0.958	0.32	0.30	0.51	**0.154**
AndMal17	Bagging(SDO, 20)	0.126	0.236	0.213	0.77	0.22	0.22	**0.091**
CICIDS17	Isolation Forest	0.010	0.997	0.962	0.97	0.98	0.97	**0.908**
CICIDS17	Bagging(COF, 10)	0.015	0.996	0.974	0.98	0.98	0.98	**0.930**
CICIDS18	HBOS	0.060	0.855	1.000	0.96	0.92	0.97	**0.896**
CICIDS18	Boosting(KMeans, 10)	0.000	1.000	0.999	1.00	1.00	1.00	**0.999**
CIDDS	kNN, SVM	0.041	0.802	0.997	0.96	0.89	0.95	**0.875**
CIDDS	Bagging(DBSCAN, 50)	0.041	0.803	0.997	0.96	0.89	0.95	**0.876**
CTU13	FASTABOD	0.269	0.012	0.793	0.73	0.02	0.06	**0.076**
CTU13	Boosting(LOF, 10)	0.000	0.500	0.034	1.00	0.06	0.04	**0.131**
ISCX12	Isolation Forest	0.010	0.987	0.974	0.98	0.98	0.98	**0.966**
ISCX12	Boosting*, Cascading, Cascade Generalization	0.000	1.000	0.974	0.99	0.99	0.98	**0.977**
Netflow-IDS	SVM	0.015	0.876	0.863	0.97	0.87	0.87	**0.853**
Netflow-IDS	Stacking	0.018	0.869	0.939	0.98	0.90	0.92	**0.890**
NGDIS-DS	COF	0.127	0.253	0.759	0.87	0.38	0.54	**0.387**
NGDIS-DS	Boosting(ODIN, 10)	0.002	0.948	0.729	0.98	0.82	0.76	**0.824**
NSLKDD	SVM	0.055	0.890	0.642	0.82	0.75	0.68	**0.633**
NSLKDD	Cascading(thr=0.99)	0.040	0.915	0.627	0.82	0.74	0.67	**0.643**
UGR16	SDO	0.033	0.290	0.398	0.95	0.34	0.37	**0.314**
UGR16	Bagging*, Boosting*, Cascading, Cascade Generalization	0.001	0.956	0.377	0.98	0.54	0.43	**0.593**
UNSW-NB15	HBOS	0.006	0.807	0.340	0.95	0.48	0.38	**0.505**
UNSW-NB15	Cascade Generalization	0.005	0.856	0.421	0.96	0.56	0.47	**0.583**

We report the dataset name, the classifier, and metrics as FPR, Precision, Recall, Accuracy, F-Measure(F1), F2, and MCC. * next to meta-learners indicates that different parameter combinations of this meta-learner allow to reach the same score.

Table 5. Differences of MCC Achieved By Each Meta-learner on Each Dataset, with Differences with Respect to the MCC Achieved by Unsupervised (Non-meta) Algorithms

Domain	Dataset	MCC Unsupervised Algorithm	Bagging	Boosting	Voting	Weighted Voting	Stacking	Stacking Generalization	Delegating	Cascading	Cascade Generalization	#Meta Better Than Unsupervised
Bio	EDA(SWELL)	0.55	0.18	**0.19**	0.11	0.14	0.03	0.03		0.12	0.11	8
Bio	EDA(WESAD)	0.82	**0.09**		**0.09**	**0.09**				0.01	0.07	5
Bio	Face	0.32					**0.13**	**0.13**			0.03	3
Bio	Fingerprint	0.71		**0.29**						**0.29**	0.23	3
Bio	Hand Gesture	0.64		**0.003**								1
Bio	HRV(SWELL)	0.80			**0.01**	**0.01**						2
Bio	HRV(WESAD)	0.73		**0.03**					0.02	0.02		3
Bio	Human Gait	0.88		**0.06**								1
Bio	Keystroke	0.40	**0.29**	**0.29**	0.25	0.25			0.01	**0.29**	**0.29**	7
Bio	Voice	0.62		0.04			0.04	0.04		0.04	**0.09**	5
IDS	AndMal17	0.15										0
IDS	CICIDS17	0.91	**0.02**	0.01								2
IDS	CICIDS18	0.90	0.08	**0.10**		0.08		0.07		0.09	0.09	6
IDS	CIDDS	0.88	**0.002**									1
IDS	CTU13	0.08		**0.05**					0.01	0.03		3
IDS	ISCX12	0.97	0.001	**0.01**						**0.01**	**0.01**	4
IDS	Netflow_IDS	0.85	0.01				**0.04**					2
IDS	NGDIS-DS	0.39	0.19	**0.44**	0.15	0.24			0.12	0.13	0.15	7
IDS	NSLKDD	0.63				0.003			0.002	**0.01**		3
IDS	UGR16	0.31	**0.28**	**0.28**	0.27	0.27				**0.28**	**0.28**	6
IDS	UNSW-NB15	0.51	0.04	0.07	0.02				0.01		**0.08**	5
Times Meta Better Than Unsupervised			**11**	**14**	**7**	**8**	**4**	**4**	**6**	**12**	**11**	
Times Meta Better Overall			**5**	**11**	**2**	**2**	**2**	**1**	**0**	**5**	**5**	

Blank cells show combinations where a meta-learner did not improve scores. Bold underlined cells highlight optimal classifier(s) for each dataset.

table shows – for each dataset – the absolute improvements of MCC in adopting a given meta-learner with respect to unsupervised algorithms. The centre of the table reports either (i) blank cells, meaning that the meta-learner did not improve MCC with respect to the most performing unsupervised algorithm in that dataset, (ii) a positive number which scores the improvement in adopting meta-learning on that dataset, or (iii) a positive bold-font number, which points to the highest MCC obtained for that dataset amongst all the (meta)algorithms we exercised in our experimental campaign.

On the bottom of the table, we also report on the number of times in which a given meta-learner reaches higher MCC than unsupervised algorithms (intuitively, the number of non-blank cells in the column), and the number of times in which the meta-learner is the preferred choice amongst all the 21 datasets considered in this study.

This last datum shows how Boosting outperforms base algorithms and other meta-learners in 11 out of the 21 datasets. In other words, we can conclude that *in the 52% of the cases, adopting Boosting allows reaching the highest MCC scores*, consequently minimizing misclassifications. Other meta-learners are not even close to these numbers, considering that Bagging, Cascading and Cascade Generalization, which are still viable choices, are preferred only in fewer than half of Boosting's cases. Note that the numbers in the last row of Table 5 sum up more than 21 as when more meta-learners show the same (highest) MCC, they are counted twice to achieve that number.

Regarding relative improvements in the second last row of Table 5, Boosting outperforms unsupervised algorithms in 14 out of 21 datasets, closely followed by both Cascading (12), Bagging (11) and Cascade Generalization (11 as well). Other meta-learners may be preferred in specific cases, but do not show noticeable overall capabilities being only slightly better than unsupervised algorithms.

The last – but not least – comment relates to the absolute improvement of MCC that follows the adoption of meta-learners. For datasets such as HRV(SWELL) and Hand Gesture, Table 5 shows a very limited improvement of MCC (0.80 to 0.81, 0.640 to 0.643) that may not be enough to motivate the adoption of more complex techniques such as meta-learners as they only marginally reduce misclassifications. However, small variations as the 0.01 for ISCX12 raise an already high MCC of 0.97 to 0.98, which is more valuable than the same improvement starting from lower MCC scores (e.g., from 0.48 to 0.49) and may sufficiently motivate the adoption of meta-learning.

6 CONCLUDING REMARKS

To conclude the paper we report here on (i) potential limits to the applicability and validity of our results, a (ii) our main findings, and (iii) future implications and follow-ups of this study.

6.1 Limits to the Applicability and Validity of our Results

We report here possible limitations to the validity and the applicability of our study. These are not to be intended as showstoppers when interpreting the conclusions of this paper. Instead, they should be considered as boundaries or implications which may limit the validity of this study in specific scenarios.

Parameters' Tuning. Finding the optimal values of algorithms' parameters is a substantial process that requires sensitive analyses and is directly linked with the target system. When using many datasets and many algorithms, it is unfeasible to conduct sensitivity analyses directed to find the optimal combination of parameters e.g., the k in kNN that maximizes a given metric for a given dataset. To automatize this process, RELOAD and similar tools enable grid searches, which try different parameters combinations and choose the one that seems beneficial for the algorithm according to a quick test executed on a part of the training set which is indeed labelled. Here labels are not used for training, but only to select parameters. However, the optimal combination of parameters for a given algorithm in a given system may not be included in those grid searches and therefore algorithms will not perform optimally.

Usage of Public Data/Tools. The heterogeneity of data sources, their potential lack of documentation and the different strategies authors used to collect data may limit the understandability of data. In addition, such datasets are not under our control: therefore, possible actions as changing the way data is generated by considering more features to improve detection scores are out of consideration e.g., in NGDIS dataset, we are forced to stick with just three (see Table 2) ordinal numeric features. Indeed, the usage of public data and public tools allows reproducibility and guarantees relying on proven-in-use data.

Comparison with Supervised Variants. The paper explicitly focuses on unsupervised algorithms, as the detection of zero-day attacks (or slightly different variants of existing attacks) is a

very important and challenging feature which is getting more and more desired in current systems which are complex, dynamic, and evolve rapidly. However, unsupervised algorithms usually show weaker detection capabilities with respect to supervised algorithms in detecting known attacks. As a result, supervised algorithms - even meta-learners e.g., ADABoost [83], which builds boosting on a (supervised) decision tree, or Random Forests (Bagging with, again, decision trees [14]) – *should be preferred when dealing with known attacks, but should indeed be used together with unsupervised (meta)classifiers* that cover against unexpected and unplanned zero-day attacks.

6.2 Lessons Learned

Before discussing possible follow ups, we summarize the main findings of this paper below.

- Analysis of unsupervised algorithms showed that it was not possible to find an algorithm that clearly outperforms others in the 21 datasets considered in this study. This would have been a huge result, but was indeed unexpected as datasets have different characteristics and different algorithms may better suit specific datasets.
- Instead, in Section 5.2 we started focusing on MC(O) meta-learners by running experiments with different groups of base-learners. We found that selecting classifiers belonging to different families does not guarantee that the resulting set will describe base-level classifiers with good synergy. While groups of base-learners with similar characteristics showed poor results, not all the groups with "diverse" algorithms as base-learners showed promising results.
- These analyses escalated in Section 5.3 into a deep comparison of metric scores achieved by unsupervised algorithms with respect to metric scores obtained by meta-learners. *The MCC score obtained by meta-learners is higher than the MCC of unsupervised algorithms for 20/21 datasets*, meaning that meta-learners guarantee less misclassification in almost all cases – and no misclassification at all for one dataset. Boosting outperforms unsupervised algorithms and other meta-learners in 11/21 datasets, consequently minimizing misclassifications.

6.3 Future Works

A first follow up of this work may consist in estimating if and how the increased complexity of meta-learners burdens hardware-software systems, potentially exhausting their resources and preventing a timely detection of malicious activities.

Moreover, to adequately evaluate the detection capabilities of MC and MCO meta-learners there is the need to conduct a separate and very extensive studies about the combinations (for MC) or permutations (for MCO, where ordering matters) of unsupervised algorithms used as base-learners that maximizes metric scores across all datasets, expanding the preliminary analysis presented in Section 5.2.

References

[1] V. Chandola, A. Banerjee, V. Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)* 41, 3 (2009), 15.

[2] M. Goldstein and S. Uchida. 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one* 11, 4 (2016), e0152173.

[3] K. Leung and C. Leckie. 2005. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume* 38. Australian Computer Society, Inc., 333–342.

[4] T. Zoppi, A. Ceccarelli, T. Capecchi, and A. Bondavalli. 2021. Unsupervised anomaly detectors to detect intrusions in the current threat landscape. *ACM/IMS Transactions on Data Science* 2, 2 (2021), 1–26.

[5] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. 2003. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 25–36.

[6] L. D'hooge, T. Wauters, B. Volckaert, and F. De Turck. 2019. In-depth comparative evaluation of supervised machine learning approaches for detection of cybersecurity threats. In *Proc. 4th Int. Conf. Internet Things, Big Data Secur. (IoTBDS)* 1, (2019), 125–136.

[7] P. S. Kenkre, A. Pai, and L. Colaco. 2015. Real time intrusion detection and prevention system. In *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)* 2014. Springer, Cham, 405–411.

[8] C. Kruegel and T. Toth. 2003. Using decision trees to improve signature-based intrusion detection. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, Berlin, 173–191.

[9] T. Zoppi, A. Ceccarelli, and A. Bondavalli. 2016. Context-awareness to improve anomaly detection in dynamic service oriented architectures. In *International Conference on Computer Safety, Reliability, and Security*. Springer, Cham, 145–158.

[10] L. Bilge and T. Dumitraş. 2012. Before we knew it: An empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM, 833–844.

[11] Casas Pedro, Johan Mazel, and Philippe Owezarski. 2012. Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. *Computer Communications* 35, 7 (2012), 772–783.

[12] Committee on National Security Systems - CNSSI No. 4009 "Committee on National Security Systems (CNSS) Glossary", April 2015.

[13] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 1, 1 (2004), 11–33.

[14] L. Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32.

[15] Y. Freund and R. E. Schapire. 1996. Experiments with a new boosting algorithm. *In icml* 96, (1996), 148–156.

[16] L. Breiman. 1996. Bagging predictors. *Machine Learning* 26, 2 (1996), 123–140.

[17] R. E. Schapire. 1990. The strength of weak learnability. *Machine Learning* 5, (1990), 197–227. https://doi.org/10.1007/BF00116037

[18] Wolpert David. 1992. Stacked generalization. *Neural Networks* 5 (1992), 241–259. 10.1016/S0893-6080(05)80023-1

[19] E. Alpaydin and C. Kaynak. 1998. Cascading classifiers. *Kybernetika* 34 (1998), 369–374.

[20] J. Gama and Brazdil P. Cascade generalization. *Machine Learning* 41, 3 (2000), 315–343.

[21] C. Ferri, P. Flach, and J. Hernandez-Orallo. 2004. Delegating classifiers. In *Proceedings of the Twenty-first International Conference on Machine Learning, (ICML'04)*, 289–296.

[22] Ortega Julio, Koppel Moshe, and Argamon Shlomo. 2001. Arbitrating among competing classifiers using learned referees. *Knowledge and Information Systems* 3 (2001), 470–490. 10.1007/PL00011679

[23] P. Chan and S. Stolfo. 1993. Toward parallel and distributed learning by metalearning. In *Working Notes of the AAAI-93 Workshop on Knowledge Discovery in Databases*. 227–240.

[24] Lemke Christiane, Budka Marcin, and Gabrys Bogdan. 2013. Metalearning: A survey of trends and technologies. *Artificial Intelligence Review*. DOI: https://doi.org/10.1007/s10462-013-9406-y

[25] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. 2009. *Metalearning: Applications to data mining*. Springer, Berlin.

[26] J. Vanschoren. 2010. *Understanding machine learning performance with experiment databases. PhD thesis*, Arenberg Doctoral School of Science, Engineering & Technology, Katholieke Universiteit Leuven.

[27] M. Gharib and A. Bondavalli. 2019. On the evaluation measures for machine learning algorithms for safety-critical systems. In *2019 15th European Dependable Computing Conference (EDCC)*. IEEE, 141–144.

[28] Check Point Research. 2019. Cyber Attack Trend: 2019 Mid-Year Report, vol. 1, 2019.

[29] ENISA. 2018. Threat Landscape Report 7, 2018.

[30] Chen Liming and Algirdas Avizienis. 1978. N-version programming: A fault-tolerance approach to reliability of software operation. *Proc. 8th IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-8)*. 1. 1978.

[31] Nordmann Lars and Hoang Pham. 2018. Weighted voting systems. *IEEE Transactions on Reliability* 48, 1 (1999), 42–49.

[32] ENISA. 2018. Threat landscape report 7, 2018.

[33] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho. 2019. A survey of network-based intrusion detection data sets. *Computers & Security*.

[34] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security* 31, 3 (2012), 357–374.

[35] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009*. IEEE Symposium on. IEEE, 1–6.

[36] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho. 2017. Flow-based benchmark data sets for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security. ACPI*, 361–369.
[37] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *Military Communications and Information Systems Conference (MilCIS) 2015*. IEEE, 1–6.
[38] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, 108–116.
[39] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie. 2017. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications* 87 (2017), 185–192.
[40] S. Garcia, M. Grill, J. Stiborek, and A. Zunino. 2014. An empirical comparison of botnet detection methods. *Computers & Security* 45, (2014), 100–123.
[41] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani. 2018. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In *2018 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 1–7.
[42] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Theron. 2018. UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers & Security* 73 (2018), 411–424.
[43] BIT – Biometrics Ideal Test, CASIA-FingerprintV5. Retrieved on 15 December, 2020 from http://biometrics.idealtest.org/
[44] MathWorks - FingerPrint Matching: A simple approach, https://it.mathworks.com/matlabcentral/fileexchange/44369-fingerprint-matching-a-simple-approach (online), accessed: 2019-11-20
[45] Warwick R. Adams. 2017. High-accuracy detection of early parkinson's disease using multiple characteristics of finger movement while typing. *PloS one* 12, 11 (2017), e0188226.
[46] S. Koldijk, M. Sappelli, S. Verberne, M. Neerincx, and W. Kraaij. 2014. The SWELL knowledge work dataset for stress and user modeling research. *To appear in: Proceedings of the 16th ACM International Conference on Multimodal Interaction (ICMI 2014)* (Istanbul, Turkey, 12–16 November 2014)
[47] Philip Schmidt, Attila Reiss, Robert Duerichen, Claus Marberger, Kristof Van Laerhoven. 2018. Introducing WESAD, a multimodal dataset for wearable stress and affect detection. *ICMI 2018*, Boulder, USA, 2018
[48] A. Memo, L. Minto, and P. Zanuttigh. 2015. *Exploiting Silhouette Descriptors and Synthetic Data for Hand Gesture Recognition*. STAG: Smart Tools & Apps for Graphics, 2015
[49] A. Vajdi, M. R. Zaghian, S. Farahmand, E. Rastegar, K. Maroofi, S. Jia, and A. Bayat. 2019. Human gait database for normal walk collected by smart phone accelerometer. arXiv preprint arXiv:1905.03109.
[50] Kaggle - Voice Recognition, Jeganathan Kolappan. https://www.kaggle.com/jeganathan/voice-recognition (online), accessed: 2019-11-20
[51] Kaggle - Face Images with Marked Landmark Points, Omri Goldstein. https://www.kaggle.com/drgilermo/face-images-with-marked-landmark-points (online), accessed: 2019-11-20
[52] National Science Foundation, "Cyber-Physical Systems (CPS) - nsf20563", April 2020
[53] Y. Wang, Y. Shen, and G. Zhang. 2016. Research on intrusion detection model using ensemble learning methods. In *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 422–425.
[54] B. A. Tama, M. Comuzzi, and K. H. Rhee. 2019. TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE Access* 7 (2019), 94497–94507.
[55] O. Oriola. 2020. A stacked generalization ensemble approach for improved intrusion detection. *International Journal of Computer Science and Information Security (IJCSIS)* 18 (2020), 5.
[56] I. P. Possebon, A. S. Silva, L. Z. Granville, A. Schaeffer-Filho, and A. Marnerides. 2019. Improved network traffic classification using ensemble learning. In *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 1–6.
[57] A. Alaba, S. Maitanmi, and O. Ajayi. 2019. An ensemble of classification techniques for intrusion detection systems. *International Journal of Computer Science and Information Security (IJCSIS)* 17, 11 (2019)
[58] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha. 2020. A stacking ensemble for network intrusion detection using heterogeneous datasets. *Security and Communication Networks*, 2020.
[59] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenko-va, E. Schubert, I. Assent, and M. E. Houle. 2016. On the evaluation of outlier detection: Measures, datasets, and an empirical study. In *Lernen, Wissen, Daten, Analysen 2016*. CEUR work-shop proceedings, 2016.
[60] D. M. Powers. 2011. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, 2011
[61] D. Chicco and G. Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* 21, 1 (2020), 6.

[62] T. Zoppi, A. Ceccarelli, and A. Bondavalli. 2019. Evaluation of anomaly detection algorithms made easy with RELOAD. In *Proceedings of the 30th Int. Symposium on Soft-ware Reliability Engineering (ISSRE)*. IEEE, 446–455. DOI: https://doi.org/10.1109/ISSRE.2019.00051
[63] B. Azhagusundari and Antony Selvadoss Thanamani. 2013. Feature selection based on information gain. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* 2, 2 (2013), 18–21.
[64] Goldstein Markus and Andreas Dengel. 2012. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: Poster /Demo Track* (2012), 59–63.
[65] H-P. Kriegel and A. Zimek. Angle-based outlier detection in high-dimensional data. *Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery Data Mining; '08*. 444–452.
[66] V. Hautamaki, I. Karkkainen, and P. Franti. 2004. Outlier detection using k-nearest neighbour graph. *Pattern Recognition. ICPR 2004. Proceedings of the 17th Int. Conference on* Vol. 3. IEEE, 430-433.
[67] M. Amer, M. Goldstein, and S. Abdennadher. 2013. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*. ACM, 2013, 8–15.
[68] Vázquez Félix Iglesias, Tanja Zseby, and Arthur Zimek. 2018. Outlier detection based on low density models. *2018 IEEE Int. Conference on Data Mining Workshops (ICDMW)*. IEEE, 2018.
[69] T. Kohonen. 1997. Exploration of very large databases by self-organizing maps. In *Proc. of Int. Conference on Neural Networks (ICNN'97)*, Vol. 1. IEEE, PL1–PL6.
[70] G. Hamerly and C. Elkan. 2004. Learning the k in k-means. In *Advances in Neural Information Processing Systems*. 281–288.
[71] Mennatallah Amer and Markus Goldstein. 2012. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proceedings of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012)*.
[72] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. 2002. Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 535–548.
[73] Martin Ester, Han-peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *2nd Int. Conference on Knowledge Discovery and Data Mining (KDD-96)*.
[74] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander. 2000. LOF: Identifying density-based local outliers. In *ACM Sigmod Record*, Vol. 29. ACM, 93–104.
[75] M. Radovanović, A. Nanopoulos, and M. Ivanović. 2014. Reverse nearest neighbors in unsupervised distance-based outlier detection. *IEEE Transactions on Knowledge and Data Engineering* 27, 5 (2014), 1369–1382.
[76] F. T. Liu, K. M. Ting, and Z. H. Zhou. 2008. Isolation forest. In *2008 8th IEEE Int. Conference on Data Mining*. IEEE, 413–422.
[77] J. H. M. Janssens, F. Huszar, E. O. Postma, and H. J. van den Herik. 2012. Stochastic outlier selection. Technical Report TiCC TR 2012-001, Tilburg University, Tilburg Center for Cognition and Communication, Tilburg, The Netherlands.
[78] J. A. Hartigan and M. A. Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society*. Series C, 28, 1 (1979), 100–108.
[79] Stan Z. Li. 2009. *Encyclopedia of biometrics: I-Z*. Vol. 2. Springer Science & Business Media, 2009.
[80] Roberts Chris. 2007. Biometric attack vectors and defences. *Computers & Security* 26, 1 (2007), 14–25.
[81] W. Dahea and H. S. Fadewar. 2018. Multimodal biometric system: A review. *International Journal of Research in Advanced Engineering and Technology* 4, 1 (2018), 25–31.
[82] Archive of full metric scores (online). Retrieved on 15 December, 2020 from https://drive.google.com/file/d/1D3M9tLxtG9yvG689LbAONykjBz3XXHJ8/view?usp=sharing
[83] Robert E. Schapire. 2013. *Explaining adaboost. Empirical inference*. Springer, Berlin, 37–52.
[84] T. Zoppi, A. Ceccarelli, L. Salani, and A. Bondavalli. 2020. On the educated selection of unsupervised algorithms via attacks and anomaly classes. *Journal of Information Security and Applications* 52, 102474.
[85] Hindy Hanan et al. 2018. A taxonomy and survey of intrusion detection system design techniques, network threats and datasets. arXiv preprint arXiv:1806.03517 (2018).
[86] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman. 2019. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* 2, 1 (2019), 20.
[87] S. Boughorbel, F. Jarray, and M. El-Anbari. 2017. Optimal classifier for imbalanced data using Matthews correlation coefficient metric. *PloS One* 12, 6 (2017), e0177678.
[88] Q. Zhu. 2020. On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset. *Pattern Recognition Letters*.
[89] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas. 2020. IoT type-of-traffic forecasting method based on gradient boosting neural networks. *Future Generation Computer Systems* 105, 331–345.
[90] A. Bansal and S. Kaur. 2018. Extreme gradient boosting based tuning for classification in intrusion detection systems. In *International Conference on Advances in Computing and Data Sciences*. Springer, Singapore, 372–380.

[91] M. Z. Alom and T. M. Taha. 2017. Network intrusion detection for cyber security using unsupervised deep learning approaches. *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, Dayton, OH, 2017, 63–69. DOI: https://doi.org/10.1109/NAECON.2017.8268746

[92] Zavrak Sultan and Murat İskefiyeli. 2020. Anomaly-based intrusion detection from network flow features using variational autoencoder. *IEEE Access* 8 (2020), 108346–108358.

[93] Y. Zhang, P. Li, and X. Wang. 2019. Intrusion detection for IoT based on improved genetic algorithm and deep belief network. *IEEE Access* 7, 31711–31722.

[94] L. Torrey and J. Shavlik. 2010. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI global, 242–264.

Received August 2020; revised December 2020; accepted May 2021