A Metadata-Assisted Cascading Ensemble Classification Framework for Automatic Annotation of Open IoT Data

# A Metadata-Assisted Cascading Ensemble Classification Framework for Automatic Annotation of Open IoT Data

Federico Montori*, Kewen Liao†‡, Matteo De Giosa§, Prem Prakash Jayaraman‡,
Luciano Bononi*, Timos Sellis¶, Dimitrios Georgakopoulos‡

*University of Bologna, Italy
†Australian Catholic University, Australia
‡Swinburne University of Technology, Australia
§University of Milano Bicocca, Italy
¶ Athena Research and Innovation Center, Greece

Corresponding author's E-mail: federico.montori2@unibo.it

***Abstract*—Public Internet of Things (IoT) platforms, such as Thingspeak, significantly increased the availability of open IoT data and enabled faster and cheaper development of novel IoT applications by reducing or even eliminating the need for deploying their own IoT sensors and platforms. However, open IoT data is often heterogeneous, sparse, fuzzy, and lacks accurate description (which we refer to as IoT metadata). These limitations make open IoT data challenging to integrate and use, and prevent the efficient development of IoT applications. In fact, while several sensor data description models have been proposed and standardized, open IoT data currently lack or include only partial metadata description. Therefore, novel techniques for automatically annotating open IoT data are needed to fully unleash the power of open IoT. This paper proposes a novel Metadata-Assisted Cascading Ensemble classification framework (MACE) for the automatic annotation of IoT data. MACE is capable of sequentially combining standalone classifiers, enabling it to cope with heterogeneous IoT data and different domains of information (*e.g.* numerical and textual), which have not been considered previously. MACE incorporates a novel ensemble approach for automatically selecting, sorting, filtering, and assembling classifiers in a way that improves annotation performance. The paper presents extensive experimental evaluations of MACE using public IoT datasets. Results demonstrate that the MACE framework significantly outperforms existing solutions for open IoT data by as much as 10% in classification accuracy.**

***Index Terms*—Internet of Things, Classification, Annotation, Open IoT Data, IoT Metadata, Sensors**

## I. Introduction

The connected future is set to be dominated by a significant growth of the heterogeneous Internet of Things (IoT) devices that are estimated to surpass the total number of mobile phones by 2022 [11]. This has led to a phenomenal increase in IoT data that is contributed by IoT devices across the globe. A significant subset of IoT data is available for use by any third-party IoT application via public IoT platforms [27] such as the Environmental Protection Agency (EPA) (`https://www.epa.gov/`) and ThingSpeak (`https://thingspeak.com/`).

In this paper, we refer to such IoT data as "open IoT data". Open IoT data offers enormous potential to address grand challenges that have been too hard to solve before, in domains such as Smart Cities, Healthcare and Environmental Monitoring, as well as to enhance the participation of citizens in traditional policymaking. Open IoT data dramatically improves the effectiveness of environmental and smart city-oriented solutions, by filling the gap of missing information or anticipating obstacles (stemming from increased urban migration, climate change, etc.) [27]. More specifically, various open IoT data projects are shaping smart cities of the future, by relying on both institutional and crowdsourced open IoT data. Examples include InfoAmazonia Colombia (`https://colombia.infoamazonia.org`), the Open Data Barometer (`https://opendatabarometer.org`), and BlindSquare (`https://www.blindsquare.com`) among many others. However, the heterogeneity and ambiguity of open IoT data significantly reduce the ability of IoT applications to integrate and use such data. Heterogeneity is due to the diversity in the type and the way IoT data observations are made by different IoT devices and platforms (*e.g.* different sensor type and accuracy, location discrepancy, non-alignment in time, etc.) while ambiguity is caused by the fact that open IoT data contributors use different annotation policies, if any, to annotate the data.

The Semantic Web 3.0 vision has fueled the development of several sensor data description models (*e.g.* Semantic Sensor network and SOSA), capturing the IoT sensor description in the form of metadata that includes an observation type (*e.g.* temperature, humidity), and a corresponding unit of measure (*e.g.* Celsius, Fahrenheit). However, most publicly available open IoT data lack such IoT sensor metadata and, often the type of the observation and its unit of measure produced by the IoT device is unclear [39].

This paper is a significant extension of the conference paper [28] appeared at WISE'18.

Federico Montori and Kewen Liao contributed equally to this work.

Such challenges demand the development of automatic classification techniques for annotating public open IoT data which makes the data usable by facilitating integration with the IoT application.

To address these open IoT data annotation challenges, this paper proposes Metadata-Assisted Cascading Ensemble (MACE) – a novel ensemble classification framework to tackle the challenge of annotating open IoT data. MACE incorporates techniques to tackle heterogeneity and ambiguity issues by classifying open IoT data that 1) lacks metadata description and 2) provides partial, incomplete, or inaccurate textual metadata (*e.g.*, an IoT device that produces temperature (IoT data) may be described/annotated using non-machine-interpretable names such as "temp", or "t1").

In our previous effort in addressing the problem of automatic annotation of open IoT data [28] we established via experimental evaluations that Time Series Classification (TSC) algorithms, which have been used extensively in the literature [2], do not perform well on open IoT data.

Thereby, we proposed a preliminary version of a sequential ensemble classification approach, namely Top-$k$ Sequential Ensemble (TKSE), which produced promising results. However, in [28], TKSE only supported ensembling two layers of classifiers, which were handpicked, *i.e.* there was no defined rationale for choosing efficiently classifiers nor how to sort them. In this paper, we build on our preliminary version of a sequential ensemble classification and we extend the concept into a more generic multi-layer classification framework, namely MACE, with significantly improved classification accuracy and strategies for multiple classifiers' selection, ordering, and filtering. Novel contributions of this paper include the following:

- A Metadata-Assisted Cascading Ensemble (MACE) classification framework for systematically and efficiently annotating public open IoT data. MACE is a significant improvement over TKSE [28]. In particular, we highlight (i) novel heuristics for selecting and ordering classifiers in MACE and (ii) novel strategies for filtering classes between the ensemble of classifiers in MACE.
- A new open IoT dataset for conducting experimental evaluations involving open IoT data classification. We produced this by extracting, cleaning, and manually annotating an existing dataset from ThingSpeak. We made this dataset available online to support further research on open IoT.
- An extensive experimental evaluation that illustrates the performance of MACE. In this evaluation, MACE achieved a classification accuracy between $84\%$ and $90\%$ which outperforms all existing classification techniques for open IoT data.

The rest of the paper is organized as follows: § II provides the background and the related work in the domain of IoT data classification and annotation. § III defines formally the problem and describes the proposed MACE framework with novel mechanisms for it to function effectively. § IV defines our experimental setup and introduces the datasets used throughout this study. § V details the experimental evaluations and § VI concludes the paper.

## II. Background and Related Work

IoT data is typically conceived as ordered sequences of sensor readings, which can be naturally seen as what in the literature are called "time series". Time Series Classification (TSC) problems, indeed, differ from ordinary classification problems in that data values are ordered (not necessarily in the dimension of time). Within the last years, several TSC approaches have been proposed [2] as alternatives to the One Nearest Neighbor (1NN) approach using pointwise Euclidean Distance as a similarity measure between series. The common agreement among researchers as a "hard to beat" de-facto standard distance measure between series is Dynamic Time Warping (DTW) [3], for which several approaches have been proposed in order to contrast its high time complexity [22]. The above mentioned are called *whole-series* methods, since they extract similarities by pointwise comparisons over the whole sequence. Other recent TSC approaches are *shapelet-based*, which aim at finding a subsequence, called "shapelet", yielding the highest information gain that can discriminate among classes within a tree-based classification algorithm [41][33]. A third type of method, namely *dictionary-based* approaches, splits the time series in time windows and extracts patterns out of each window as features. Such methods tend to be faster than the aforementioned ones due to feature numerosity reduction. Examples are *Bag-of-Patterns* (BOP) [25], which uses piecewise aggregate approximation (PAA) through Symbolic Aggregation approXimation (SAX) words [24] and *Bag-of-SFA-Symbols* (BOSS) [37], which encodes subsequences through Discrete Fourier Transform (DFT).

In contexts where the inference is uncertain, ensemble algorithms have been shown to be able to capture different facets and types of data distribution [35]. Ensemble methods can be further divided into styles of *parallel* and *sequential*, where in the first case a number of classifiers are built/trained independently on the original data and the class of an unseen example is guessed by aggregating the outcomes of each parallel classifier (*e.g.* through bagging, voting, stacking); in the second case, classifiers function together in a pipeline with each taking input from the output of the previous classifier (e.g. boosting, cascading). In this paper, we promote the use of cascading classifiers [1], a type of ensemble where a single classifier is active at each time. This is a convenient choice when the data to be classified is heterogeneous and each standalone classifier is able to discriminate certain classes or domains with high accuracy, but is unable to classify other categories of objects [15]. Cascading classifiers, even in recent literature, mostly operate by applying classification models in sequence until a certain threshold of confidence is reached, such as in [4]. Another example is the work in [5], in which authors apply repeatedly binary classifiers to obtain a filtering-based ensemble applicable to a non-binary problem. They order classifiers by applying first the most confident ones (confidence is measured through multiple metrics). In this paper, we make use of a class filtering process in which classifiers are selected and ordered a priori and then classes are filtered out at each stage, which is poorly explored in literature. A similar approach is found in [32], – they

denominate it a "class set reduction" – in which a genetic algorithm is proposed as a meta-heuristic method to find the best combination and ordering of classifiers to be included. Differently from our approach, they only explore a reduced case study of two classifiers, whereas our solution supports $n$-classifiers, including strategies for ordering them, and is supported by extensive experimental evaluations including up to 5 classifiers.

The classification of open IoT data stemming from heterogeneous IoT devices has been investigated in the literature. The work in [8] proposes a PAA-based approach that treats the sensor data classification as a dictionary-based TSC problem and uses interval slopes as features. A different approach was taken in [6], aiming to assess the validity of open IoT data by comparing it with certified ground truth. In [27] open IoT data was classified using only the metadata provided, i.e. the user-assigned stream name. More recently, other research efforts have been conducted in this research domain, particularly oriented to the inference of the sensor type in building automation systems (BAS) application domain [14][19][18]. The majority of these works leverage TSC, while few others use the metadata associated with sensors [13]. A more recent work classifies numerical data from sensors through Image Encoded Time Series (IETS), also assessing the high performance of statistical features [20]. TSC and semantic inference methods, however, work well when the physical place to be monitored is small enough so that similar sensor readings tend to have a consistently similar trend and similar metadata encoding. On the other hand, outside the BAS application domain, there is a need for a combined approach and very little research has been conducted in such a direction. We also acknowledge related work in the field of classification of IoT data and devices from their network traffic profiles. It is the example of [40], which uses a multi-stage classification algorithm based on several domains of information, [30], which proposes a probabilistic framework using stacked autoencoders, and [26], which uses random forest trained on a whitelist of devices to identify possibly malicious ones in a network. However, these studies assume that IoT devices provide access at network level, which is usually not the case for open IoT data.

In summary, heterogeneous open IoT data obtained from open data sources often faces the drawback of being unlabeled (lack of metadata) and sparse, leading to hardly intelligible data values. Even in instances where some metadata is available, the meaningfulness of this metadata varies significantly, posing challenges in classifying the open IoT data. TSC algorithms, which have been used extensively in literature [8], bear in general low performance on open IoT data, as the data heterogeneity hinders the discovery of discriminant patterns. Our preliminary work [28] was the first approach that employed ensemble methodologies in order to consider a combination of both numerical characteristics of the open IoT data and its metadata for classification, however, the solution only supported two handpicked classifiers. Therefore, the work presented in this paper is, to the best of our knowledge, the first complete approach able to combine automatically and efficiently an arbitrary number of classifiers for the task of open IoT data annotation.

## III. Metadata-Assisted Cascading Ensemble (MACE) Algorithmic Framework

In this section, we outline the problem of classifying open IoT data in order to enable automatic annotation and propose to tackle it through our algorithmic framework. Hereafter, we will use the term ***"datastream"*** to refer to an individual series of chronologically ordered numerical *sensor readings* – in this paper, these are also referred to as "*observations*" or "*measurements*" – each of them produced by a single real or virtual IoT sensor, together with its metadata. Therefore, we assume that each sensor reading of a single datastream is about a single data type (*e.g.* temperature). To avoid misinterpretations, a datastream is not necessarily bound to the classic concept of streaming [29], instead, datastreams can also be stored and queried through offline batch analyses.

### A. Problem Formulation

Formally, we are given $n$ IoT datastreams $\mathbf{NS} = \{\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_n}\}$ and $\forall i \in [1, n] : \mathbf{S_i} = \{\mathbf{D}_i, \mathbf{R}_i\}$. Specifically, $\mathbf{D}_i$ represents a dictionary of metadata (*e.g.* XML-like attribute-value pairs) with or without annotations (*i.e.* the attribute has a value or not). The time-ordered fractional sensor readings are contained in tuple $\mathbf{R}_i = \langle r_{i,1}, r_{i,2}, \ldots, r_{i,m} \rangle$ that resembles a numerical time series. Each $r_{i,j}$ is a reading from $i$-th stream at time $\tau_j$. For instance, consider a temperature stream $\mathbf{S_i}$ with annotated metadata $name$ and $description$ while the annotation for metadata $type$ is missing; then $\mathbf{D}_i = \{(name : \text{"}outdoorTemp\text{"}), (description : \text{"ESP8266 with DHT11"}), (type : \text{" "}), \ldots\}$ and $\mathbf{R}_i$ could be $\langle 21.5, 23, 25.4, \ldots \rangle$. Without loss of generality, we assume datastreams to have the same temporal length $m$ (thus, $\mathbf{NS}$ can be viewed as a column-ordered matrix of size $n \times m$) and time intervals $\{\tau_{j+1} - \tau_j\}$ between two consecutive readings in each datastream are near-uniform.

As, in our scenario, several textual metadata $type$ values that indicate the classes of datastreams are missing, the **goal** of the annotation problem is to recover the datastream classes (the $type$ value in $\{\mathbf{D}_i\}$) from *both* the information of sensor readings and the available metadata. When both are available, then we are in presence of data from multiple *information domains*. To achieve this goal, the possible categorical types are mapped to numerical labels $\{y_i\}$ first, i.e. datastreams are transformed to the form $\{\mathbf{S_i}, y_i\}$. Then, datastreams in $\mathbf{NS}$ are split into a training set with size $t$ and a test set with size $n - t$. Specifically, from the training set, existing classes are mapped to $c$ distinct numerical labels $\mathbf{L} = \{l_1, l_2, \ldots, l_c\}$ and, normally, $c \ll n$. In the training phase classifiers are built for the later testing phase to infer, from $\mathbf{L}$, which class each missing $y_i$ in the test set belongs to. In the following, § III-B adopts the formulation/notation defined here. Throughout the paper, we use bold symbols to denote multi-dimensional data structures such as vectors, matrices, and dictionaries. Note that, for the sake of simplicity, in our problem formulation

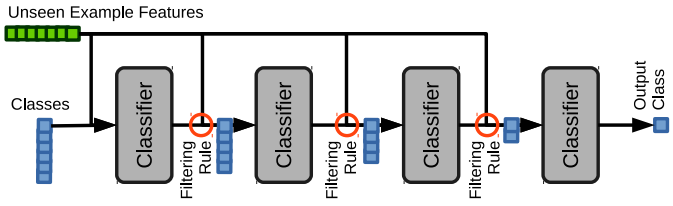



Fig. 1. A pictorial overview of the MACE framework.

the *type* metadata embeds any attribute-wise data class. This means that if a data type would ideally include one or more sub-types, then we flatten such a hierarchy by making each sub-type a *type* of its own. For instance, if a scenario admits the temperature to be measured both in Celsius and Fahrenheit, then this would result in two totally detached types; the same applies, *e.g.*, for indoor and outdoor temperature. In practice, we assume observations belonging to the same *type* to have compatible behaviors, which is necessary for applying supervised approaches.

### B. An Overview of the MACE framework

In this section, we propose a Metadata-Assisted Cascading Ensemble (MACE) algorithmic framework. For each unclassified datastream, MACE acts as a filtering-based multi-class cascading framework which, at each step, selects from the output of the current standalone classifier the most probable remaining classes that the next classifier must consider (*i.e.* an "output-filter-relay" cascading process).

The overall mechanism of $p$-stage MACE (where $p$ is the number of selected standalone classifiers in the ensemble) is displayed in Figure 1. Following the formulation from § III-A, suppose that a set of $\Theta$ supervised classifiers $\mathbf{\Gamma} = \{\Gamma_1, \ldots, \Gamma_\Theta\}$ are independently trained on the same training set $\mathbf{TRAIN} = \{(\mathbf{S_1}, y_1), \ldots, (\mathbf{S_t}, y_t)\}$ with ground truth classes $|\mathbf{L}| = c$ and $\forall i \in [1, t] : \mathbf{S_i} = \{\mathbf{D}_i, \mathbf{R}_i\}$. Then the testing set is $\mathbf{TEST} = \mathbf{NS} \backslash \mathbf{TRAIN}$ with datastreams having unknown class labels. For a test example $(\mathbf{T}, y) \in \mathbf{TEST}$, its class $y$ can be easily inferred from a standalone classifier such as $\Gamma_i(\mathbf{T})$ that outputs a rank/tuple of classes in $\mathbf{L}$ (descendingly ordered by classification probabilities) for $\mathbf{T}$.

On the other hand, within MACE we define the composed filtering classifiers in an ordered cascading ensemble as functions $\hat{\mathbf{\Gamma}} = \left\langle \hat{\Gamma_1}, \ldots, \hat{\Gamma_p} \right\rangle$. Let $\mathbf{C_{out,i}}$ be the filtered output class tuple from classifier $i$ and $f$ be the class filtering function (to be described later), then we have

$$\mathbf{C_{out,1}} = \hat{\Gamma_1}(\mathbf{T}, \Gamma_1, \mathbf{L}) = \Gamma_1(\mathbf{T}) \tag{1}$$

$$\forall i \in [2, p] : \mathbf{C_{out,i}} = \hat{\Gamma_i}(\mathbf{T}, \Gamma_i, f(\mathbf{C_{out,i-1}})) \tag{2}$$

where equation (1) is the base case and for (2) we define

$$\hat{\Gamma_i}(\mathbf{T}, \Gamma_i, f(\mathbf{C_{out,i-1}})) = \Gamma_i(\mathbf{T}) \cap f(\mathbf{C_{out,i-1}}) \tag{3}$$

and denote $\cap$ as an order-preserving (according to the tuple operand) intersection operator between a tuple output from $\Gamma$ and a filtered class set from $f$ (both contain elements in $\mathbf{L}$), rather than its conventional use between two unordered sets. Hence, we can regard $f$ as a filtering or mapping function that always keeps high-probability classes (i.e. top-ranked classes) for the multifaceted considerations of a series of classifiers or otherwise removes low-probability classes. For instance, for a test example, $\mathbf{T}$ assuming initially there are 5 classes such that $\mathbf{L} = \{1, 2, 3, 4, 5\}$ to be chosen from and the first two standalone classifiers (without filtering) produce class ranks $\Gamma_1(\mathbf{T}) = \mathbf{C_{out,1}} = \langle 2, 4, 5, 3, 1 \rangle$ and $\Gamma_2(\mathbf{T}) = \langle 3, 4, 5, 2, 1 \rangle$ respectively. Further, we assume, with the involvement of a filtering strategy/rule the first filtered class set becomes $f(\mathbf{C_{out,1}}) = \{2, 4, 5\}$. Therefore, the sequentially ensembled class output from the second classifier evaluates to $\mathbf{C_{out,2}} = \langle 3, 4, 5, 2, 1 \rangle \cap \{2, 4, 5\} = \langle 4, 5, 2 \rangle$, preserving the ordering in $\Gamma_2(\mathbf{T})$. Note that in the above recursive formulation (2) of the MACE framework, the ensemble outputs follow the property that $\forall i \in [2, p] : |\mathbf{C_{out,i}}| \leq |\mathbf{C_{out,i-1}}|$ due to $|f(\mathbf{C_{out,i}})| \leq |\mathbf{C_{out,i}}|$, and in the end $|\mathbf{C_{out,p}}| = 2$ containing its TOP-1 result as the final class prediction. In other words, in the testing phase, MACE creates filters by consecutively applying the filtering function $f$ on output ranks $\mathbf{C_{out,1}}, \ldots, \mathbf{C_{out,p}}$ until $|f(\mathbf{C_{out,p}})| = 1$.

The training phase of MACE is composed of the following two Cross-Validation (CV) steps:

- Stratified $K$-fold Cross-Validation with standalone classifiers from $\mathbf{\Gamma}$.
- Stratified $K$-fold Cross-Validation with the cascading ensemble $\hat{\mathbf{\Gamma}}$.

Together above with the MACE recurrence relation formula (2), the first CV step is to determine the final selection and ordering of standalone classifiers (§ III-C) from $\{\Gamma_1, \ldots, \Gamma_\Theta\}$ for composing an ordered sequential ensemble $\left\langle \hat{\Gamma_1}, \ldots, \hat{\Gamma_p} \right\rangle$. The second CV step is for training the designed filtering strategies (§ III-D) in MACE, that is, training the parameter of $f$. We pick the fold number $K = 5$ for both CV steps while stratification ensures classes are balanced in the training set. In the rest of this section, we detail both CV procedures.

### C. Selecting and Ordering Classifiers

It is not hard to see that, despite the performance of individual classifiers, the efficacy of MACE heavily relies on the organization and configuration of cascading classifiers. Deciding an optimal subset $\mathbf{\Gamma}' \subseteq \mathbf{\Gamma}$ of standalone classifiers to form $\hat{\mathbf{\Gamma}}$ as well as their optimal ordering in $\hat{\mathbf{\Gamma}}$ implies a computationally intensive cross-search phase in which we need to evaluate all the $k$-permutations of $\Theta$ elements *for each* $k \leq \Theta$. To overcome this issue, we instead propose two simple yet effective and efficient heuristics in MACE for (i) selecting the subset of classifiers and (ii) ordering them. Let us first define the *top-accuracy* of a standalone classifier $\Gamma_i$ with a function $\mathcal{A}_i(k)$ that returns the $K$-fold CV accuracy of its top-$k$ predicted classes ranked by probability. In other words, the function calculates the percentage of true positives included in the classifier's predicted top-$k$ rank over all test examples. For instance, $\mathcal{A}_i(1)$ is the actual accuracy of $\Gamma_i$, while, by construction, $\forall i : \mathcal{A}_i(c) = 100\%$. Furthermore, all $\mathcal{A}_i(k)$ functions are monotonically increasing by construction.

In order to decide $\mathbf{\Gamma}'$ from $\mathbf{\Gamma}$, we employ a *dominating heuristic* based on the top-accuracy values of standalone classifiers. Now, given the top-accuracies, we would ideally keep in $\mathbf{\Gamma}'$ only the classifiers from $\mathbf{\Gamma}$ that have the best top-accuracy for *at least one* value of $k$. In other words, only non-dominated classifiers are kept where, conversely, a *dominated classifier* $i$ satisfies:

$$\forall k \in [1, c]\ \exists j \neq i : \mathcal{A}_j(k) \geq \mathcal{A}_i(k). \tag{4}$$

In order to determine $\hat{\mathbf{\Gamma}}$, selected classifiers in $\mathbf{\Gamma}'$ are ordered based on another *backward search heuristic* as follows. The classifier search starts backwards from the largest $k = c$ down to $k = 1$. At any $c > k \geq 1$, the *dominating classifier* with the highest top-$k$ accuracy has an index $\arg\max_i \mathcal{A}_i(k)$ (ties not allowed) and the heuristic composes the ensemble with such *unique* dominating classifiers found from backward search as

$$\hat{\mathbf{\Gamma}} = \left\langle \Gamma_{\arg\max_i \mathcal{A}_i(k_1)}, \Gamma_{\arg\max_i \mathcal{A}_i(k_2)}, \ldots, \Gamma_{\arg\max_i \mathcal{A}_i(k_p)} \right\rangle \tag{5}$$

where $c > k_1 > k_2 > \ldots > k_p \geq 1$ and the constrained unique condition in the search is:

$$\forall j \in [1, p], h \in [j+1, p] : \arg\max_i \mathcal{A}_i(k_j) \neq \arg\max_i \mathcal{A}_i(k_h). \tag{6}$$

Essentially, the condition is to decide what the discrete points $k_j$'s are for extracting unique classifiers in a backward non-repeatable fashion. The search is also made more efficient with the dominating heuristic applied before in order to remove the dominated classifiers. The rationales behind our dominating and backward search heuristics are i) classifiers performing better for higher values of $k$ are suitable as front filtering classifiers as they are likely to get rid of wrong classes at an early stage; ii) vice versa, classifiers performing better for lower values of $k$ are more powerful in guessing the right class in the end, becoming a much easier task when many wrong classes have been already filtered out; iii) classifiers consistently performing worse than others should not be included in the ensemble. The effectiveness of the heuristics is demonstrated later in § V. In particular, examples of these heuristics in practice are given in § V-B.

### D. Class Filtering Strategies

A cascading filtering strategy defines the filtering factor/ratio/map between every two consecutive cascading classifiers in MACE. In order to properly configure $p$ selected and ordered classifiers to achieve better performance, the filtering function $f$ aiming at gradually removing wrong classes has to be determined, since, an inappropriate filter would impact negatively on the performance by either missing many classes or introducing much noise. The straightforward approach is a top-$k$ filtering strategy that tries all *discrete* values of $k \in [c]$ and chooses, at each filtering step $u$, the optimal top $k_u^*$ classes to keep from $\mathbf{C_{out,u}}$ which yields the highest accuracy in the second CV step for training the cascading ensemble $\hat{\mathbf{\Gamma}}$. However, such a method would imply trying all the possible values of $k$ for each of the $p-1$ filtering steps, *i.e.* as slow as taking $O(c^{p-1})$ computational steps. For this reason, we extend $f$ to be a generalized function such that $\forall u \in [1, p] : k_u = f(z, \mathbf{C_{out,u}})$ where $z$ is a constant filtering parameter. Such generalized definition reduces the ensemble parameter space into a common filtering parameter $z$ to be optimized across all classifiers. In terms of computation, this cuts the number of CV steps down to $O(1)$ (*i.e.* cross-validating the ensemble over all the possible values of $z$, which is constant). Function $f$ and its parameter $z$ can be designed differently for different filtering strategies. Here, we propose three specific filtering strategies that will be used in our experiments. The filtering strategies are named Top-$k$, PF, and SoF and are grouped into two macro-categories that will be described below: *rank-based* and *distribution-based*.

*1) Rank-Based Strategy:* This category of strategies aims to directly extract the most likely classes from the produced rank of output classes, that is, the list of classes descendingly ordered by their output classification probabilities. The straightforward Top-$k$ strategy, which simply slices from the top of the rank by directly specifying a value of $k$ for *each* filtering stage, is a part of this category. However, this strategy suffers from computational inefficiency in searching fixed $k$'s as stated above. In order to overcome this issue, we revise the Top-$k$ strategy such that the number of filtered classes between two sequential stages is determined heuristically as the maximum value of $k$ at which the former classifier $\hat{\Gamma}_i$ gets dominated by the latter $\hat{\Gamma}_{i+1}$, *i.e.* the maximum $k$ such that $\mathcal{A}_{i+1}(k) \geq \mathcal{A}_i(k)$. For instance, let us consider two classifiers $\Gamma_a$ and $\Gamma_b$, such that $\Gamma_a$ gets dominated by $\Gamma_b$ for $k \geq 6$, *i.e.* the top-accuracy of $\Gamma_b$ is greater than that of $\Gamma_a$ for such values of $k$. Then, after ordering the classifiers as $\hat{\mathbf{\Gamma}} = \langle \Gamma_b, \Gamma_a \rangle$, the Top-$k$ strategy selects heuristically as 6 the number of top-ranked classes to be kept after applying $\Gamma_b$.

We also define another rank-based adaptive strategy called Percentage Filtering (PF), which is defined by $\forall u \in [1, p] : k_u = |f(z, \mathbf{C_{out,u}})| = \lceil z \cdot |\mathbf{C_{out,u}}| \rceil$ where $z \in (0, 1)$. The generalized function $f$ simply specifies the top-ranked $k_u$ classes to be kept in a filter in terms of a ratio over the size of output classes. For instance, if $z = 0.25$, then the upper quartile of $\mathbf{C_{out}}$ is passed onto the next cascading stage. Note that when the size of cascading ensemble $p = 2$, PF is similar to top-$k$, for which PF can be seen as its generalization.

*2) Distribution-Based Strategy:* This category includes strategies taking into account the distributions of probabilities output by the classifiers in order to locate the portion of classes to be passed on. Under this category, we define an alternative strategy called Survival of the Fittest (SoF), which keeps the classes such that $P(l) \geq \mu + z\sigma$, where $P(l)$ is the probability attributed by a classifier to a class $l$, $\mu$ is the mean of the normalized classification probabilities and $\sigma$ is the standard deviation of these probabilities, tuned by a parameter $z$. Then, SoF can be defined by $\forall u \in [1, p] : f(z, \mathbf{C_{out,u}}) = \{l \in \mathbf{C_{out,u}} \mid P(l) \geq \mu_u + z\sigma_u\}$. This way, only outstanding classes are selected, also, this strategy produces filters of adaptive sizes as different unseen examples can cause different classification probability distributions.

Algorithm 1 below summarizes the entire algorithmic process of the MACE ensemble classification framework. The

feature extraction (line 1) is a generalized process depending on the type of features needed by the single standalone classifiers. Used standalone classifiers are detailed in § V. After such step, for notation simplicity, $\mathbf{TRAIN}$ and $\mathbf{T}$ are substituted with extracted feature data to be trained and tested respectively. In addition, the cascading process (lines 6-10) in the algorithm exits either when $|\mathbf{C_{out}}| = 1$ (perhaps several classifiers are left unused) or $\Gamma_u$ is the last classifier in the cascading ensemble.

**Algorithm 1** The overall algorithmic process of MACE

**Require:** Training set $\mathbf{TRAIN} = \{(\mathbf{S_1}, y_1), \ldots, (\mathbf{S_t}, y_t)\}$, a test example $(\mathbf{T}, y) \in \mathbf{TEST}$, a set of $\Theta$ classifiers $\mathbf{\Gamma}$
**Ensure:** output $y \in \mathbf{L}$
1: extract classes $\mathbf{L}$ and features for $\mathbf{TRAIN}$ and $\mathbf{T}$
*# Training phase*
2: perform stratified $K$-fold CV with each $\Gamma_i \in \mathbf{\Gamma}$ on $\mathbf{TRAIN}$ to extract top-accuracies $\mathcal{A}_i$'s
3: select $\mathbf{\Gamma'} \subseteq \mathbf{\Gamma}$ according to dominating heuristic eqn (4)
4: order $\mathbf{\Gamma'}$ to form $\hat{\mathbf{\Gamma}}$ according to backward search heuristic eqns (5) and (6)
5: select a filtering strategy $f$ with parameter $z$ and perform stratified $K$-fold CV with $\hat{\mathbf{\Gamma}}$ to train $z$ on $\mathbf{TRAIN}$
*# Testing phase*
6: $\mathbf{C_{out}} = \hat{\Gamma_1}(\mathbf{T}, \Gamma_1, \mathbf{L})$ # $\hat{\Gamma_i}$'s are computed from eqn (3)
7: **for** $\hat{\Gamma_u} \in \hat{\mathbf{\Gamma}} \setminus \hat{\Gamma_1}$ based on the ordering in $\hat{\mathbf{\Gamma}}$ **do**
8: $\quad \mathbf{C_{out,u}} = \hat{\Gamma_u}(\mathbf{T}, \Gamma_u, f(z, \mathbf{C_{out}}))$
9: $\quad \mathbf{C_{out}} = \mathbf{C_{out,u}}$
10: **end for**
11: **return** $y = \text{top-1}(\mathbf{C_{out}})$

## IV. Experimental Setup

We evaluate our proposal by running three types of experiments. In the first experiment (§ V-A) we test individually a large set of standalone classifiers as a baseline against our open IoT datasets. In the second experiment (§ V-B) we present the overall performance of MACE against the baseline. This experiment has multiple goals: first of all, it shows that our MACE heuristic ensemble strategy outperforms all the standalone classifiers that are part of it, while still holding a comparable training time. In fact, the whole process takes two CV steps: the top-accuracy estimation, which takes $O(|\mathbf{\Gamma}| \cdot t \cdot K)$ (where $t$ is the average time taken by standalone classifiers and $K$ is the number of folds in the CV stage), and the tuning of $z$, which takes $O(|\hat{\mathbf{\Gamma}}| \cdot t \cdot K \cdot Z)$ (where $Z$ is the number of tested $z$ values). Since $K$ and $Z$ are small constants, the time becomes $O(|\mathbf{\Gamma}| \cdot t)$. Furthermore, we show how this method outperforms Majority Voting, a widely used parallel ensemble strategy with comparable training time. Finally, in the third experiment (§ V-C) we evaluate the effectiveness of our heuristics for selecting and ordering classifiers against all the other possible combinations of classifiers obtainable from the same initial pool. This experiment is twofold: we compare the accuracy of the MACE heuristic ensemble first with its brute-force optimum and then with the whole set of all possible combinations and orderings.

In all experiments, we performed a stratified split over the datasets assigning 70% to the training set and 30% to the test set. All tested algorithms in this paper are implemented in Python 3.6.9 on top of `scikit-learn` [31] for the standalone algorithms, except the implementation of BOPF [23], for which we used the original C++ code provided by the authors. The code of the MACE framework is available at `https://github.com/matteodeggi/IoT_Classification`.

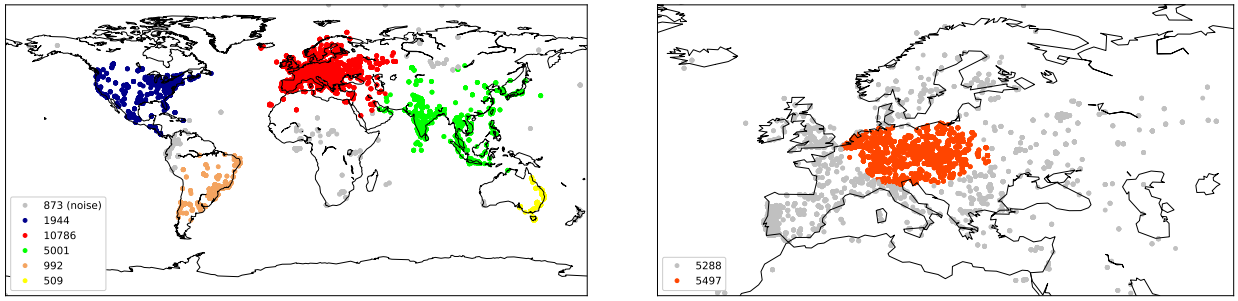
In the remainder of this section, we detail our open IoT datasets and describe briefly all the standalone classifiers used in the experiments.

### A. Open IoT Datasets

Here, we introduce the open IoT datasets we have created by extracting and cleaning open IoT data obtained from public data sources. The datasets cover a good representation of publicly available IoT data that differ in spatial granularity ranging from city-wide area, region-wide area to country-wide area. Specifically, the ThingSpeak dataset is our contribution to the current literature as we performed the extraction and it needed a thorough adaptation including cleaning and pre-processing to be used for the problem of annotation. The datasets are available at `https://github.com/stradivarius/TSopendatastreams`.

*1) ThingSpeak (abbr. **TS**):* a dataset that we extracted from the online cloud platform of the same name (`https://thingspeak.com/`) to which users can subscribe and push sensor data produced by their IoT devices onto personal public "channels" through dedicated APIs. Each channel hosts a set of datastreams (one for each measurement made by the sensor connected to the IoT device) and user-annotated metadata: a name, a description, a name for each datastream, and a geolocation in GPS coordinates. Metadata is user-assigned, thus it can vary in accuracy significantly. We created our dataset in a similar way as in [28], with a significant improvement both in the methodologies and in the number of resulting instances. We scraped all the TS channels (756,322 at the time of writing) through a dedicated HTTP call[1] that returns a JSON object, containing the metadata and the last 8000 readings in year 2019 from the datastreams belonging to the queried channel. Subsequently, all the datastreams were made independent from their channel, filtering out those with either no location data, a poor amount of sensor readings or no public access. We then clustered, for each datastream, the data points in 15min time chunks and kept the average as a single data point value for each of them. In order to make the datastreams consistent with each other, we also operated a spatial and temporal clustering to select a *country-wide* area with a reasonable number of observations within a reduced time window (∼24hrs). Hence, we applied a DBSCAN algorithm [38] to all the locations of the datastreams isolating the major clusters. As can be seen in Figure 2(a), the clusters roughly reflect the continents. We then selected the most populous one and operated a second filtering DBSCAN to identify the most

[1] `https://thingspeak.com/channels/{#channel}/feed.json?results=8000\&start=2019-01-0100:00:00`




(a) DBSCAN on the whole world ($\varepsilon = 0.01, min\_samples = 300$). (b) DBSCAN on Europe ($\varepsilon = 0.45, min\_samples = 1700$).

Fig. 2. Spatial clustering operations through DBSCAN in building the TS dataset. The number of dots per cluster is indicated in the legend.

densely populated country-wide area. Figure 2(b) shows the chosen area in Europe, which comprehends parts of Germany, Poland, Czech Republic, Slovakia, Hungary, and Austria. At the same time, we identified the best time window by clustering the measurements per day and selecting the day for which the number of datastreams with at least $\psi$ observations in such day is maximum. The parameters $\psi$ for temporal clustering and the parameters $\varepsilon$ and $min\_samples$ for DBSCAN have been tuned experimentally through a grid search. Finally, we homogenized all the datastreams interpolating the missing points by means of cubic splines, then we annotated manually the dataset, filtering out datastreams for which the class was uncertain from human analysis (after an automatic preprocessing which includes translation into English through Google Translate API). The final dataset contains 2121 datastreams with metadata (therefore, with multiple information domains), each of them with 96 data points and belonging to 21 different classes: appliance temperature, humidity, pressure, indoor air temperature, outdoor air temperature, wind speed, light, air quality, wind direction, wind temperature, voltage, current intensity, wireless RSSI, heat index, dewpoint, rain index, UV, PM 1, PM 2.5, PM 10 e $CO_2$. This dataset is released publicly by us and comprehends a *country-wide* area.

*2) Swiss Experiment (abbr.* ***Swissex****):* is a web platform that enables publishing environmental sensor data located within the Swiss Alps mountain range in real-time. Data is highly noisy, comes from different microscopic locations and it is taken within different time spans. The sampling rate is also different among sensors making the phase shift of data series very significant. Neither semantic annotation nor timestamps were originally provided. To the best of our knowledge, the Swiss Experiment dataset (for which an annotated version is available at `http://lsirpeople.epfl.ch/qvhnguye/benchmark/`) is one of the few IoT heterogeneous datasets used in research for tasks similar to ours [8]. The dataset contains 346 datastreams without metadata (therefore, with a single information domain), each of them with 445 data points and belonging to 11 different classes: $CO_2$ (carbon dioxide), humidity, lysimeter, moisture, pressure, radiation, snow height, temperature, voltage, wind speed, and wind direction. The original data is organized in time series of slightly different lengths, therefore we cut each time series to the length of the shortest stream in the dataset. Swissex is an example of a *region-wide* dataset.

*3) Urban Observatory (abbr.* ***UrbObs****):* is a pioneering programme led by the University of Newcastle within the development of an urban sensor network in the city of Newcastle, UK, which produces publicly available real-time environmental data [21]. The data from one day has been extracted from `http://newcastle.urbanobservatory.ac.uk/` and pre-processed into a dataset of 1065 datastreams without metadata (therefore, with a single information domain), each of them with 864 data points and belonging to 16 different classes: $NO_2$ (nitrogen dioxide), wind direction, humidity, wind speed, temperature, pressure, wind gust, rainfall, soil moisture, average speed, congestion, traffic flow, journey time, sound, CO (carbon monoxide), NO (nitrogen monoxide). IoT data in this dataset is highly correlated, both because the region of interest is *city-wide* and because it comes from the same source. As a matter of fact, the UrbObs data does not suffer from the open crowdsourced open IoT data issue reported in § III, as data is actually annotated; we used it as a comparison for our methods.

### B. Standalone Classifiers

For our experiments, we make use of three families of supervised classification algorithms: Time Series Classification (TSC) algorithms, standalone classifiers learning from our engineered statistical features, and a Natural Langauge Processing (NLP) classifier using metadata.

*1) Time Series Classification (TSC) Classifiers:* TSC is the state-of-the-art approach adopted to address the problem of annotating open IoT data, which, being by nature organized in series, appears to be highly suitable for this type of algorithms. We make use of the following well-known TSC algorithms:

- *One-Nearest-Neighbor with Euclidean Distance (1NN-ED):* a whole-series naïve method that operates a 1NN approach using the sum of the distances of data points between two series.
- *One-Nearest-Neighbor with Dynamic Time Warping (1NN-DTW):* considered the golden standard in TSC, it is a whole-series method that operates an optimal alignment between two series when calculating the distance. Despite being well-performing it is computationally expensive. We use the fast implementation presented in [36].

- *Bag of Pattern Features (BOPF):* a linear-time dictionary-based algorithm using the information gain of SAX-encoded patterns [23].
- *Slope Distribution Encoding (SDE):* a dictionary-based algorithm that has been used in literature to deal with IoT data, therefore meaningful for comparison [8].
- *Learning Time-Series Shapelets (LTS)*: a shapelet-based algorithm that finds a matching of subsequences in series with a high information gain [17].

*2) Bag-of-Summaries (BOS) Classifiers:* With Bag-of-Summaries (BOS) we mean a set of statistical features extracted from a time series that leverage different aspects and therefore can provide several planes of separation between classes depending on their information gain. More in detail, we extracted 11 features from the datastreams without normalization: mean, median, maximum, minimum, standard deviation, root mean square error (RMS), quantile, inter-quantile range (IQR), kurtosis, and range. With the engineered feature sets, we then experimented a set of standalone vanilla classification algorithms widely used in machine learning: C4.5 Decision Tree, Support Vector Machines (SVM), k-Nearest Neighbors (kNN), Logistic Regression (LR), Ridge Classifier and Gaussian Naïve Bayes (GNB). These algorithms have been optimized via a grid search on their parameters. Together with standalone classifiers, we also experimented standard ensemble classifiers using two approaches that rely on the replication of one base classifier either in parallel (bagging) or in sequence (boosting). For both categories we chose decision tree as a base classifier, resulting in Random Forest (RF) for bagging ensemble and Gradient Booster (GBoost) for boosting ensemble.

*3) Metadata-Based NLP Classifier:* As said, sometimes open IoT data comes with textual metadata (in our case the TS dataset) that carry meaningful information. In particular, MACE takes into account the datastream name, other metadata such as description and channel name can also be considered. Since most datastream names are in the form of abbreviations, a fuzzy string matching-based classifier focusing on the "shapes" of words would be more appropriate than a semantic-based NLP classifier. Building on such considerations, a simple supervised dictionary-based NLP classifier is adopted with its earlier version introduced in [27]. Algorithm 2 outlines the classification algorithm: in the training phase, a "dictionary" for each class $L_j \in \mathbf{L}$ is constructed in the form of Bag-of-Words ($BOW_j$) including all datastream names in metadata attributed to datastreams within the same class (line 2); in the testing phase, for each class $L_j$, the respective minimum edit distance $d_j = min\{ed(w,s) \,|\, s \in BOW_j\}$ of a testing example $w$ is computed (lines 4-5). The predicted class is the one with the minimum edit distance; we also compute the probability of belonging to a class by inverting and normalizing the distances. We chose as $ed$ the Damerau-Levenshtein edit distance [10] normalized by the maximum length between two words, which is why we name the classifier Dictionary Damerau-Levenshtein NLP (DDL-NLP). Other edit distances, such as the Jaccard and the Jaro-Winkler, have been tested with slightly inferior accuracies.

**Algorithm 2** DDL-NLP Algorithm

**Require:** Training set $\mathbf{TRAIN} = \{(\mathbf{S_1}, y_1), \ldots, (\mathbf{S_t}, y_t)\}$, test example $(\mathbf{T}, y)$
**Ensure:** $y \in \{1, 2, \ldots, c\}$
1: **for all** $(\mathbf{S}_i, y_i) \in \mathbf{TRAIN}$ **do**
2: $\quad BOW_{y_i} \leftarrow \mathbf{D_i}('name')$ $\quad$ # $\mathbf{S_i} = \{\mathbf{D}_i, \mathbf{R}_i\}$
3: **end for**
4: **for** $j := 1$ **to** $c$ **do**
5: $\quad d_j = min\{ed(\mathbf{D}('name'), s) \mid s \in BOW_j\}$ # $\mathbf{T} = \{\mathbf{D}, \mathbf{R}\}$
6: **end for**
7: **return** $y = \{L_j \mid d_j \in min\{d_1, \ldots, d_c\}\}$

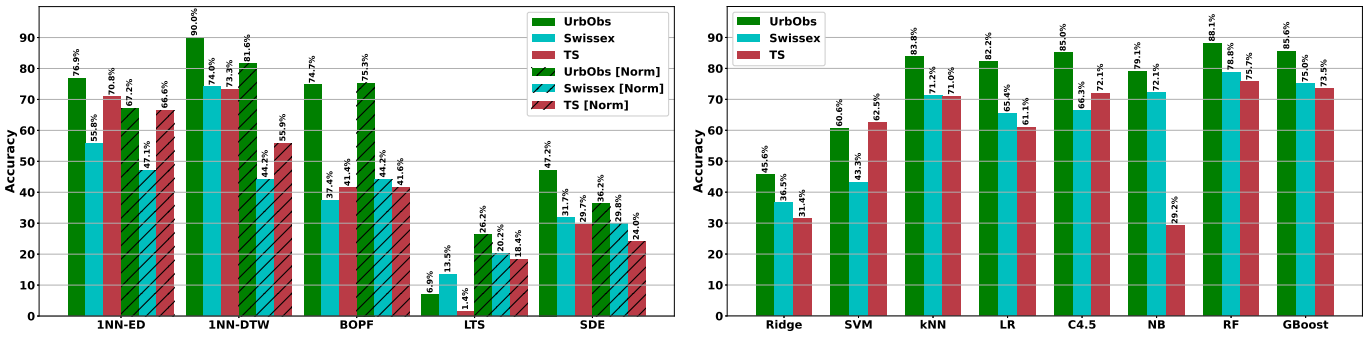
## V. Experiments

In this section, we assess the performance of the MACE framework with all the cascading filtering strategies proposed in § III-D. By using the MACE framework we imply that the ensemble of classifiers are selected through the *dominating heuristic* and ordered through the *backward search heuristic*, both introduced in § III-C, as they are embedded in the framework itself.

### A. Performance of Standalone Classifiers

The accuracy of the considered TSC and BOS standalone classifiers across datasets can be observed in Figure 3, while the DDL-NLP classifier has been tested separately. TSC classifiers (Figure 3(a)) have been applied both on $z$-normalized data – as per their canonical application – and on non-normalized data. From the figure, whole-series algorithms have decent performances on non-normalized data, better than their normalized counterpart; this happens because such methods, in contrast with the others, can capture properties of the data magnitude when applied to non-normalized data. However, looking at the performance of other TSC algorithms, we realize how data trend is not a sufficient discriminant. Both shapelet-based and dictionary-based algorithms fail across IoT datasets, with the exception of the city-wide UrbObs, on which BOPF yields an acceptable accuracy.

The performance of the considered running standards classifiers on BOS features across open IoT datasets can be observed in Figure 3(b). It is immediately noticeable that BOS algorithms perform better than TSC algorithms, as they can capture several characteristics of the data types that are not necessarily connected with their trend (*e.g.* how a temperature stream typically looks like, rather than the similarity between two temperature trends). As BOS features can capture different facets of data distribution, choosing sequentially which classes they discriminate the most is typically a tree-based approach, in fact, we observe that a simple decision tree outperforms most TSC classifiers and RF, being its ensemble, appears to be the most promising classifier for these open IoT datasets. We also observe that the golden standard non-normalized 1NN-DTW achieves similar performance, because it is able to capture in a different way the similar characteristics as BOS features do, however, we recall that it is an exceptional overkill for this problem as it is computationally much more




(a) TSC algorithms (b) BOS algorithms

Fig. 3. Performance evaluation of TSC and BOS standalone algorithms with optimized parameters against the open IoT Datasets

expensive than BOS methods by several orders of magnitude [28]. The DDL-NLP algorithm standalone (not shown in the figure) yields an accuracy of **73.2%**, which is aligned with the best-performing BOS classifiers, although using a completely different dimension of information.

### *B. Accuracy and F1-Score of the MACE Framework*

This experiment aims to evaluate the performance of the MACE framework given a sufficiently comprehensive initial pool of classifiers $\mathbf{\Gamma}$. The pool has been defined so that $|\mathbf{\Gamma}| = 5$, as a higher number would jeopardize our capability of evaluating the dominating heuristic against an optimum (see § V-C), which means brute-forcing over all the possible combinations and orderings of the classifiers. As a result, we did not consider classifiers that have poor standalone performances (such as Ridge classifier or BOPF) as well as those that have an impractically long running time (such as 1NN-DTW). The final pool of classifiers is $\mathbf{\Gamma} = \{1\text{NN-ED}, \text{GradBoost}, \text{RF}, \text{kNN}, \text{SVM}\}$ for UrbObs and Swissex and $\mathbf{\Gamma} = \{1\text{NN-ED}\ , \text{GradBoost}, \text{RF}, \text{kNN}, \text{DDL-NLP}\}$ for ThingSpeak. The rationale behind this choice is that, when possible, we want to exploit classifiers operating in different domains, in order to bring much more expressiveness to the final ensemble. ThingSpeak possesses metadata as a powerful source of alternative information from a "different point of view", which is highly desirable in ensembles. This is why we included DDL-NLP in its pool. By using these initial pools we executed the MACE framework by using all the filtering strategies reported in § III-D, each of them with their parameter $z$ tuned through CV.

Before assessing the performance, we visually show how MACE selects and orders the classifiers in the pool through the heuristics, according to § III-C. Figure 4 shows the top-accuracies $\mathcal{A}_i(k)$ of all classifiers $\Gamma_i \in \mathbf{\Gamma}$ against all three datasets in experiments, with $k$ on the $x$-axis. For UrbObs in Fig. 4(a), GradBoost classifier, kNN classifier, and SVM classifier are always dominated by some other classifier and hence should be discarded, leaving classifiers 1NN-ED and RF. Then, for the purpose of validating the backward search heuristic, we start looking backwards from $k = 16$. When $k > 7$, 1NN-ED uniquely dominates RF, so 1NN-ED is put as the first classifier in the ensemble and, since from $k = 7$ downwards RF dominates 1NN-ED, clearly RF is put as the second. As a result, for UrbObs we will select 1NN-ED followed by RF and, when using the Top-$k$ filtering strategy, heuristically set their filtering value of $k$ to 7. A similar process happens in Fig. 4(b), where only classifiers 1NN-ED and RF are kept after applying the dominating heuristic. At $k = 10$ their tie breaks with 1NN-ED uniquely dominating RF, swapping their condition at $k = 7$, where RF starts to take the lead until the end $k = 1$. Hence, according to the unique condition (6), the only $k_j$'s to consider are the change points $k = 10$ and 7 since 1NN-ED also dominates the interval $[8, 9]$ and RF dominates $[1, 6]$, where the unique classifier condition is not met within these intervals. In a similar way, for ThingSpeak, classifiers DDL-NLP and RF are chosen.

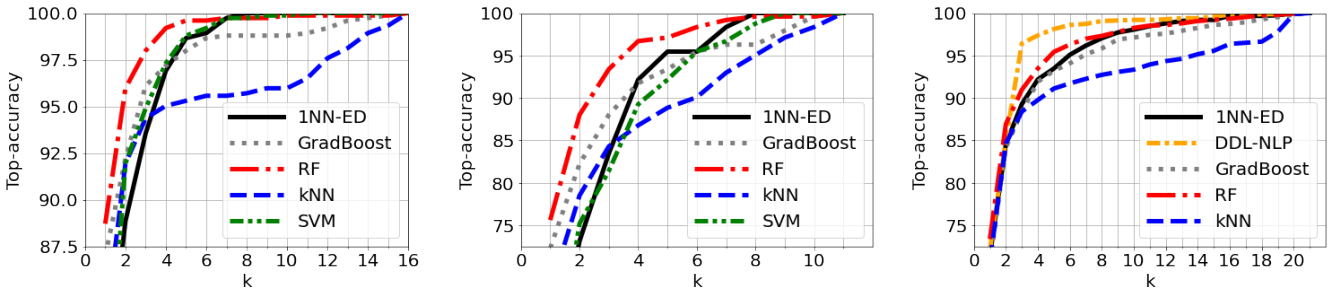
The accuracy results for each dataset are reported in Table I, where we observe a positive outcome: first of all, the MACE framework outperforms each and every standalone classifier in the pool for all the filtering strategies. This happens by a couple of percentage points for UrbObs up to more than ten for Thingspeak, proving the effectiveness of our solution. Secondly, we report in Table II the F1-Score of each experiment. Since our heuristics are accuracy-oriented in CV phase, we consider the F1-Score to be a side-effect (in fact, F1-scores were not presented in § V-A for space constraints).

| | **UrbObs** | **Swissex** | **TS** |
|---|---|---|---|
| Best standalone | 88.1% | 78.8% | 75.7% |
| Majority Voting | 89.1% | 83.7% | 79.6% |
| MACE with Top-$k$ | 90.6% | 84.6% | 84.3% |
| MACE with PF | 90.6% | 83.2% | 83.4% |
| MACE with SoF | 90.6% | 82.7% | 88.2% |
| MACE brute-force (SoF) | 92.2% | 86.5% | 88.5% |

TABLE I
ACCURACY OF ENSEMBLE STRATEGIES WITH AN INITIAL POOL $\mathbf{\Gamma}$ OF 5 CLASSIFIERS.

| | **UrbObs** | **Swissex** | **TS** |
|---|---|---|---|
| Best standalone | 87.9% | 83.2% | 71.9% |
| Majority Voting | 86.4% | 81.6% | 57.8% |
| MACE with Top-$k$ | 87.9% | 83.2% | 67.7% |
| MACE with PF | 87.9% | 83.2% | 63.7% |
| MACE with SoF | 87.9% | 80.0% | 79.4% |
| MACE brute-force (SoF) | 88.7% | 84.0% | 76.7% |

TABLE II
F1-SCORES OF ENSEMBLE STRATEGIES WITH AN INITIAL POOL $\mathbf{\Gamma}$ OF 5 CLASSIFIERS.




(a) UrbObs (b) Swissex (c) ThingSpeak

Fig. 4. CV top-accuracy of standalone classifiers over all three datasets. Note that these are CV accuracies, therefore different from the ones presented in § V-A, which are reported against the test sets.

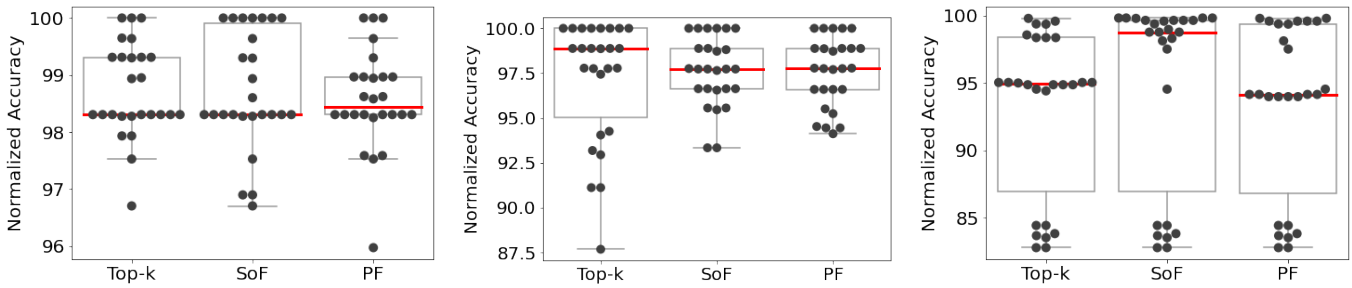
Nevertheless, we observe that the MACE framework obtains higher or equal values compared to the standalone ones for at least one filtering strategy. For UrbObs we notice that the filtering strategies are somewhat homogeneous and yield the same accuracy, this is probably due to the already high performance of individual classifiers on the dataset. Furthermore, we report, for each dataset, the accuracy and F1-Score obtained through majority voting. For a well-calibrated strategy, we adopted Soft Voting, which predicts the class label based on the maximum of the averages of the predicted probabilities. The latter strategy handles better classifiers operating in different domains.

In summary, MACE is designed for learning from heterogeneous IoT datasets containing multiple information domains. Indeed, in the experiments, we observe that the MACE framework outperforms Soft Voting, especially when classifiers belong to multiple information domains (*i.e.,* in ThingSpeak).

More in detail, despite the more complex structure of MACE in comparison with Majority Voting, our heuristic (described in § III) implies a linear computational complexity, which is in line with parallel approaches. For the sake of completeness, Tables I and II also include the superior best accuracy and corresponding F1-Score (as a by-product) of MACE from its slower brute-force implementation. At the same time, MACE with heuristics yields a consistently higher accuracy in presence of multiple information domains, while holding similar (in most cases slightly better, rarely slightly worse – a marginal difference of 1%) accuracy to parallel ensembles given a single information domain. Moreover, even though the three class filtering strategies are performing differently (a one-size-fits-all solution is hard to obtain in presence of such heterogeneity in IoT datasets), we observe that for all of them the above mentioned performance is consistent.

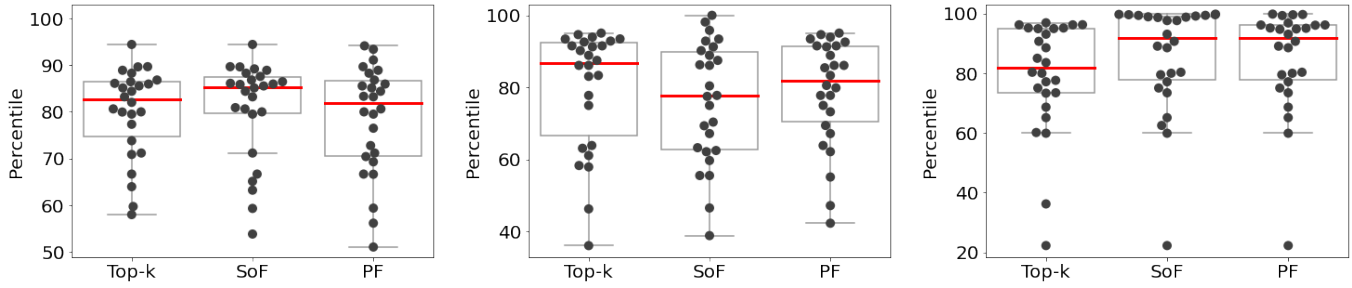
### C. Evaluation of the Selection and Ordering Heuristics

In this section we discuss the performance of the heuristic strategies presented in § III-C within the MACE framework. Specifically, we compare the accuracy of MACE against (i) the maximum accuracy obtained by brute-forcing with the same initial classifier pool $\mathbf{\Gamma}$ and (ii) the accuracies obtained by all the combinations of classifiers belonging to the same initial pool $\mathbf{\Gamma}$. Differently from the results shown in § V-B, here we do not always use the set of five mentioned classifiers as the starting pool $\mathbf{\Gamma}$, rather, we define multiple starting pools. Let us define the set $\mathbf{\Gamma}$ used in § V-B as $\mathbf{\Gamma_{tot}}$ instead. We then run several tests by setting $\mathbf{\Gamma}$ to every single member of the power set $\mathbb{P}(\mathbf{\Gamma_{tot}})$ that contains at least two elements (we do not use initial pools that are empty or singletons). When $|\mathbf{\Gamma_{tot}}| = 5$, this results in 26 different initial pools, each of them as a separate experiment with a different starting pool $\mathbf{\Gamma}$, such that $2 \leq |\mathbf{\Gamma}| \leq 5$. For *each* of these experiments, we compute the accuracy of *every* possible MACE ensemble obtainable from $\mathbf{\Gamma}$, in other words, every ordered $k$-permutation of classifiers, with $k$ in the range of $|\mathbf{\Gamma}|$. In Figures 5 and 6 we show the results for each of the three datasets as well as each filtering strategy in the form of boxed scatterplots, which best highlight the distributions. In both figures, each dot corresponds to the result of one of the 26 experiments, which are repeated over the datasets and for each filtering strategy. Figure 5 shows, for each experiment, the accuracy achieved by MACE normalized by the best accuracy achievable through brute-forcing over the same initial pool $\mathbf{\Gamma}$. A value of 100% means that the heuristic selects the actual best possible combination and order. This figure aims to show how much our heuristic selection differs from the actual optimum. In the vast majority of cases, we observe a very tiny difference in accuracy. For UrbObs, the accuracy of the heuristic is, on average, 1-2% lower than the optimum for all three filtering strategies; similar results are displayed for Swissex. For both datasets, no major differences are found among the filtering strategies. Such differences are more noticeable on ThingSpeak, for which SoF is by far the best strategy in this evaluation, whereas Top-$k$ and PF present a 5% lower accuracy for the heuristic on average. We also observe, for all three filtering strategies, a cluster of results that perform slightly worse. These occur when DDL-NLP $\in \mathbf{\Gamma}$ and RF $\notin \mathbf{\Gamma}$, thus, according to Figure 4(c), DDL-NLP is always selected as standalone, because the CV accuracy of the numeric algorithm yields a lower performance. In Figure 6, for each experiment, we do not take into account only the best ensemble obtainable from $\mathbf{\Gamma}$ (as in Figure 6), instead, we consider all the possible ensembles. The figure shows, for each experiment, the percentile at which MACE, with the filtering strategy tuned through CV, is ranked against all the other ensembles obtainable from the same pool $\mathbf{\Gamma}$ across all filtering strategies. Thus, this figure aims to show how good the standing/rank of the heuristic is, together with the CV




(a) UrbObs (b) Swissex (c) ThingSpeak

Fig. 5. Relative performance of the MACE framework by using three different filtering strategies over all the datasets. The $y$-axis shows the accuracy of MACE normalized by the respective optimum obtained from the same initial pool $\mathbf{\Gamma}$. All dots are different experiments with each from a different starting pool.




(a) UrbObs (b) Swissex (c) ThingSpeak

Fig. 6. Relative performance of the MACE framework by using three different filtering strategies over all the datasets. The $y$-axis shows the percentile of MACE against all the other possible combinations obtained from the same initial pool $\mathbf{\Gamma}$. All dots are different experiments each with a different starting pool.

tuning, compared to all the possible alternatives. For instance, a value of $75\%$ means that, for that initial pool $\mathbf{\Gamma}$, the accuracy of the heuristic ensemble sits in the $75^{th}$ percentile of the accuracies of all the ensembles obtainable from $\mathbf{\Gamma}$ (*i.e.* $75\%$ of them are worse than the MACE ensemble). For UrbObs, more values are distributed in the topmost part, averaging around 80% and spanning down to 50%, which is highly positive. In fact, according to Figure 5, we observe a maximum error of 4% compared to the maximum. This suggests that, when the percentile of MACE is low, most probably accuracies are distributed close to each other and, even if many others are better, the difference is negligible. In the case of Swissex, we notice a similar behavior, with slightly worse overall performances. It is however interesting to notice how, for this dataset, distribution-based strategies tend to perform worse than rank-based ones. ThingSpeak, instead, displays highly positive results, as the average percentile of MACE is around 90% with the exception of the Top-$k$ strategy. In fact, we notice how SoF in this case outperforms the other strategies both in absolute and relative evaluations. Furthermore, we observe that, even though in Figure 5(c) sometimes the heuristic accounts for $85\%$ the accuracy of the optimum, we still find it to be ranked as one of the top combinations, according to Figure 6(c).

## VI. Conclusion and Future Works

In this paper, we proposed a novel algorithmic framework called MACE to tackle the challenge of annotation and classification of open IoT datastreams produced from heterogeneous IoT environments. First, we collected significant datasets from the literature and released a new comprehensive dataset extracted from ThingSpeak that composes our experiments at open IoT scales of *city-wide*, *region-wide* and *country-wide*. Through experimental evaluations of a number of well-known classifiers both in the scope of TSC and BOS we observed that, although IoT datastreams are reminiscent of time series datasets, due to the heterogeneity in the observations produced by IoT devices, classic TSC approaches perform poorly while vanilla classifiers based on statistical features perform significantly better when considering the numerical characteristics of the IoT datastream. Secondly, we proposed MACE, which uses a novel cascading ensemble approach to take advantage of different domains and dimensions of the IoT data such as 1) available textual metadata, 2) statistical characteristics and 3) numerical data points. MACE heuristically selects and orders classifiers in a pipeline in order to optimize the classification performance. Through extensive experimental evaluations and comparisons with state-of-the-art approaches in the literature, we validated the significant gain in accuracy of the proposed MACE algorithm with diverse filtering strategies.

### *Current Limitations and Future Directions*

The approach proposed in this paper focuses in particular on Open Data scenarios with their own distinctive features and assumptions. We believe that MACE could be extended to generic IoT scenarios as well, in order to concur in building efficient future autonomic systems [16], however, this implies

tackling additional issues that the current MACE implementation is not focused upon or not entirely able to cope with.

IoT ecosystems are known to suffer from data mislabeling, sometimes intentional [34], while Open Data is generally trusted to be much less problematic. Since MACE is based on supervised learning, data mislabeling would negatively impact its performance. Especially when mislabeling or tampering occurs at a large scale, the supervised learning model essentially learns from noises and yields arbitrary predictions. Tackling mislabeling problems requires a whole separate study, as label noise types and tampering patterns are diverse (e.g., label noise can severely harm decision tree-based algorithms). Some existing studies [12] outlined label noise-robust, label noise-tolerant, and label noise-cleansing algorithms. The cleansing approaches appear to be easier for integration with MACE (e.g., cleaning training data via anomaly detection, nearest neighbors clustering, etc.), however, the application of these methods deserves a separate and rigorous evaluation.

Open Data itself is not subject to privacy concerns, however, private IoT environments, in order to permit data analysis by third parties, at times need to undergo a phase of metadata obfuscation. It is also the case of data regulation enforcement, as it happens for instance for GDPR [9]. The regulation step occurs prior to the classification phase, and while MACE can deal with partial/inaccurate IoT metadata that could be impacted by privacy regulations, a further in-depth study on their consequences should be performed. As a matter of fact, metadata inaccuracies enforced by GDPR are different from the ones that genuinely appear in the Open Data, so MACE can only deal with them to a certain extent. For instance, if a privacy regulation implies that data should be encrypted, the classification of MACE over encrypted data will not succeed and a whole different approach [7] is needed. We believe that a future study on these issues, though challenging, could give a significant usability boost to MACE in the modern era.

## References


[1] Alpaydin, E., Kaynak, C.: Cascading classifiers. Kybernetika **34**(4), 369–374 (1998)

[2] Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **31**(3), 606–660 (2017)

[3] Batista, G.E., Wang, X., Keogh, E.J.: A complexity-invariant distance measure for time series. In: Proceedings of the 2011 SIAM international conference on data mining. pp. 699–710. SIAM (2011)

[4] Biglari, M., Soleimani, A., Hassanpour, H.: A cascaded part-based system for fine-grained vehicle classification. IEEE Transactions on Intelligent Transportation Systems **19**(1), 273–283 (2017)

[5] Biglari, M., Soleimani, A., Hassanpour, H.: A cascading scheme for speeding up multiple classifier systems. Pattern Analysis and Applications **22**(2), 375–387 (2019)

[6] Borges Neto, J.B., Silva, T.H., Assunção, R.M., Mini, R.A., Loureiro, A.A.: Sensing in the collaborative Internet of Things. Sensors **15**(3), 6607–6632 (2015)

[7] Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. In: NDSS Symposium 2015. p. 04_1_2. Internet Society (2015)

[8] Calbimonte, J.P., Yan, Z., Jeung, H., Corcho, O., Aberer, K.: Deriving semantic sensor metadata from raw measurements. In: Proceedings of the 5th International Workshop on Semantic Sensor Networks at ISWC. pp. 33–48. CEUR-WS (2012)

[9] Chaudhuri, A.: Internet of things data protection and privacy in the era of the general data protection regulation. Journal of Data Protection & Privacy **1**(1), 64–75 (2016)

[10] Damerau, F.J.: A technique for computer detection and correction of spelling errors. Communications of the ACM **7**(3), 171–176 (1964)

[11] Ericsson: Internet of things forecast. https://www.ericsson.com/en/mobility-report/internet-of-things-forecast, accessed: 2018-04-30

[12] Frénay, B., Verleysen, M.: Classification in the presence of label noise: a survey. IEEE transactions on neural networks and learning systems **25**(5), 845–869 (2013)

[13] Gao, J., Bergés, M.: A large-scale evaluation of automated metadata inference approaches on sensors from air handling units. Advanced Engineering Informatics **37**, 14–30 (2018)

[14] Gao, J., Ploennigs, J., Berges, M.: A data-driven meta-data inference framework for building automation systems. In: Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments. pp. 23–32 (2015)

[15] García-Borroto, M., Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A.: Cascading an emerging pattern based classifier. In: Mexican Conference on Pattern Recognition. pp. 240–249. Springer (2010)

[16] Gill, S.S., Xu, M., Ottaviani, C., Patros, P., Bahsoon, R., Shaghaghi, A., Golec, M., Stankovski, V., Wu, H., Abraham, A., Singh, M., Mehta, H., Ghosh, S.K., Baker, T., Parlikad, A.K., Lutfiyya, H., Kanhere, S.S., Sakellariou, R., Dustdar, S., Rana, O., Brandic, I., Uhlig, S.: Ai for next generation computing: Emerging trends and future directions. Internet of Things **19**, 100514 (2022)

[17] Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 392–401. ACM (2014)

[18] Holmegaard, E., Kjærgaard, M.B.: Mining building metadata by data stream comparison. In: 2016 IEEE Conference on Technologies for Sustainability (SusTech). pp. 28–33. IEEE (2016)

[19] Hong, D., Wang, H., Whitehouse, K.: Clustering-based active learning on sensor type classification in buildings. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. pp. 363–372 (2015)

[20] Iddianozie, C., Palmes, P.: Towards smart sustainable cities: Addressing semantic heterogeneity in building management systems using discriminative models. Sustainable Cities and Society p. 102367 (2020)

[21] James, P.M., Dawson, R.J., Harris, N., Joncyzk, J.: Urban Observatory Environment. Newcastle University (2014). https://doi.org/10.17634/154300-19

[22] Jeong, Y.S., Jeong, M.K., Omitaomu, O.A.: Weighted dynamic time warping for time series classification. Pattern Recognition **44**(9), 2231–2240 (2011)

[23] Li, X., Lin, J.: Linear Time Complexity Time Series Classification with Bag-of-Pattern-Features. In: Data Mining (ICDM), 2017 IEEE International Conference on. pp. 277–286. IEEE (2017)

[24] Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Mining and knowledge discovery **15**(2), 107–144 (2007)

[25] Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems **39**(2), 287–315 (2012)

[26] Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N.O., Guarnizo, J.D., Elovici, Y.: Detection of unauthorized iot devices using machine learning techniques. arXiv preprint arXiv:1709.04647 (2017)

[27] Montori, F., Bedogni, L., Bononi, L.: A collaborative Internet of Things architecture for smart cities and environmental monitoring. IEEE Internet of Things Journal **5**(2), 592–605 (2018)

[28] Montori, F., Liao, K., Jayaraman, P.P., Bononi, L., Sellis, T., Georgakopoulos, D.: Classification and annotation of open internet of things datastreams. In: International Conference on Web Information Systems Engineering. pp. 209–224. Springer (2018)

[29] O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., Motwani, R.: Streaming-data algorithms for high-quality clustering. In: Data Engineering, 2002. Proceedings. 18th International Conference on. pp. 685–694. IEEE (2002)

[30] Ortiz, J., Crawford, C., Le, F.: Devicemien: network device behavior modeling for identifying unknown iot devices. In: Proceedings of the International Conference on Internet of Things Design and Implementation. pp. 106–117 (2019)

[31] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

[32] Rahman, A.F.R., Fairhurst, M.C.: Serial combination of multiple experts: A unified evaluation. Pattern Analysis & Applications **2**(4), 292–311 (1999)

[33] Rakthanmanon, T., Keogh, E.: Fast shapelets: A scalable algorithm for discovering time series shapelets. In: Proceedings of the 2013 SIAM International Conference on Data Mining. pp. 668–676. SIAM (2013)

[34] Restuccia, F., D'Oro, S., Melodia, T.: Securing the internet of things in the age of machine learning and software-defined networking. IEEE Internet of Things Journal **5**(6), 4829–4842 (2018)

[35] Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review **33**(1-2), 1–39 (2010)

[36] Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intelligent Data Analysis **11**(5), 561–580 (2007)

[37] Schäfer, P.: The BOSS is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery **29**(6), 1505–1530 (2015)

[38] Schubert, E., Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Dbscan revisited, revisited: why and how you should (still) use dbscan. ACM Transactions on Database Systems (TODS) **42**(3), 19 (2017)

[39] Siow, E., Tiropanis, T., Wang, X., Hall, W.: TritanDB: Time-series Rapid Internet of Things Analytics. arXiv preprint arXiv:1801.07947 (2018)

[40] Sivanathan, A., Gharakheili, H.H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., Sivaraman, V.: Classifying iot devices in smart environments using network traffic characteristics. IEEE Transactions on Mobile Computing **18**(8), 1745–1759 (2018)

[41] Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 947–956. ACM (2009)


**Federico Montori** received the B.S. and M.S. degrees (summa cum laude) in computer science and the Ph.D. degree in computer science and engineering from the University of Bologna, Italy, in 2012, 2015, and 2019, respectively. He was a Visiting Researcher at Swinburne University of Technology (Australia), Luleå Tekniska Universitet (Sweden), and Technische Universität Ilmenau (Germany). He is currently a Senior Assistant Professor at the University of Bologna. He participated in several EU projects and he is currently WP Leader for the H2020 Project Arrowhead Tools. His primary research interests include mobile crowdsensing (MCS), pervasive and mobile computing, IoT automation, and data analysis for IoT scenarios.

**Kewen Liao** holds a PhD degree in Computer Science from The University of Adelaide since 2014. He is currently a Senior Lecturer in Information Technology and a Director of the HilstLab at Australian Catholic University (ACU), Sydney. He did his postdocs at The University of Melbourne and Swinburne University of Technology. He has over 40 publications including from premier venues of ICDE, WSDM, WWW, CHI, IJCAI, CIKM etc. His research interests include data science, machine learning, and theoretical computer science.

**Matteo De Giosa** received his bachelor's degree (summa cum laude) in Information Science for Management from the University of Bologna and his master's degree (summa cum laude) in Data Science from the University of Milano-Bicocca in 2019 and 2021, respectively. He is currently pursuing a career as a Data Scientist.

**Prem Prakash Jayaraman** is the Head of the Digital Innovation Lab and a Full Professor at Swinburne University of Technology. Previously he was a Post Doctoral Research Scientist in the Digital Productivity and Services Flagship of Commonwealth Scientific and Industrial Research Organization (CSIRO – Australian Government's Premier Research Agency). He is broadly interested in the research areas of the Internet of Things, Mobile and Cloud Computing, Health Informatics, and the application of Data Science techniques and methodologies in real-world settings. He was a key contributor and one of the architects of the Open Source Internet of Things project (OpenIoT) that won the prestigious Black Duck Rookie of the Year Award in 2013 (https://github.com/OpenIotOrg/openiot). He is the recipient of Swinburne's Vice Chancellor's Team Award for Digital Innovation in 2018 and is the recipient of 2 best paper awards (IEA-AIE 2010 and HICSS 2016) and several hackathon challenges including, Unearthed Mining Hackathon 2015, Melbourne, The 4th International Conference on IoT (2014) at MIT media lab, Cambridge, MA and IoT Week 2014, London.

**Luciano Bononi** (M), (MSC, Summa cum laude, 1997, Ph.D., 2001), is a Full Professor of Computer Networks, Internet of Things, Wireless and Mobile Systems, and Mobile Applications at the Department of Computer Science and Engineering of the University of Bologna. He has co-authored more than 140 peer-reviewed conference and journal publications and 8 book chapters, receiving four best paper awards, and his research areas include wireless systems and networks, protocol architectures, Internet of Things, Internet of Energy, Smart Mobility, modeling, simulation, performance evaluation, mobile services, and mobile applications. He has been involved in more than 10 international research projects, and he is an Associate Editor of three international Journals and guest edited more than 10 special issues. He was chair in more than 15 IEEE/ACM conferences and TPC member in more than 150 IEEE/ACM conferences on the above research topics. He is the founder and director of the Laboratory of Wireless Systems and Mobile Applications at CSE.

**Timos Sellis** received the Ph.D. degree in computer science from the University of California, Berkeley, in 1986. He is the director of the Archimedes Research Unit on AI, Data Science, and Algorithms at Athena Research and Innovation Center (Greece). Till the end of 2012, he was the director of the Institute for Management of Information Systems (IMIS) and a professor at the National Technical University of Athens, Greece. Between 2013 and 2015, he was a professor at RMIT University, Australia, and between 2016 and 2020 a Professor at the School of Software and Electrical Engineering of Swinburne University of Technology in Australia, and the Director of Swinburne's Data Science Research Institute. His research interests include big data, data streams, graph data management, data integration, and spatio-temporal database systems. He is a fellow of the IEEE and the ACM. In 2018 he was awarded the IEEE TCDE Impact Award, in recognition of his impact in the field and for contributions to database systems research and broadening the reach of data engineering research.

**Dimitrios Georgakopoulos** is a full Professor of Computer Science in Swinburne University of Technology's Faculty of Science, Engineering & Technology, and the inaugural Director of the IoT Lab in the university's Digital Innovation Capability Platform. He also leads the Industry 4.0 program at Swinburne's Manufacturing Futures Research Institute. Before that, he served as Research Director (2008-2014) of CSIRO's ICT Centre and a Professor at RMIT University (2014-2016). Prior to joining CSIRO, he held research and management positions in several industrial laboratories in the USA, including Telcordia Technologies, Microelectronics and Computer Corporation (MCC) in Austin, Texas; GTE Laboratories in Boston, Massachusetts; and Bell Communications Research (Bellcore) in Piscataway, New Jersey. He authored/co-authored 190+ journal and conference publications in computer science, which include three seminal papers in the areas of Service Computing, Workflow Management, and Context Management in the Internet of Things (IoT).