Evolving Systems (2018) 9:95–118
https://doi.org/10.1007/s12530-017-9190-z

ORIGINAL PAPER

Predictive intelligence to the edge: impact on edge analytics

Natascha Harth[1] · **Christos Anagnostopoulos**[1] · **Dimitrios Pezaros**[1]

Received: 22 December 2016 / Accepted: 29 May 2017 / Published online: 28 August 2017

Abstract
We rest on the edge computing paradigm where pushing processing and inference to the edge of the Internet of Things (IoT) allows the complexity of predictive analytics to be distributed into smaller pieces physically located at the source of the contextual information. This enables a huge amount of rich contextual data to be processed in real time that would be prohibitively complex and costly to deliver on a traditional centralized Cloud. We propose a lightweight, distributed, predictive intelligence mechanism that supports communication efficient aggregation and predictive modeling within the edge network. Our idea is based on the capability of the edge nodes to (1) monitor the evolution of the sensed time series contextual data, (2) locally determine (through prediction) whether to disseminate contextual data in the edge network or not, and (3) locally re-construct undelivered contextual data in light of minimizing the required communication interaction at the expense of accurate analytics tasks. Based on this on-line decision making, we eliminate data transfer at the edge of the network, thus saving network resources by exploiting the evolving nature of the captured contextual data. We provide comprehensive analytical, experimental and comparative evaluation of the proposed mechanism with other mechanisms found in the literature over real contextual datasets and show the benefits stemmed from its adoption in edge computing environments.

Keywords Edge analytics · Predictive intelligence · Evolving data streams · Communication efficiency · Context prediction · Exponential smoothing

1 Introduction

Edge analytics (Satyanarayanan et al. 2015) is an approach to efficient contextual data analysis in which computation is performed on sensing devices (sensors, actuators, controllers, concentrators), network switches or other devices (concentrators) instead of transmitting the whole data to a centralized computing environment/Cloud. By sending all the data from billions of IoT devices to the cloud can overwhelm the existing infrastructure. To overcome these issues, Edge Computing (EC) (The mobile-edge computing initiative 2016; Stojmenovic and Wen 2014) is emerging bringing contextual data processing, networking, and analytics closer to the IoT devices and applications. EC represents a shift in which *intelligence* is pushed from the cloud to the edge, localizing certain kinds of analysis, e.g., aggregation operators over data streams, regression analyses, information inference and reasoning, and local decision-making (Yi et al. 2015). This enables quicker response times, unencumbered by network latency, as well as reduced traffic, by *intelligently processing and relaying the appropriate analyzed data*. Pushing analytics algorithms to IoT devices alleviates the processing strain on enterprise data management as the number of connected devices and the amount of data generated and collected increases (Vulimiri et al. 2015; Cheng et al. 2016).

✉ Christos Anagnostopoulos
christos.anagnostopoulos@glasgow.ac.uk

Natascha Harth
n.harth.1@research.gla.ac.uk

Dimitrios Pezaros
dimitrios.pezaros@glasgow.ac.uk

[1] School of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK

1.1 Motivation

In IoT environments contextual information sources are considered as continuous/evolving data streams (multivariate time series), where analytics tasks are applied to extract statistical dependencies, aggregate analyics, and infer new knowledge. Context-aware applications, crowdsensing applications (Ganti et al. 2011; Lane et al. 2010), environmental monitoring (Oliveira and Rodrigues 2011), forest monitoring (Awang and Suhaimi 2007; Zervas et al. 2011; Kang et al. 2013; Anagnostopoulos et al. 2016) (through unnamed vehicles), agriculture monitoring (Nittel 2009), road traffic monitoring, surveillance, video analytics (Simoens et al. 2013), marine environment monitoring (Xu et al. 2014), watershed monitoring systems (Eidson et al. 2009; Nguyen et al. 2010) over large-scale data streams require efficient, accurate and timely data analysis in order to facilitate (near) real-time decision-making, data stream mining, and situational context awareness (Kolomvatsos et al. 2016).

We abstract an edge network architecture through *edge nodes* forming a layer between *sensing/actuator nodes* and the cloud. Several Sensing and Actuator Nodes (SAN) are connected to each Edge Node (EN), e.g., cloudlet, sink node. Since ENs are located close to the SANs, contextual data should be intelligently transferred to them in real-time and in an energy efficient manner. Each SAN performs measurements and locally determines whether to transfer these measurements to the ENs or not in light of minimizing the required communication interaction (overhead) at the expense of accurate analytics tasks performed on the ENs. Based on this context, our idea rests on locally predict *whether* to disseminate sensed data or not within an edge network to achieve quality analytics by being communication efficient by exploiting the evolving nature of the captured contextual data and its reconstruction. However, this comes at the expense of the quality of analytics tasks. The fundamental requirement to materialize such predictive intelligence at the edge network is: (1) the autonomous nature of SANs to locally perform sensing and disseminate data under analytics quality-driven rules and (2) the capability of the ENs to locally perform lightweight data reconstruction and robust analytics tasks.

1.2 Literature review

A baseline approach for materializing analytics tasks on the cloud is simply all IoT devices to transmit the contextual data from all sensing nodes to certain sink nodes (back-end system). This has been realized in many previous studies (McConnell and Skillicorn 2005; Tulone and Madden 2006; Goel and Imielinski 2001; Anagnostopoulos and Triantafillou 2014, 2015, 2017a, b). In this case, analytics tasks are carried out by the back-end system on the cloud only, and not by the SANs or ENs at the edge of the network, despite their increasing computing capacity. Evidently, this solution, while practical, has many disadvantages, such as a high energy consumption incurred by transmitting the raw data to the cloud, the need for wireless link bandwidth, and high latency (Stojmenovic and Wen 2014). In the era of EC, instead, the desiderata are:

- *push* the analytics tasks close to the contextual data sources, i.e., to the ENs, which have to follow the evolution of the contextual data streams;
- *push* intelligence to SANs and ENs to *collaboratively* support edge analytics. ENs have to intelligently communicate with the SANs in an energy-efficient way, since communication efficiency is crucial to the prolonged lifetime of the edge network to support edge analytics.

We have distinguished two basic methodologies for edge analytics.

- *Distributed analytics* This methodology is based on the observation that the SANs and ENs create the possibility of analyzing and building (training) predictive analytics models in a distributed way. In this class of edge analytics, e.g., Simonetto and Leus (2014), Kejela et al. (2014) and Gemulla et al. (2011), contextual data and/or model's meta-data are circulated within the edge network, which evidently requires energy for data and meta-data dissemination adding extra communication overhead.
- *Group-based centralized analytics* This methodology refers to a group-based communication and single localized computation/processing scheme e.g., Anagnostopoulos et al. (2012, 2014, 2016); Anagnostopoulos and Hadjiefthymiades (2014); McConnell and Skillicorn (2005); Papithasri and Babu (2016); Manjeshwar and Agrawal (2001). Specifically, an EN is responsible for a group of SANs and maintains a set of historical contextual data of each SAN within the group. Such localized method is communication efficient due to the reduced length of routing path from SANs to the cloud. To support such type of edge analytics, energy is consumed on communication, i.e., sending and receiving data from SANs to the EN, and computation, i.e., ENs are processing local data. However, since the cost of local processing and analytics tasks is nontrivial, we should take into account the trade-off between intra-edge-network communication and localized computation (Jiang et al. 2011).

Both above mentioned basic methodologies are required to be efficient to support edge analytics in terms of computation and communication. The computational efficiency of the analytics tasks is a challenging research area, where recently distributed and large-scale statistical and machine learning algorithms emerge, e.g., Bottou et al. (2017); this is beyond the scope of this paper.

In this work, we depart from the mechanism of the *selective data delivery* (Jiang et al. 2011) and provide a generalization of this mechanism to be adopted on EC environments with the aim to support communication-efficient edge analytics. Our generalized mechanism relies on the principle of bounded-loss approximation at the ENs by the context prediction at the SANs. SANs locally decide on delivering contextual data to ENs based on local predictions, while ENs locally re-construct the contextual data given an approximation (re-construction) error bound. This error bound is controlled by the SANs.

Evidently, there is a trade-off, that we should pay attention, between contextual data communication and accuracy of analytics due to approximation/re-construction. On the one hand, by selectively transmitting contextual data increases the edge network life time and the available bandwidth, since less data are circulated. On the other hand, this comes at the expense of the quality of the predictive analytics tasks, due to local data re-construction at the ENs.

1.3 Research objectives and contribution

The research objectives of our generalized mechanism are: (1) ENs employ a mechanism to re-construct the undelivered contextual data by following the evolving nature of the data streams; (2) SANs are equipped with real-time context prediction (time series forecast) and data delivery decision. To secure an upper bound on the re-construction error at ENs (which plays a significant role in the quality of the aggregation and predictive analytics tasks), SANs control their local prediction/forecast error based on the principle of selective data delivery. This is achieved by *splitting* this predictive intelligence to SANs and ENs: the former nodes *locally predict* the expected data and *locally decide* on their delivery given a prediction error bound; the latter nodes *locally re-construct* the undelivered given a controlled re-construction error bound by the SANs. The mechanism is applied when SANs need to communicate with ENs and when ENs need to re-construct the un-delivered data before proceeding with the scheduled analytics task. Should the IoT applications tolerate certain *quality* in the derived analytics tasks, e.g., prediction accuracy, model fitting approximation error and misclassification error, our mechanism is proved to be communication efficient as shown in our Sect. 5.

To the best of our knowledge this is the first generalized mechanism that explores the potentials of predictive intelligence on the EC paradigm over evolving data streams. The key contributions in this paper are:

- We present a distributed, communication efficient predictive intelligence mechanism for local prediction and local re-construction within an edge network;
- We provide the theoretical prediction and re-construction error boundaries and their relationship;
- We provide a comprehensive sensitivity analysis of the basic parameters of our model and showcase the trade-off between accuracy (quality) of edge analytics (focusing on aggregation and multivariate linear regression) with communication overhead;
- We provide a comparative theoretical and experimental assessment with the selective data delivery mechanism (Manjeshwar and Agrawal 2001) where the EN neighborhood's formation is adopted from Papithasri and Babu (2016) in light of re-construction, aggregation, data prediction errors and communication overhead.
- We provide computational complexity analysis of our mechanism, which is higly computational efficient with $O(d)$ prediction and re-construction time over d-dimensional contextual data streams, and its relationship to the Autoregressive Integrated moving Average model (Muth 1960).
- We experiment with real contextual data from sensors and actuators networks.

1.4 Organization

The paper is organized as follows: in Sect. 2 we present our rationale and basic concept of the edge predictive intelligence formulated by certain definitions, preliminaries, and the fundamental metrics for evaluating our mechanism. Section 3 reports on the predictive intelligence split to the SAN and EN perspectives elaborating on certain policies for data delivery and re-construction. In Sect. 4 we provide a theoretical analysis of the prediction and re-construction error boundaries, the computational complexity of our mechanism and its relation to the linear forecasting model Autoregressive Integrated moving Average model (Muth 1960). In Sect. 5 we provide a sensitivity analysis of our mechanism with the basic model parameters, a theoretical and experimental comparative assessment with the selective data delivery mechanism (Manjeshwar and Agrawal 2001) and showcase the performance of the proposed mechanism with two real contextual datasets. Finally, Sect. 6 concludes the paper with future research agenda on edge analytics.

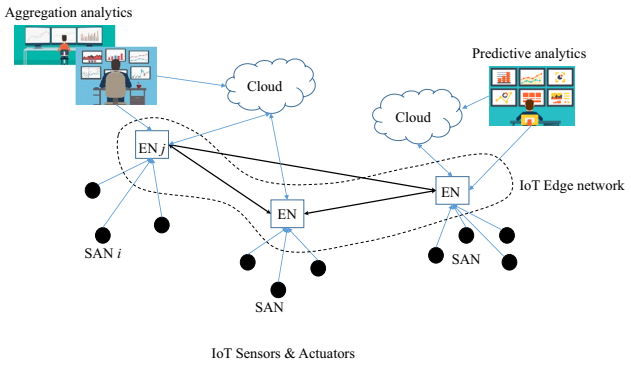

Fig. 1 The edge network with the ENs and the corresponding SANs provide communication efficient predictive modeling and analytics to end-users, analysts, and to IoT applications

2 Edge predictive intelligence

2.1 Rationale

We consider an edge network with connected ENs forming an arbitrary topology. Each EN j is connected with n_j SANs in a tree-like topology with root the EN and leaves its SANs as shown in Fig. 1. A SAN i is connected with its unique EN j and $\mathcal{N}_j = \{1, \ldots, n_j\}$ denotes the SAN set of the EN j, i.e., $i \in \mathcal{N}_j$.

A SAN i at every time instance $t = 1, 2, \ldots$ senses a d-dimensional row vector $\mathbf{x}_t = [x_{1t}, \ldots, x_{dt}] \in \mathbb{R}^d$ of contextual parameters, like temperature, humidity, sound, wind speed, air pollutant chemical compounds, etc. Hereinafter, we refer to $\mathbf{x}$ as *context vector*. The SAN i can communicate with its EN j in the edge network by transferring context vectors. To materialize the proposed predictive intelligence, the SAN i is equipped with a context vector prediction (time series forecasting) algorithm $f_i(\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-N})$, which uses the recent past $N \geq 1$ sensed context vectors stored in a sliding window $\mathcal{W}$ of size N to predict the context vector $\hat{\mathbf{x}}_t$ at time instance t. That is:

$$\hat{\mathbf{x}}_t = f_i(\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-N}) = f_i(\mathcal{W}) \tag{1}$$

and window $\mathcal{W} = (\mathbf{x}_{t-N}, \ldots, \mathbf{x}_{t-1})$. The SAN i, after sensing the context vector $\mathbf{x}_t$ at time t, locally predicts the predicted context vector $\hat{\mathbf{x}}_t$, thus, the *local prediction error* is:

$$e_t = d^{-\frac{1}{2}} ||\mathbf{x}_t - \hat{\mathbf{x}}_t|| \tag{2}$$

where $||\mathbf{x}|| = \left(\sum_{k=1}^{d} x_k^2\right)^{1/2}$ is the Euclidean norm of $\mathbf{x}$ and $d^{-1/2}$ is a normalization factor to ensure that $e_t \in [0, 1]$, given that the context vector $\mathbf{x} \in [0, 1]^d$ is scaled in the d-dimensional unit cube, i.e., each dimension $x_k, k = 1, \ldots, d$ is normalized (ranges) in [0, 1]. Such prediction capability yields the SAN able to decide whether to send context vectors $\mathbf{x}$ to its EN j or not for further processing. SAN i relies on a θ-based context vector delivery decision rule:

- **Case 1** If the predicted $\hat{\mathbf{x}}_t$ differs from the actual sensed $\mathbf{x}_t$ with respect to a *decision threshold* $\theta \in [0, 1]$, i.e., $e_t > \theta$, then the SAN i sends the actual $\mathbf{x}_t$ to the EN j.
- **Case 2** Otherwise, i.e., $e_t \leq \theta$, the SAN i does not send $\mathbf{x}_t$ to the EN j. In this case, the EN j is responsible for *reconstructing* a context vector locally for further processing.

In Case 1, the EN j receives the transmitted context vector $\mathbf{x}_t$ from SAN i. In Case 2, the EN j is equipped with a re-construction function

$$\tilde{\mathbf{x}}_t = g_j(\mathbf{u}_{t-1}, \ldots, \mathbf{u}_{t-M}) = g_j(\mathcal{W}) \tag{3}$$

of the recent $M \geq 1$ context vectors $\mathbf{u}$ from a sliding window $\mathcal{W} = (\mathbf{u}_{t-M}, \ldots, \mathbf{u}_{t-1})$ to locally predict (reconstruct) the undelivered vector $\mathbf{x}_t$, notated by $\tilde{\mathbf{x}}_t$ through historical context vectors. Specifically, the context vectors $\mathbf{u}$ in the EN's sliding window $\mathcal{W}_j$ correspond to either the actual received context vectors $\mathbf{x}$ from the SAN i (Case 1) or the past locally re-constructed context vectors $\tilde{\mathbf{x}}$ from g_j (Case 2), i.e.,

$$\mathbf{u}_t = \begin{cases} \mathbf{x}_t & \text{if } e_t > \theta \text{ (Case 1)} \\ \tilde{\mathbf{x}}_t = g_j(\mathcal{W}), & \text{otherwise; (Case 2)} \end{cases}$$

The *re-construction error (difference)* at the EN j is then:

$$a_t = \begin{cases} 0 & \text{Case 1,} \\ ||\mathbf{x}_t - \tilde{\mathbf{x}}_t|| & \text{Case 2.} \end{cases} \tag{4}$$

The sliding window at SAN i contains only actual (sensed) context vectors $\mathbf{x}$, while the sliding window at EN j contains either actual context vectors $\mathbf{x}$ (received from SAN i) or re-constructed context vectors $\tilde{\mathbf{x}}$ locally generated by EN j. The difference (norm) between the predicted context vector $\hat{\mathbf{x}}$ on SAN i and the reconstructed context vector $\tilde{\mathbf{x}}$ at EN j is $||\hat{\mathbf{x}} - \tilde{\mathbf{x}}|| = ||\mathbf{e} - \mathbf{a}||$, with $\mathbf{a} = \tilde{\mathbf{x}} - \mathbf{x}$ and $\mathbf{e} = \hat{\mathbf{x}} - \mathbf{x}$. This difference is zero when both the predictor and the re-constructor on SAN i and EN j, respectively, result in the same *error*. Overall, when $e_t > \theta$, the reconstruction difference $a_t = 0$, while when $e_t \leq \theta$, the reconstruction difference $a_t \geq 0$. For an analysis on the reconstruction error and prediction error, please refer to Sect. 4.

Given a decision threshold $\theta \in (0, 1)$ at SAN i, we study the performance of certain predictive analytics tasks on EN j. We qualitatively derive sufficient conditions for this and reveal that the decision is a function of both the desired error bound and the correlation among the sensed

contextual data values. When the decision threshold is very tight or the correlation is not significant, the SAN i always has to send its context vectors to the EN j. Due to the characteristics and inherent dynamics of the SANs' contextual data, when the underlying data streams distribution evolves over time, prediction / forecasting techniques may not work efficiently for a set of less predictable contextual data. Moreover, there might be correlations among contextual data from neighboring SANs (data locality in $\mathcal{N}_j$), thus, the EN j is capable of learning those statistical correlations in a communication-efficient way, as will be shown later. We provide certain definitions and preliminaries before elaborating on our distributed intelligence mechanism.

2.2 Definitions and preliminaries

Definition 1 (*Sliding window*) A sliding window $\mathcal{W}$ is specified by a fixed-size temporal extent $N > 0$ ('horizon') by appending new context vectors and discarding older ones on the basis of their appearance.

For instance, at time t, a sliding window $\mathcal{W}$ is a sequence of all context vectors observed from $t - N$ to $t - 1$, i.e., $\mathcal{W} = (\mathbf{x}_{t-N}, \mathbf{x}_{t-N+1}, \ldots, \mathbf{x}_{t-1})$. As an example, an analytics query over $\mathcal{W}$ could be: 'continuously return all context vectors of the past hour, i.e., N=60 min'. The sliding window is the most widely used in continuous aggregation and fusion analytics functions (Dallachiesa et al. 2015; Patroumpas and Sellis 2011; Abadi et al. 2003, 2005).

The *aggregation analytics* tasks are evaluated over the contents of a window $\mathcal{W}$. The aggregated results change over time as the window slides. We use the classification from Gray and Chaudhuri (1997) that divides aggregation functions into three categories: distributive, algebraic, and holistic. Let $\mathcal{W}$, $\mathcal{W}_1$, and $\mathcal{W}_2$ be windows. An aggregation analytics function $h : \mathcal{W} \rightarrow \mathbb{R}^d$ is distributive if $h(\mathcal{W}_1 \cup \mathcal{W}_2)$ can be computed from $h(\mathcal{W}_1)$ and $h(\mathcal{W}_1)$ for all $\mathcal{W}_1$, $\mathcal{W}_2$. An aggregation analytics function h is algebraic if there exists a 'synopsis function' σ such that for all $\mathcal{W}$, $\mathcal{W}_1$, and $\mathcal{W}_2$: (1) $h(\mathcal{W})$ can be computed from $\sigma(\mathcal{W})$; (2) $\sigma(\mathcal{W})$ can be stored in constant memory; and (3) $\sigma(\mathcal{W}_1 \cup \mathcal{W}_2)$ can be computed from $\sigma(\mathcal{W}_1)$ and $\sigma(\mathcal{W}_2)$. An aggregation analytics function h is holistic if it is not algebraic. Among the standard aggregates, `MAX` and `MIN` are distributive, `AVG` is algebraic, since it can be computed from a synopsis containing `SUM` and `COUNT`, and `QUANTILE`, `MEDIAN` are holistic.

Example 1 We can define the `AVG` and `MAX` analytics functions: $h^{avg}(\mathcal{W}) = \frac{1}{N}\sum_{k=t-N}^{t} \mathbf{x}_k$ and $h^{max}(\mathcal{W}) = [\max\{x_{1k}\}, \ldots, \max\{x_{dk}\}]_{k=t-N}^{t}$, respectively.

In our case, the aggregation analytics function h is running on EN j *for each* sliding window $\mathcal{W}$ containing M received and/or re-constructed context vectors from the SAN $i \in \mathcal{N}_j$ depending on Case 1 and Case 2. Note that such functions are built-in constructs in IoT-application specific continuous analytics queries.

Example 2 The aggregation analytics query 'every minute find the average temperature and the maximum humidity over context streams 'temperature' and 'humidity' collected during the past hour' in Continuous Query Language (Arasu et al. 2006) involving `AVG` and `MAX` operators in a sliding window $\mathcal{W}$, $N = 60$ min can be expressed as follows:

```
SELECT AVG(temperature), MAX(humidity)
FROM Context Streams [RANGE 60 MINUTES
SLIDE 1 MINUTE]
```

Note, typical progressive aggregates like `SUM`, `MIN` and `AVG` requires constant time $O(1)$ per value since there is no need to scan the entire window (Patroumpas and Sellis 2006, 2010). However, more advanced aggregation analytics functions like outliers detection or concept drift detection in a sliding window $\mathcal{W}$ require multiple scanning of the $\mathcal{W}$. Aggregation analytics functions can be also combined on a EN to infer certain events that might trigger decision making.

Example 3 Consider the evaluation of a situational context (localized event stream processing) for the past 10 min as the activation of the following rule with conjunctive predicates associated with `AVG` and `MAX` aggregation analytics functions over 'temperature' and 'wind-speed' sliding windows from two corresponding SANs:

```
EVENT := IF AVG(temperature) ≥ 90 AND
MAX(wind-speed) ∈ [10,20] WITHIN 10 min-
utes THEN ACTION is 'warning'
```

Definition 2 (*Aggregation analytics difference*) Consider an EN j and its SAN $i \in \mathcal{N}_j$. The aggregation analytics difference β_i between the analytics result on EN j derived from aggregation function h over the window $\mathcal{W}$ in the EN j and the *actual* analytics result derived from h over the window $\mathcal{W}^*$, which contains only the actual context vectors from SAN i to EN j (ground truth) is:

$$\beta_i = ||h(\mathcal{W}) - h(\mathcal{W}^*)||. \quad (5)$$

The aggregation analytics difference β_i denotes how much the aggregation results over the window $\mathcal{W}$ on ED j with context vectors $\mathbf{u}$ *differ* from the aggregation results over the window $\mathcal{W}^*$ with context vectors $\mathbf{x}$, should SAN i have sent all context vectors to ED j. Obviously, if we encounter only the Case 1, then $\beta_i = 0, \forall i \in \mathcal{N}_j$. Now,

since we allow SAN i to decide on sensing context vectors w.r.t. θ and EN j being able to re-construct undelivered context vectors, then $\beta_i \geq 0$. The concept is how much an IoT application tolerates this difference in analytics results in light of communication efficiency in the edge network.

One of the most important predictive models for predictive analytics is the multivariate linear regression approximation (Kuhn and Johnson 2013). Consider an EN j and its SAN i, $i \in \mathcal{N}_j$. The SAN i generates context vectors $\mathbf{x}_t = [\mathbf{x}_t^{in}, y_t^{out}] \in \mathbb{R}^d$, represented as input–output pairs $(\mathbf{x}_1^{in}, y_1^{out}), \ldots, (\mathbf{x}_T^{in}, y_T^{out}) \in \mathbb{R}^{d-1} \times \mathbb{R}, T > 0$. The EN j either receives or re-constructs context vectors from SAN i, i.e., $\mathbf{u}_t = [\mathbf{u}_t^{in}, z_t^{out}] \in \mathbb{R}^d$. The objective of a *linear regression analytics* task on EN i is to estimate a coefficient vector $\mathbf{w}_i \in \mathbb{R}^d$, which interprets the statistical dependency of input $\mathbf{u}^{in}$ with output z^{out} which minimizes the objective function:

$$\mathcal{J}(\mathbf{w}_i) = \min_{\mathbf{w}_i \in \mathbb{R}^d} \frac{1}{T} \sum_{t=1}^{T} \left(z_t^{out} - (\mathbf{u}_t^{in})^\top \mathbf{w}_i\right)^2 + \lambda ||\mathbf{w}_i||^2 \tag{6}$$

where λ is a regularization parameter. In our mechanism, the coefficient vector $\mathbf{w}_i$ *approximates* the actual coefficient vector $\mathbf{w}_i^*$, which refers to the statistical dependency of the actual context vectors $\mathbf{x}$ in SAN i, due to Case 2 w.r.t. θ. Obviously, if Case 2 never occurs, then $\mathbf{w}_i \equiv \mathbf{w}_i^*$, but then SAN i sends *all* context vectors to EN j, where the latter trains the linear regression model. In other words, the actual coefficient vector $\mathbf{w}_i^*$ minimizes the objective function:

$$\mathcal{J}(\mathbf{w}_i^*) = \min_{\mathbf{w}_i^* \in \mathbb{R}^d} \frac{1}{T} \sum_{t=1}^{T} \left(y_t^{out} - (\mathbf{x}_t^{in})^\top \mathbf{w}_i^*\right)^2 + \lambda ||\mathbf{w}_i^*||^2 \tag{7}$$

Example 4 Consider SAN i with context vector $\mathbf{x} = [x_1, x_2, x_3]$ referring to the contextual parameters humidity, wind speed and temperature. The corresponding EN j is responsible for learning the statistical dependency $\mathbf{w}_i$ between temperature (dependent variable $y^{out} = x_3$) with humidity and wind speed (independent variables $\mathbf{x}^{in} = [x_1, x_2]$).

Moreover, the regression analytics task on EN j is applied from context vectors coming from different SANs.

Example 5 Consider the SAN i and SAN ℓ with $i, \ell \in \mathcal{N}_j$ sensing context vectors $\mathbf{x}_i = [x_{i1}, x_{i2}]$ and $\mathbf{x}_\ell = [x_{\ell 1}, x_{\ell 2}, x_{\ell 3}]$, respectively. The EN j is responsible, e.g., for learning the linear dependency $y^{out} = x_{i2}$ and $\mathbf{x}^{in} = [x_{\ell 1}, x_{\ell 2}]$ between the contextual parameters from those SANs in $\mathcal{N}_j$.

Given a set of M actual context vectors $\mathbf{x}_m = [\mathbf{x}_m^{in}, y_m^{out}]$, $m = 1, \ldots, M$, the predicted outputs $\hat{z}_m^{out}$ provided by the approximated regression model at EN j is $\hat{z}_m^{out} = (\mathbf{x}_m^{in})^\top \mathbf{w}_i$, thus, and the corresponding root mean squared error (RMSE) is:

$$\epsilon_i = \left(\frac{1}{M} \sum_{m=1}^{M} \left(y_m^{out} - \hat{z}_m^{out}\right)^2\right)^{1/2} \tag{8}$$

Similarly, the predicted outputs $\hat{y}_m^{out}$ provided by the *actual* regression model $\mathbf{w}_i^*$, i.e., trained by the actual context vectors $\mathbf{x}$, is $\hat{y}_m^{out} = (\mathbf{x}_m^{in})^\top \mathbf{w}_i^*$ and the corresponding RMSE is:

$$\epsilon_i^* = \left(\frac{1}{M} \sum_{m=1}^{M} (y_m^{out} - \hat{y}_m^{out})^2\right)^{1/2} \tag{9}$$

Definition 3 (*Regression analytics difference*) Consider an EN j and its SAN $i \in \mathcal{N}_j$.The regression analytics difference γ_i is defined as the absolute difference of the RMSE ϵ_i derived from the approximated regression line (coefficient vector $\mathbf{w}_i$) and the RMSE ϵ_i^* derived from the actual regression line (actual coefficient vector $\mathbf{w}_i^*$) trained by the *actual* SAN i's context vectors:

$$\gamma_i = |\epsilon_i - \epsilon_i^*|. \tag{10}$$

The RMSE ϵ_i^* is the linear regression error we obtain over the actual training pairs $(\mathbf{x}^{in}, y^{out})$ since $\mathbf{w}_i^*$ is the actual regression coefficient. However, since EN j may not receive the actual pairs all the time due to Case 2, then the derived regression coefficient $\mathbf{w}_i$ results to a RMSE $\epsilon_i \neq \epsilon_i^*$. We require to tolerate a low γ_i difference by being communication-efficient in the edge network.

Statistical learning analytics, like the discussed linear regression analytics, that have local computation are suited for the EC paradigm. Therefore, the regression learning task should be iterative in nature, which processes a single training pair at a time. Since the computation is carried out on the EN, the training algorithm should be lightweight and robust. The optimization algorithm suitable for these cases are based on the method of online learning (Bottou 2010) and the Stochastic Gradient Descent (SGD) is the most prominent among them. In this context, the EN j incrementally updates the coefficient vector $\mathbf{w}_{i,t}$ at time instance t by moving a small step size (learning rate) $\eta \in (0, 1)$ along the negative gradient of the minimization function in Eq. (6) as shown in Algorithm 1. The training algorithm converges when there is no significant improvement of the $\mathbf{w}_t$ coefficient, i.e., when $||\Delta \mathbf{w}_{i,t}|| \leq \delta$, given a convergence threshold $\delta > 0$.

Input: Learning rate η, convergence threshold δ
Output: Regression coefficient $\mathbf{w}_i$
while $\|\Delta \mathbf{w}_t\| > \delta$ **do**
  Get either actual context vector $\mathbf{x}_t$ from SAN i or locally re-construct $\tilde{\mathbf{x}}_t$;
  Set the training pair $\mathbf{u}_t = (\mathbf{u}_t^{in}, z_t^{out})$;
  Update $\mathbf{w}_{i,t+1} = \mathbf{w}_{i,t} - \eta \frac{\partial \mathcal{J}}{\partial \mathbf{w}_{i,t}}$
end

Algorithm 1: Stochastic Gradient Descent Algorithm in EN j for SAN i.

Hence, given a decision threshold $\theta > 0$, our aim is to examine the impact of our predictive intelligence mechanism on (1) the re-construction difference a, (2) the aggregation analytics difference β, and (3) the regression analytics difference γ in light of communication efficiency by saving significant network bandwidth.

3 Distributed predictive intelligence

The intelligence of the proposed mechanism is split into two parts: (1) the SAN's intelligence with respect to the local prediction algorithm f_i following the evolving nature of the data streams and (2) the EN's intelligence with respect to the local re-construction algorithm g_j that supports the analytics tasks introduced in Sect. 2.2.

3.1 Sensor–actuator node intelligence

Consider a SAN i and let us elaborate on the first part. Very complex prediction models are not practical in the discussed EC paradigm due to the limited (energy-constrained) computational capacity of the SANs. Fortunately, simple linear predictors are sufficient to capture the temporal correlation of realistic contextual data as shown by previous studies (Chu et al. 2006; Chowdappa et al. 2015; Anagnostopoulos et al. 2010). A sliding window-based linear predictor is one of popular approaches to predicting the future based on past N measurements.

In this work, we are seeking to reduce the computational power for prediction and to use a small fraction of the SAN's computing power by adopting a predictive function with low complexity and computational effort. Multivariate exponential smoothing, used for time series forecast, is an ideal predictor adopted in our case, as its computational complexity is $O(d)$ in a d-dimensional space (elaborated in Sect. 4). A simple exponential smoothing weighs the current sensed context vector $\mathbf{x}_t$ and the historic context vectors (Durbin and Koopman 2012). This simple smoothing function is adopted as the prediction function f_i for the θ-based decision making.[1]

[1] Double exponential smoothing (Holt–Winters time series smoothing) could be adopted dealing with the same computational complexity.

At each time t, a smoothed context vector $\mathbf{s}_t$ is calculated by using the current sensed context vector $\mathbf{x}_t$ and the previous smoothed vector $\mathbf{s}_{t-1}$, i.e.,

$$\mathbf{s}_t = \alpha \mathbf{x}_t + (1-\alpha)\mathbf{s}_{t-1} \tag{11}$$

initializing with $\mathbf{s}_0 = \mathbf{x}_0$. The relationship between the history of the measured data and the current data is represented by $\alpha \in [0,1]$. A higher α denotes more importance to the current values and less importance to the historic values. Normally, $\alpha = 0.7$ (Durbin and Koopman 2012). The calculated smoothed vector $\mathbf{s}_{t-1} = [s_{1,t-1}, \ldots, s_{d,t-1}]$ refers to the predicted context vector $\hat{\mathbf{x}}_t$, that is:

$$\hat{\mathbf{x}}_t = f_i(\mathcal{W}) = \mathbf{s}_{t-1}, \tag{12}$$

with the window $\mathcal{W} = (\mathbf{s}_{t-1})$ at SAN i containing only the recent smoothed context vector. Hence, the complexity of f_i is $O(d)$; we require d computations for smoothing the $\mathbf{s}_t$ in Eq. (11) at time instance t. The forwarding decision of the actual $\mathbf{x}_t$ to the EN j depends whether the prediction error $e_t = ||\mathbf{s}_{t-1} - \mathbf{x}_t||$ exceeds the threshold θ.

3.2 Edge node intelligence

On the other side, the EN j, at time instance t either receives $\mathbf{x}_t$ (Case 1) or nothing (Case 2). In Case 1, the EN j simply inserts the delivered $\mathbf{x}_t$ into its corresponding window $\mathcal{W}$ (which is associated with the SAN $i \in \mathcal{N}_j$) discarding the oldest context vector, i.e., $\mathbf{u}_t = \mathbf{x}_t$. In Case 2, the EN j encounters an undelivered vector problem, since there is nothing to push in the sliding window $\mathcal{W}$. Such undelivered context vectors must be re-constructed with the available context vectors $\mathbf{u}$ reside currently in the $\mathcal{W}$ at EN j. In order to achieve this, we propose three re-construction policies, i.e., variants of the re-construction function $g_j(\mathcal{W})$. We should stress that, we require a computationally efficient re-construction function on EN j, thus, being relatively a small overhead compared to the analytics tasks. Those policies are introduced below.

Policy 1 This policy, in Case 2, uses the *most* recent context vector from $\mathcal{W}$ at EN j, i.e., the first element of the sliding window, as the re-constructed context vector. Therefore, the re-constructed context vector is inserted into the $\mathcal{W}$ and the oldest context vector from the window is discarded. Note

that, after this insertion, there are two duplicates of the most recent context vector in the window. There might be also the case where the entire window (of length N) would have contained the same context vector if the SAN i had not sent a context vector in the last N time instances. This denotes that, during this recent history of N time instances, the maximum difference of the sequentially sensed context vectors measured on SAN i is less that θ. In this case, it is redundant to send *similar* context vectors to EN j given a threshold θ. In Case 1, the EN j simply inserts the delivered $\mathbf{x}_t$ into the window and discards the oldest context vector. Policy 1 at EN j is provided by the Algorithm 2.

Policy 3 This policy applies the exponential smoothing algorithm (discussed above) for re-constructing the undelivered context vector in the EN j. In Case 1, the EN j simply inserts the delivered $\mathbf{x}_t$ into the window and discards the oldest context vector. Moreover, after this insertion, the EN j calculates the smoothing context vector $\mathbf{s}'_t$ based on the delivered $\mathbf{x}_t$ and the previously calculated smoothed context vector, i.e.,

$$\mathbf{s}'_t = \alpha \mathbf{x}_t + (1-\alpha)\mathbf{s}'_{t-1}.$$

In Case 2, this policy re-constructs the $\tilde{\mathbf{x}}_t$ with the recently smoothed context vector $\mathbf{s}'_{t-1}$ (exploiting the context vector

Data: Sliding windows $\mathcal{W}_{ij}$ at EN j for each connected SAN i.
Result: Reconstruction context vectors at EN j for each connected SAN i
for *SAN* $i \in \mathcal{N}_j$ **do**
  if $e_{i,t} > \theta_i$ **then**
    SAN i sends $\mathbf{x}_i$ to EN j;
    $\mathbf{u}_t = \mathbf{x}_t$;
  else
    `/* reconstruction by using the last context vector */`
    $\mathbf{u}_t = \mathbf{u}_{t-1}$;
  end
  `/* update the sliding window */`
  $\mathcal{W}_{ij} = \mathcal{W}_{ij} \setminus \{\mathbf{u}_{t-N}\}$;
  $\mathcal{W}_{ij} = \mathcal{W}_{ij} \cup \{\mathbf{u}_t\}$;
end

Algorithm 2: Algorithm of Policy 1.

Policy 2 This policy, in Case 2, re-constructs the undelivered context vector $\tilde{\mathbf{x}}_t$ as the average vector of the current context vectors in the window $\mathcal{W}$, i.e.,

$$\tilde{\mathbf{x}}_t = g_j(\mathcal{W}) = \frac{1}{N}\sum_{k=t-N}^{t-1} \mathbf{u}_k.$$

This re-constructed value is then inserted into the window discarding the oldest one. In Case 1, the EN j simply inserts the delivered $\mathbf{x}_t$ into the window and discards the oldest context vector. Policy 2 at EN j is provided by the Algorithm 3.

delivery in Case 1) and discards the oldest context vector from the window. Note that, the series of the smoothed vectors $\mathbf{s}'_t$ in EN j is not the same with the series of the smoothed vectors $\mathbf{s}_t$ in SAN, since the vectors $\mathbf{s}'_{t-1}, \mathbf{s}'_{t-2}, \ldots$ are calculated by the $\mathbf{u}_{t-1}, \mathbf{u}_{t-2}, \ldots$ vectors from the window $\mathcal{W}_i$ on EN j. Moreover, in Case 2, after the re-construction of $\tilde{\mathbf{x}}_t$ with $\mathbf{s}'_{t-1}$, the smoothed context vector for time instance t is $\mathbf{s}'_t = \alpha\tilde{\mathbf{x}}_t + (1-\alpha)\mathbf{s}'_{t-1} = \mathbf{s}'_{t-1}$. Overall, Policy 3 at the EN j has as follows:

Data: Sliding windows $\mathcal{W}_{ij}$ at EN j for each connected SAN i.
Result: Reconstruction context vectors at EN j for each connected SAN i
for *SAN* $i \in \mathcal{N}_j$ **do**
  if $e_{i,t} > \theta_i$ **then**
    SAN i sends $\mathbf{x}_i$ to EN j;
    $\mathbf{u}_t = \mathbf{x}_t$;
  else
    `/* reconstruction based on the local current average (centroid)`
    `context vector */`
    $\mathbf{u}_t = \frac{1}{N}\sum_{k=t-N}^{t-1} \mathbf{u}_k$;
  end
  `/* update the sliding window */`
  $\mathcal{W}_{ij} = \mathcal{W}_{ij} \setminus \{\mathbf{u}_{t-N}\}$;
  $\mathcal{W}_{ij} = \mathcal{W}_{ij} \cup \{\mathbf{u}_t\}$;
end

Algorithm 3: Algorithm of Policy 2.

$$\begin{cases} \mathbf{s}'_t = \alpha \mathbf{x}_t + (1-\alpha)\mathbf{s}'_{t-1}, & \text{Case 1} \\ \tilde{\mathbf{x}}_t = \mathbf{s}'_{t-1} \text{ and } \mathbf{s}'_t = \mathbf{s}'_{t-1}, & \text{Case 2.} \end{cases} \tag{13}$$

Policy 3 at EN j is provided by the Algorithm 4.

the evolution of the difference of the predicted context vector $\hat{\mathbf{x}}_t$ at SAN i with the reconstructed context vector $\tilde{\mathbf{x}}_t$ at EN j based on Case 2; this depends on the decision threshold θ. Then, we obtain that:

$$\xi_t = \begin{cases} 0, & e_t > \theta \\ ||\hat{\mathbf{x}}_t - \tilde{\mathbf{x}}_t||, & e_t \leq \theta \end{cases} \tag{14}$$

Data: Sliding windows $\mathcal{W}_{ij}$ at EN j for each connected SAN i.
Result: Reconstruction context vectors at EN j for each connected SAN i
for *SAN* $i \in \mathcal{N}_j$ **do**
  if $e_{i,t} > \theta_i$ **then**
    SAN i sends $\mathbf{x}_i$ to EN j;
    /* updating the local smoothed context vector with the received context vector */
    $\mathbf{s}'_t = \alpha \mathbf{x}_t + (1-\alpha)\mathbf{s}'_{t-1}$;
  else
    /* reconstruction by one-step ahead forecasting */
    $\mathbf{u}_t = \mathbf{s}'_{t-1}$;
    /* update the local smoothed vector */
    $\mathbf{s}'_t = \mathbf{s}'_{t-1}$;
  end
  /* update the sliding window */
  $\mathcal{W}_{ij} = \mathcal{W}_{ij} \setminus \{\mathbf{u}_{t-N}\}$;
  $\mathcal{W}_{ij} = \mathcal{W}_{ij} \cup \{\mathbf{u}_t\}$;
end

Algorithm 4: Algorithm of Policy 3.

4 Theoretical analyses

4.1 Prediction error and reconstruction difference analysis

In this section, we analyze the relation of the local prediction error e_t at SAN i and its corresponding re-construction difference a_t at EN j provided by a re-construction algorithm (policy) $g_j(\mathcal{W})$. The aim of this analysis is to demonstrate the evolving feature of our distributed mechanism to follow the contextual data streams on the SANs and then to decide on their delivery or not to their corresponding ENs. Specifically, we analyze the evolution of the local conditional expectation of the prediction error $\mathbb{E}[e_t | e_t \leq \theta]$ conditioned on the event (Case 2): $\{e_t \leq \theta\}$ and its relation with the expected reconstruction difference $\mathbb{E}[a_t]$. The idea is that our mechanism 'remotely' monitors the evolution of the reconstruction difference in EN j by experiencing a local prediction error at SAN i. This provides us with further insight on the upper bound of the reconstruction difference, based on the local prediction (forecasting) error. The derived knowledge of this relation provides us further insights on the adopted policy (Policy 1, 2, or 3) at the EN j, thus, being able to adapt to different policies based on the evolution of the data streams.

Let us focus on the pair of nodes: SAN i and EN j, where SAN i is adopting a prediction algorithm $\hat{\mathbf{x}} = f_i(\mathcal{W}_i)$ and EN j is adopting a reconstruction algorithm $\tilde{\mathbf{x}} = g_j(\mathcal{W}_j)$. Furthermore, consider the time series ξ_t, which monitors

Based on the definition of this time series, we identify two cases corresponding to the predicted and reconstructed vectors conditioned on the event $\{e_t \leq \theta\}$:

1. Case A: $\hat{\mathbf{x}}_t = \tilde{\mathbf{x}}_t$. In this case, the predicted context vector at SAN i is the same as the reconstructed context vector at EN j, e.g., SAN and EN are adopting the same algorithms for prediction and reconstruction.
2. Case B: $\hat{\mathbf{x}}_t = \tilde{\mathbf{x}}_t + \boldsymbol{\rho}_t$, where $\boldsymbol{\rho}_t$ is the *vector discrepancy* of the predicted context vector and the reconstructed context vector, given that $e_t \leq \theta$ at SAN i, with $\mathbb{E}[||\boldsymbol{\rho}||] < \infty$.

Consider the Case A.

Proposition 1 *The expected reconstruction difference* $\mathbb{E}[a]$ *at ENs is bounded by the expected prediction error* $\mathbb{E}[e]$ *at SANs, i.e.,* $\mathbb{E}[a] \leq \mathbb{E}[e]$.

Proof The expected reconstruction difference $\mathbb{E}[a]$ of the reconstruction difference in Eq. (4) is analyzed as follows:

$$\begin{aligned} \mathbb{E}[a] &= \mathbb{E}[||\mathbf{x} - \tilde{\mathbf{x}}|| | e \leq \theta] P(e \leq \theta) = \mathbb{E}[||\mathbf{x} - \hat{\mathbf{x}}|| | e \leq \theta] P(e \leq \theta) \\ &= \mathbb{E}[e | e \leq \theta] P(e \leq \theta) = \int_0^\theta e p_i(e) de \leq \mathbb{E}[e] \end{aligned} \tag{15}$$

where $P(e \leq \theta)$ is the probability of Case 2, where no context vector is delivered from SAN i to EN j, w.r.t., θ, $p_i(e)$ is

the probability distribution of the local prediction error at SAN i. □

In Case A, where the predicted vector at SAN i is the same as the reconstruction vector at EN j, the expected reconstruction difference is bounded by the expected prediction error. Hence, the evolution of the reconstructed context vectors at the EN j are *known* to the SAN i, where the latter produces those vectors from its local predictor f_i. This means that the SAN i knows the upper bound of the expected reconstruction error that the EN j will experience, thus, can adjust the decision threshold θ to satisfy the accuracy needs of the launched IoT analytics application.

Note that, the expected prediction error $\mathbb{E}[e]$ can be incrementally approximated at the SAN i by adopting the recursive approximation of the error mean $\tilde{e}_t$ as: $\tilde{e}_t = \tilde{e}_{t-1} + \frac{1}{t}(e_t - \tilde{e}_{t-1})$ for a large $t > 0$. Consider now the Case B.

Proposition 2 *The expected reconstruction difference* $\mathbb{E}[a]$ *at ENs is bounded by the expected prediction error* $\mathbb{E}[e]$ *at SANs and the expectation of* ξ, *i.e.,* $\mathbb{E}[a] \leq \mathbb{E}[e] + \mathbb{E}[\xi]$.

Proof The expected reconstruction difference $\mathbb{E}[a]$ of the reconstruction difference in (4) is:

$$\begin{aligned}\mathbb{E}[a] =\ & \mathbb{E}[||\mathbf{x} - \tilde{\mathbf{x}}|||e \leq \theta]P(e \leq \theta) = \mathbb{E}[||(\mathbf{x} - \hat{\mathbf{x}}) + (\hat{\mathbf{x}} - \tilde{\mathbf{x}})|||e \\ & \leq \theta]P(e \leq \theta) \leq \mathbb{E}[||\mathbf{x} - \hat{\mathbf{x}}|||e \leq \theta]P(e \leq \theta) + \mathbb{E}[||\boldsymbol{\rho}|||e \\ & \leq \theta]P(e \leq \theta) = \int_0^{\theta} ep_i(e)de + \mathbb{E}[||\boldsymbol{\rho}|||e \leq \theta]P(e \leq \theta) \\ & \leq \mathbb{E}[e] + \mathbb{E}[||\boldsymbol{\rho}|||e \leq \theta]P(e \leq \theta)\end{aligned}$$

In this case, the expected reconstruction difference is bounded at least by the expected prediction error, known at SAN i and the conditional expectation discrepancy $\mathbb{E}[||\boldsymbol{\rho}|||e \leq \theta]P(e \leq \theta)$ derived by the intrinsic difference of the reconstructed and predicted vectors. This conditional expectation refers to the expected value $\mathbb{E}[\xi]$ of the time series ξ_t defined in Eq. (14). That is:

$$\mathbb{E}[\xi] = \mathbb{E}[||\hat{\mathbf{x}} - \tilde{\mathbf{x}}|||e \leq \theta]P(e \leq \theta) = \mathbb{E}[||\boldsymbol{\rho}|||e \leq \theta]P(e \leq \theta).$$

□

The evolution of the times series ξ_t can be monitored in a *training* phase where the EN j sends the reconstructed vectors $\tilde{\mathbf{x}}_t$ to the SAN i. After this training phase, the SAN i is aware of the expected discrepancy by approximating the mean of the time series $\tilde{\xi}_t = \tilde{\xi}_{t-1} + \frac{1}{t}(\xi_t - \tilde{\xi}_{t-1})$. Based on this learned evolution of the time series ξ, the SAN i knows the upper bound of the expected reconstruction error that the EN j will experience, thus, can adjust the application specific decision threshold θ. Moreover, during this training phase, the SAN i can send the pairs $(\mathbf{x}_t, \hat{\mathbf{x}}_t)$ to the EN j to locally approximate both the expected prediction error and the expected discrepancy. In this context, the EN j can adjust the current reconstruction policy (Policy 1, 2, or 3) by selecting the policy that corresponds to the minimum $\mathbb{E}[a]$. In Sect. 5, we experiment with the proposed policies adopted by EN j to demonstrate the applicability of our model.

4.2 Predictability analysis and computational complexity

The adopted prediction and reconstruction algorithms (f_i and g_j) are based on the Exponentially Weighted Moving Average (EWMA) for one-step ahead prediction and reconstruction vectors $\hat{\mathbf{x}}_{t+1}$ and $\tilde{\mathbf{x}}_{t+1}$, respectively (at SAN i and EN j). That is, given a multivariate contextual vector time series $\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots$, the forecast on SAN i (and the reconstruction in EN j) has as follows:

$$\hat{\mathbf{x}}_{t+1} = (1-\alpha)\sum_{\tau=0}^{\infty} \alpha^{\tau}\mathbf{x}_{t-\tau} = (1-\alpha)\mathbf{x}_t + \alpha\hat{\mathbf{x}}_t \tag{17}$$

The recursion in Eq. (17) requires $O(d)$ space, i.e., only the d-dimensional context vector $\hat{\mathbf{x}}_t$ is stored for predictions, while the prediction requires $O(d)$ time. Now, we could rewrite Eq. (17) to demonstrate its strong relation with the linear Auto-Regression model (AR) and its prevalence in terms of predictability and computational complexity. It will be shown that the adopted EWMA consists of two parts: the AR model and the Moving Average (MA) model (Box and Jenkins 1990). Specifically, let us denote the one-step ahead forecast/prediction error vector $\mathbf{e}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$. Note, we analyze here the forecast error in the SAN i. Similar reasoning proceeds with the reconstruction difference in the EN j, where, as proved by Propositions 1 and 2, the expected reconstruction difference is bounded by the expected prediction error. Hence, we proceed with the dynamics of the forecasting on the SAN i.

By substituting in Eq. (17) the prediction error, we obtain:

$$\mathbf{x}_t - \mathbf{e}_t = (1-\alpha)\mathbf{x}_{t-1} + \alpha(\mathbf{x}_{t-1} - \mathbf{e}_{t-1}),$$

thus, the change in times series is expressed by

$$\Delta\mathbf{x}_t = \mathbf{e}_t - \alpha\mathbf{e}_{t-1}. \tag{18}$$

By taking $\mathbf{e}_t$ to be a time series of independent $\mathcal{N}(\mathbf{0}, \sigma_e^2\mathbf{I})$ variables, we observe that we deduce from the adopted EWMA recursion in Eq. (17) the simple Autoregressive Integrated Moving Average model (ARIMA) in Eq. (18). ARIMA model (Box and Jenkins 1990) is a generalization of an autoregressive moving average model fitted to the $\mathbf{x}_t$ time

series to predict future context vectors in the series, where vectors show evidence of non-stationarity.

The derived ARIMA model in our case, consists of two parts: the linear autoregressive (AR) part and the moving average (MA) part. The AR part involves regressing the variable on its own lagged/past contextual vector. In our case, the lag $p = 1$. The MA part involves modeling the error term as a linear combination of error terms occurring contemporaneously and at one time instance in the past. The predictions adopted by the EWMA (i.e., ARIMA with only one past context vector; $p = 1$) produced by the recursion in Eq. (17) are the minimum mean square error predictions (Muth 1960) by minimizing the expected squared prediction error: $\mathbb{E}\left[||\mathbf{x}_t - \hat{\mathbf{x}}_t||^2\right]$. Moreover, by adopting the multivariate AR(p) with lag $p > 1$, i.e., storing more than one past context vectors $\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-p}$ to proceed with $\hat{\mathbf{x}}_t$ forecasting, then the space complexity is $O(dp)$. In this case, in terms of prediction on SAN i, the computational time for calculating the linear autoregressive coefficients is $O(d^2p)$ based on the ordinary least squares (OLS) auto-regressive estimation. Given some evidence of non-stationarity, those coefficients should be re-estimated regularly, which implies huge complexity not only on the SAN but also on the EN, where the latter node has to maintain n multivariate AR(p) models; one for each connected SAN. In that case, the computational complexity for reconstruction is $O(npd^2)$ at the EN. For those reasons of predictability, computational complexity, and scalability performances we chose the EWMA in SAN and in EN for prediction and reconstruction (as provided in Policy 3).

5 Performance evaluation

5.1 Datasets and experimental setup

In our experiments, we used two real datasets for assessing the performance of the proposed edge prediction intelligence mechanism. The first contextual dataset (DS1) was adopted by UCI (Vito et al. 2008). This dataset contains twelve SANs of chemical compounds and environmental parameters: CO, PT08.S1 (tin oxide), Non Metanic HydroCarbons, Benzene, PT08.S2 (titania), NOx, PT08.S3 (tungsten oxide), NO_2, PT08.S4, PT08.S5 (indium oxide), temperature, relative humidity, and absolute humidity. All these contextual parameters are required to measure the air pollution of a specific area. These data are collected every hour and refer to $T = 9357$ 12-dimensional measurements with $n = 12$ SANs and one EN. Inside this dataset missing values occur. For each SAN, we impute those missing values by adopting the missing value imputation method of linear interpolation. This method exploits two data points (x_0, y_0) and (x_1, y_1) to reconstruct a linear function to find for a specific x value the missing value y as follows: $y = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$.

The second contextual dataset (DS2) refers to 4-dimensional contextual data collected by Raspberry Pi SANs deployed at the School of Computing Science, University of Glasgow (Hentschel et al. 2016). We used four different SANs' that measured: two different room temperatures (room F121 and room S123), humidity and sound (room F121). This data is collected by an interval of 10 min and refer to $T = 1000$ 4-dimensional measurements with $n = 4$ SANs and one EN. For comparison and reproduction, both datasets are *normalised* and *scaled*, i.e., each contextual parameter $x \in \mathbb{R}$ is mapped to $\frac{x-\mu}{\sigma}$ with mean value μ and variance σ^2, and scaled in [0,1] using $\frac{\max\{x\}-x}{\max\{x\}-\min\{x\}}$. That is, each context vector $\mathbf{x}$ is normalized and scaled in the d-dimensional unit cube $\mathbf{x} \in [0, 1]^d$, with $d \in (12, 4)$ for DS1 and DS2, respectively, and $\theta \in [0, 1]$.

Based on our sensitivity analysis of our mechanism in Sect. 5.4, the experimental assessment was carried out with different values of the decision threshold $\theta \in \{10^{-5}, 10^{-3}, 10^{-2}, 0.05, 0.06, 0.1, 0.2\}$. Using the normalized and scaled datasets DS1 and DS2, the physical meaning of θ is interpreted as the percentage change of a measured/sensed time series value x_t by: 0.0002, 0.02, 2, 10, 12, 20 and 40% respectively for the chosen θ values, respectively. Those θ values derived from our sensitivity analysis which examines the impact of θ on the local prediction error e_t in Eq. (2) in SAN and the reconstruction error on the EN in our mechanism. The local predictor/exponential smoother in the SAN, in our experiment, uses $\alpha \in \{0.5, 0.7\}$ as suggested in Durbin and Koopman (2012). Moreover, in Policy 3, the reconstruction smoother function in EN adopts $\alpha = 0.7$. The sliding window size is set to $N = 10$. This selected size represents for DS1 a history of the last 10 h and for DS2 a history of the last 100 min. Our experimental set up includes seven θ values and two α values over three different policies (Policy 1, 2, and 3) for reconstruction on the EN. This leads to an overall of 42 experiments for each of the aggregation analytics function $h(\mathcal{W})$: i.e., `AVG`, `MAX` and `MIN`, the linear regression analytics function and the reconstruction difference for evaluation. In order to objectively assess the performance of our mechanism, we implemented the baseline mechanism and also compare our mechanism with the TEEN model (Manjeshwar and Agrawal 2001). Which leads to an overall of 315 experimental results and one baseline solution. The baseline mechanism is produced by capturing the continuous contextual data and transmitting them from all SANs to an EN, without any predictive intelligence on SAN or EN. The TEEN model along with a theoretical comparative assessment is provided in Sect. 5.3, while in Sect. 5.4 we provide the comparative assessment of our model and TEEN.

5.2 Performance metrics

We firstly define the performance metrics of *counter of communications*, i.e., the number of sensed values are sent from a SAN i to its EN j. By adopting the baseline mechanism, for DS1, the number of communications is allocated with $12 \cdot 9357 = 112,284$ and, for DS2, the total number of communications is $4 \cdot 1000 = 4000$. To better illustrate and compare our mechanism with the baseline, the counter of communications is indicated as 100%. Using our mechanism, the counter is only increasing if θ is exceed and values are transmitted from the SANs to the EN. The overall communication of n SANs over T sensing values, in our method is then:

$$c(T) = \sum_{t=1}^{T} \sum_{i=1}^{n} I_{i,t}, \tag{19}$$

with $I_{i,t} = 1$ if SAN i sends its sensed value to the EN; otherwise $I_{i,t} = 0$. Evidently, the overall communication of n SANs over T sensing values, in the baseline method is $T \cdot n$, since $I_{i,t} = 1, \forall i, t$. The percentage of communication is then $\frac{c(T)}{Tn}$.

The re-construction difference a in Eq. (4), and the aggregation analytics difference β in Eq. (5) is evaluated using the *symmetric mean absolute percentage error (SMAPE)*. During the experiments, we calculated the average SMAPE per SAN for each time instance $t \in \{1, \ldots, T\}$. This metric is used to represent a percentage error of SMAPE $\in [0, 100]$ because of its unbiased properties (Tofallis 0000). SMAPE is defined for reconstruction difference a and aggregation analytics difference β as:

$$\text{SMAPE} = \begin{cases} \frac{100}{T} \sum_{t=1}^{T} \frac{a_t}{||\mathbf{x}_t|| + ||\tilde{\mathbf{x}}_t||}, & ||\mathbf{x}_t|| + ||\tilde{\mathbf{x}}_t|| > 0 \text{ for } a, \\ \frac{100}{T} \sum_{t=1}^{T} \frac{\beta_t}{||h(\mathcal{W})|| + ||h(\mathcal{W}^*)||}, & ||h(\mathcal{W})|| + ||h(\mathcal{W}^*)|| > 0 \text{ for } \beta. \end{cases} \tag{20}$$

For the regression analytics difference, we simply illustrate the metric γ_t per SAN as defined in Eq. (10). Our major aim is to demonstrate the trade-off between communication and re-construction difference/aggregation analytics/regression analytics difference. That is, we can tolerate some *slight* increase in the analytics error by gaining a *significant decrease* in the number of communications, thus, being communication efficient and prolonging the edge network lifetime. It is evaluated how the aggregated analytics functions $h(\mathcal{W})$, regression analytics function, and re-construction inside an EN behave with decreasing the number of communications towards the EN. This decrease of communications is produced by increasing the value of θ and changing the value of α inside the SAN (exponential smoother). The behaviour of SMAPE and γ depends on the the reconstruction policies inside the EN as will be shown in the remainder.

5.3 Model comparison

In order to demonstrate the fact that our generalized mechanism departs from the selective data delivery mechanism in distributed environments, we compare with the well known TEEN model (Manjeshwar and Agrawal 2001). Specifically, the TEEN model focuses only on the prediction mechanism on the SAN i and there is no focus on how the EN j reconstructs the conditionally received context vector. Based on this limited functionality, the TEEN model assumes that the context vector series $\mathbf{x}_t$ has no trend, which means that it forecasts zero change in the level from one time instance to the next. To proceed with a theoretical and experimental comparative assessment of our generalized mechanism with the TEEN mechanism, the SAN i is adopting the TEEN model for forecasting and the corresponding EN j is adopting the TEEN model for re-constructing the undelivered contextual vectors, which corresponds to Policy 1. The forecast (prediction) of a SAN i, which adopts the TEEN model (hereinafter it is referred to as TEENSAN), at $t+1$ is the time series vector at time t:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{x}_t, \tag{21}$$

which is different with our SAN i predictor, where it estimates a local mean before turning around and using it as a forecast for the next time instance. This indicates that our SAN i's next forecast is computed by interpolating between the last observed context vector $\mathbf{x}_t$ and the forecast that had been made for it $\hat{\mathbf{x}}_t$, i.e., $\hat{\mathbf{x}}_{t+1} = \alpha \mathbf{x}_t + (1-\alpha)\hat{\mathbf{x}}_t$. It is clear that TEENSAN i is adopting the random walk model, that is $\alpha = 1$. Based on the fact that the constant-forecast model is adopted when $\alpha = 0$, our SAN i predictor is an interpolation between the mean model and the TEENSAN (random walk) model w.r.t. the way it adapts to *new* context vectors. This is expected to performs more accurately (in terms of forecasting as will be proved in Proposition 3) than either of the situations where the random walk model over-responds and the mean model under-responds to new context vectors. Moreover, we can write the forecast formula of our SAN i as:

$$\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{x}}_t + \alpha \mathbf{e}_t,$$

with error vector $\mathbf{e}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$, to demonstrate the qualitative difference with the TEENSAN i model. This version provides a nice interpretation of the meaning of α in our SAN compared to the TEENSAN, which indicates the fraction of

the forecast error that is believed to be due to an unexpected change in the level of the series rather than an unexpected one-time event. In the limit of $\alpha = 1$, which is the TEEN model, all of the variation from one period to the next is believed to be due to a change in the fundamental level rather than just a temporary deviation. While, for $\alpha \in (0, 1)$, our SAN *i* predictor adapts to significant changes in the fundamental level of the series from one period to the next.

From an error analysis perspective, let us denote with $\tilde{\mathbf{e}}_t$ and $\mathbf{e}_t$ the prediction error of the TEENSAN *i* predictor and our SAN *i* predictor, respectively.

Proposition 3 *For $\alpha \in (0, 1)$, the expected prediction error of SANs is bounded by the expected prediction error of TEENSANs, i.e.,*

$$\mathbb{E}[||\mathbf{e}||] \leq \alpha \mathbb{E}[||\tilde{\mathbf{e}}||] < \mathbb{E}[||\tilde{\mathbf{e}}||].$$

Proof In the SAN, based on the exponential smoothing, we obtain the recursion:

$$\hat{\mathbf{x}}_{t+1} = \sum_{k=0}^{t-1} \alpha(1-\alpha)^k \mathbf{x}_{t-k} + (1-\alpha)^t \mathbf{x}_0. \tag{22}$$

Hence, we obtain the following relation between the prediction error of the TEENSAN and SAN, i.e.,

$$\mathbf{e}_{t+1} = (1-\alpha)\mathbf{x}_t - \boldsymbol{\phi}_t + \alpha \tilde{\mathbf{e}}_{t+1}, \tag{23}$$

where the factor $\boldsymbol{\phi}_t$ is defined as:

$$\boldsymbol{\phi}_t = \sum_{k=1}^{t-1} \alpha(1-\alpha)^k \mathbf{x}_{t-k} + (1-\alpha)^t \mathbf{x}_0. \tag{24}$$

Obviously, for $\alpha = 1$, i.e., TEENSAN, we obtain that $\boldsymbol{\phi}_t = \mathbf{0}, \forall t$, thus, $\tilde{\mathbf{e}}_t = \mathbf{e}_t$. In the case where $\alpha \in (0, 1)$ then the norm of the prediction errors of TEENSAN and SAN are as follows:

$$||\mathbf{e}_{t+1}|| \leq ||(1-\alpha)\mathbf{x}_t - \boldsymbol{\phi}_t|| + \alpha ||\tilde{\mathbf{e}}_{t+1}|| \tag{25}$$

Now, let us take the expectation $\mathbb{E}[||(1-\alpha)\mathbf{x}_t - \boldsymbol{\phi}_t||]$. Given that $1 - \alpha < 1$ and considering the unconditional expectation $\mathbb{E}[||\mathbf{x}||] = \mu < \infty$ (weakly stationary), we obtain that:

$$\begin{aligned}\mathbb{E}[||(1-\alpha)\mathbf{x}_t - \boldsymbol{\phi}_t||] &= (1-\alpha)\mu - \alpha\mu \sum_{k=1}^{t-1} (1-\alpha)^k - (1-\alpha)^t \mu \\ &= (1-\alpha)\mu - (1-\alpha)\mu[1-(1-\alpha)^{t-1}] - (1-\alpha)^t \mu,\end{aligned}$$

since $\sum_{k=1}^{t-1}(1-\alpha)^k = \frac{1-\alpha}{\alpha}(1-(1-\alpha)^{t-1})$. For large t we obtain the limit: $\lim_{t\to\infty} \sum_{k=1}^{\infty}(1-\alpha)^k = \frac{1-\alpha}{\alpha}$. Hence, the limit of the expectation is then $\lim_{t\to\infty} \mathbb{E}[||(1-\alpha)\mathbf{x}_t - \boldsymbol{\phi}_t||] = (1-\alpha)\mu - (1-\alpha)\mu = 0$. This means that for $\alpha \in (0, 1)$:

$$\mathbb{E}[||\mathbf{e}||] \leq \alpha \mathbb{E}[||\tilde{\mathbf{e}}||] < \mathbb{E}[||\tilde{\mathbf{e}}||].$$

□

Based on Proposition 3, the SAN predictor performs better than the TEENSAN in terms of context vector prediction. Section 5.4 reports on the experimental comparative assessment and sensitivity analysis of our mechanism and the TEEN model.

5.4 Sensitivity analysis and comparative assessment

In this section we provide a sensitivity analysis of our mechanism, especially focused on the decision threshold θ. Based on this analysis, we demonstrate the impact of θ on the data reconstruction quality given a certain pair: SAN *i* and EN *j*. The outcome of this analysis is to provide useful insight on the *appropriate* range of the θ values to ensure highly quality reconstructed data. In order to investigate this impact, we assess the sensitivity of our mechanism based on the following metrics adopted in signal processing for time series data reconstruction: (1) Kullback–Leibler (KL) divergence, (2) sum of squared residuals, (3) variance of the actual and the reconstructed time series, (4) coefficient of variation of the actual and reconstructed time series. For the sake of readability, we suppress the subscript of dimension *k* from the variable x_{kt} for $k \in (1, d)$ and focus on the communication pair (SAN *i*, EN *j*).

The first baseline sensitivity metric is the sum of squared residuals (SSR). SSR, or the sum of squared errors of time series reconstruction, is the sum of the squares of residuals; deviations of the reconstructed $\tilde{x}_t$ time series from the actual values of the time series x_t, i.e.,

$$SSR_t = \sum_{\tau=1}^{t} (x_\tau - \tilde{x}_\tau)^2. \tag{26}$$

SSR measures the discrepancy between the time series reconstructed at EN *j* due to the applied Policy 1, 2, or 3 and decision threshold θ and the actual time series x_t observed at SAN *i*. A small RSS value indicates a tight fit of the EN reconstruction model to the actual time series at SAN.

A more advanced metric for the sensitivity analysis is the Coefficient of Variation (CV). CV (also known as *relative standard deviation*), is a standardized measure of dispersion of the probability distribution $p(x_t)$ of the time series x_t. It is expressed as the ratio of the standard deviation σ_x to the absolute mean value $|\mu_x|$, i.e.,

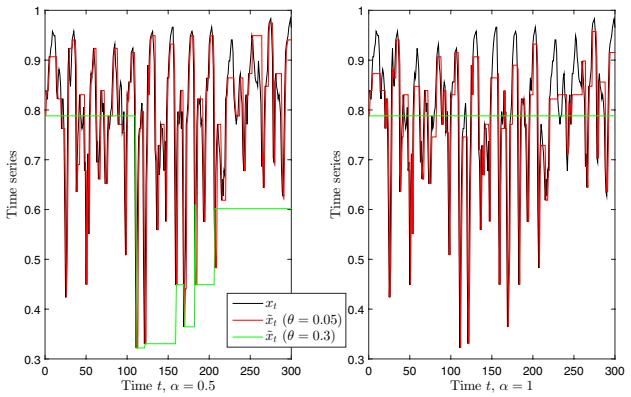

Fig. 2 Actual x and reconstructed $\tilde{x}$ time series at the EN for Policy 1 with $\theta \in (0.05, 0.3)$ and *left* $\alpha = 0.5$ in SAN, *right* $\alpha = 1$ in SAN, i.e TEEN

$$CV(x) = \frac{\sigma_x}{|\mu_x|}. \tag{27}$$

We adopt $CV(x)$ for our sensitivity analysis because the standard deviation σ_x of the time series x_t must be understood in the context of the mean of the time series compared with the $CV(\tilde{x})$ of the reconstructed time series at the EN, i.e., $\frac{\sigma_{\tilde{x}}}{|\mu_{\tilde{x}}|}$. We use CV for comparing *both* the mean and the variance of the reconstructed time series in EN j with the actual time series in SAN i data. Note that the value of the CV is independent of the unit, i.e., it is a dimensionless number. The rationale behind investigating the sensitivity of our mechanisms w.r.t. the CV is that many natural processes indeed show a correlation between the average value and the amount of variation around it. By observing the coefficients of variation $CV(\tilde{x})$ and $CV(x)$ of the reconstructed and actual time series, respectively, we can assess the *discrepancy* of the reconstructed time series at EN j w.r.t. the actual time series at SAN i due to the adoption of certain Policy 1, 2, or 3 and decision threshold θ. Ideally, we would desire the reconstructed mean and variance of the times series in EN j to 'follow' the actual mean and variance of the actual time series in SAN i. However, due to the applied threshold θ, the variance of the reconstructed signal decreases thus the $CV(\tilde{x})$ deviates significantly from the $CV(x)$ with high values of θ.

Departing from the SSR and the CV, we further investigate the sensitivity of our model by examining the probability distribution of the time series $p(x)$, which is observed at the SAN i and reconstructed probability distribution $p(\tilde{x})$ at the EN j. This provides a *holistic* insight of the discrepancy of the statistics based on the applied Policies 1, 2, or 3 and the impact of the decision threshold θ. For this investigation, we adopt the Kullback-Leibler (KL) divergence. KL divergence from $p(x)$ to $p(\tilde{x})$ denotes the *information loss* when we attempt to reconstruct time series $\tilde{x}$ for the actual time series x provided

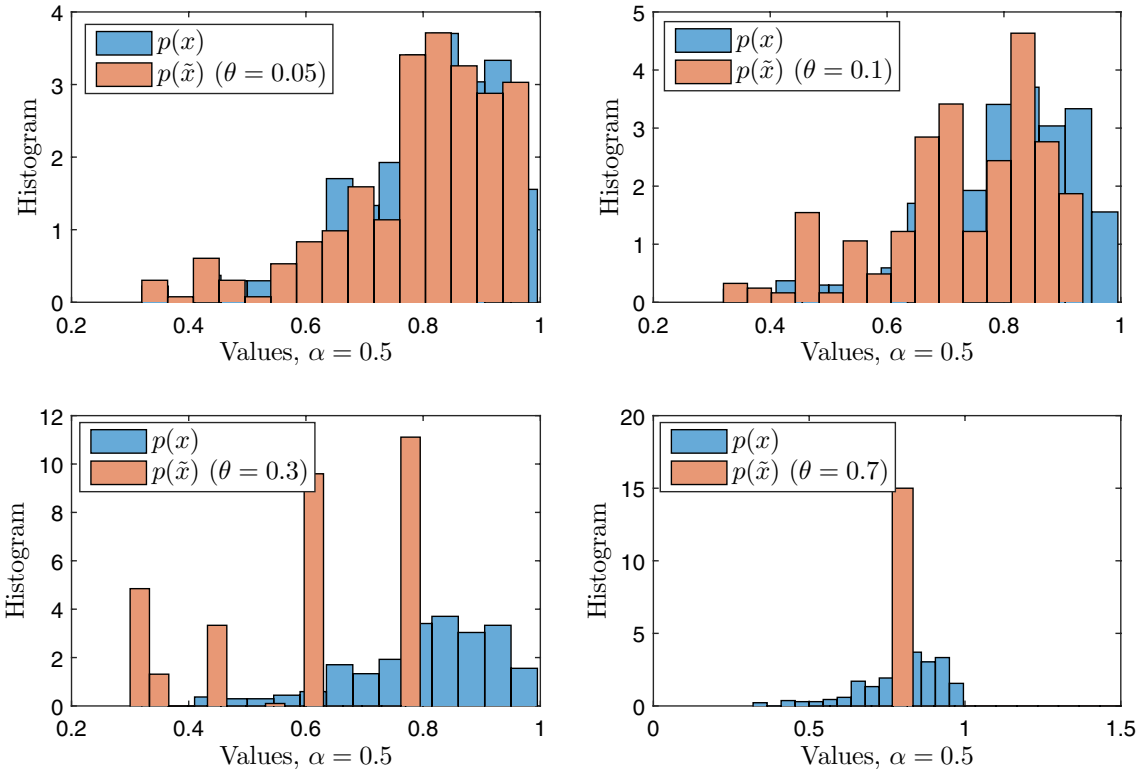

Fig. 3 Histograms for actual time series and reconstructed time series at EN for Policy 1 with $\theta \in (0.05, 0.1, 0.3, 0.7)$ and $\alpha = 0.5$

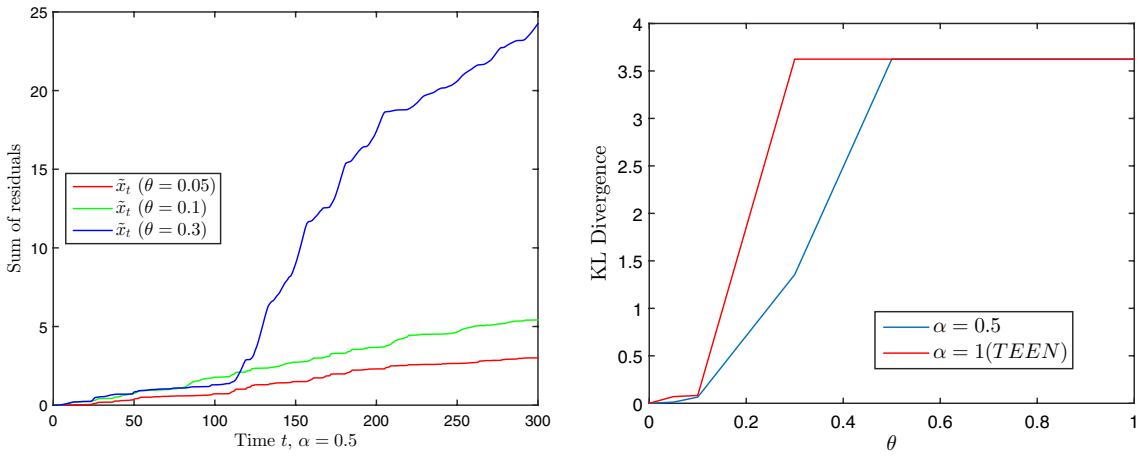

Fig. 4 *Left* Cumulative sum of squared time series residuals $(x - \tilde{x})^2$ for reconstructed $\tilde{x}$ time series at the EN for Policy 1 with $\theta \in (0.05, 0.1, 0.3)$ and $\alpha = 0.5$ in SAN; *right* KL divergence of the reconstructed probability distribution from the actual probability distribution for Policy 1 with $\alpha = 0.5$ and TEEN (at SAN)

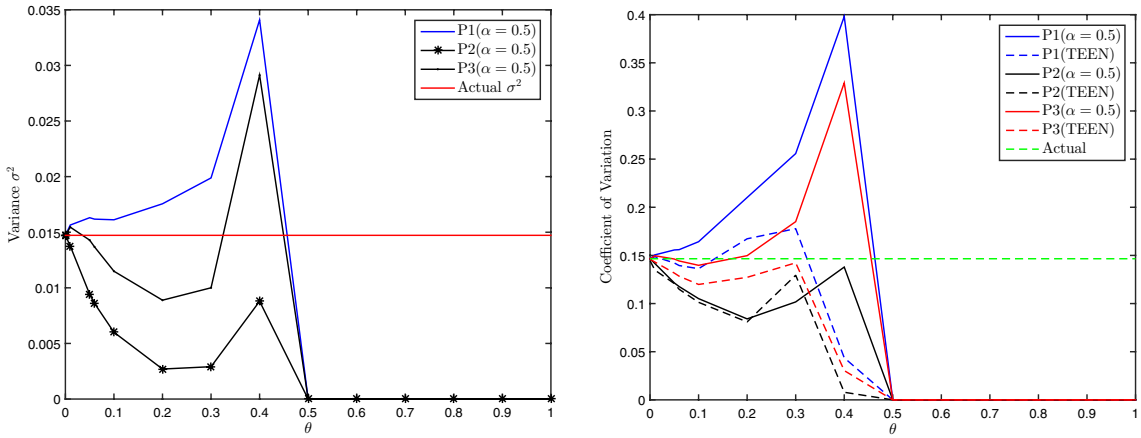

Fig. 5 *Left* Variance σ^2 of the actual and reconstructed time series at EN for Policy 1, 2, and 3 against θ for $\alpha = 0.5$; *right* coefficient of variation (CV) of the actual and reconstructed time series at EN for Policy 1, 2, and 3 against θ for $\alpha = 0.5$ and TEEN (at SAN)

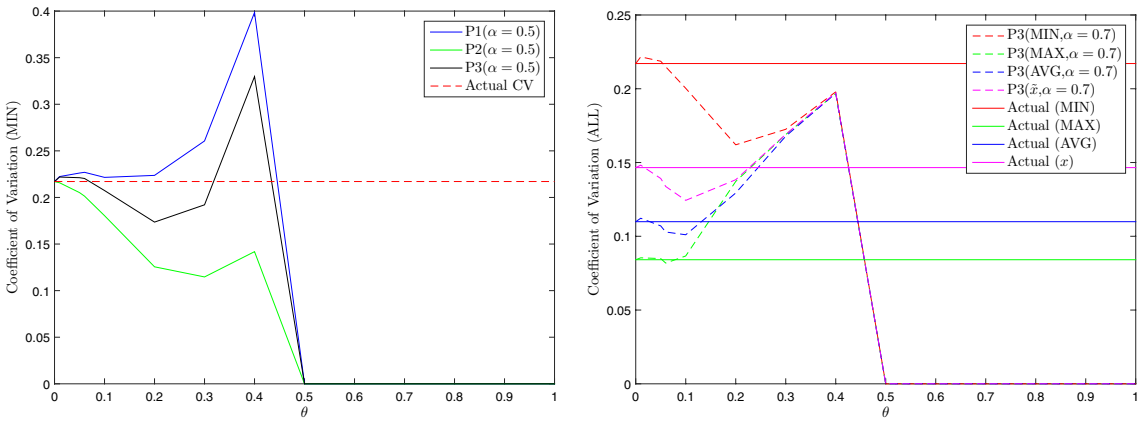

Fig. 6 *Left* Coefficient of variation (CV) of the actual and reconstructed time series at EN for Policy 1, 2, and 3 against θ for $\alpha = 0.5$ for the `MIN` aggregation; *right* coefficient of variation (CV) of the actual and reconstructed time series at EN for Policy 3 against θ for $\alpha = 0.7$ for all the aggregation operators `AVG`, `MIN`, `MAX`, and the reconstructed time series

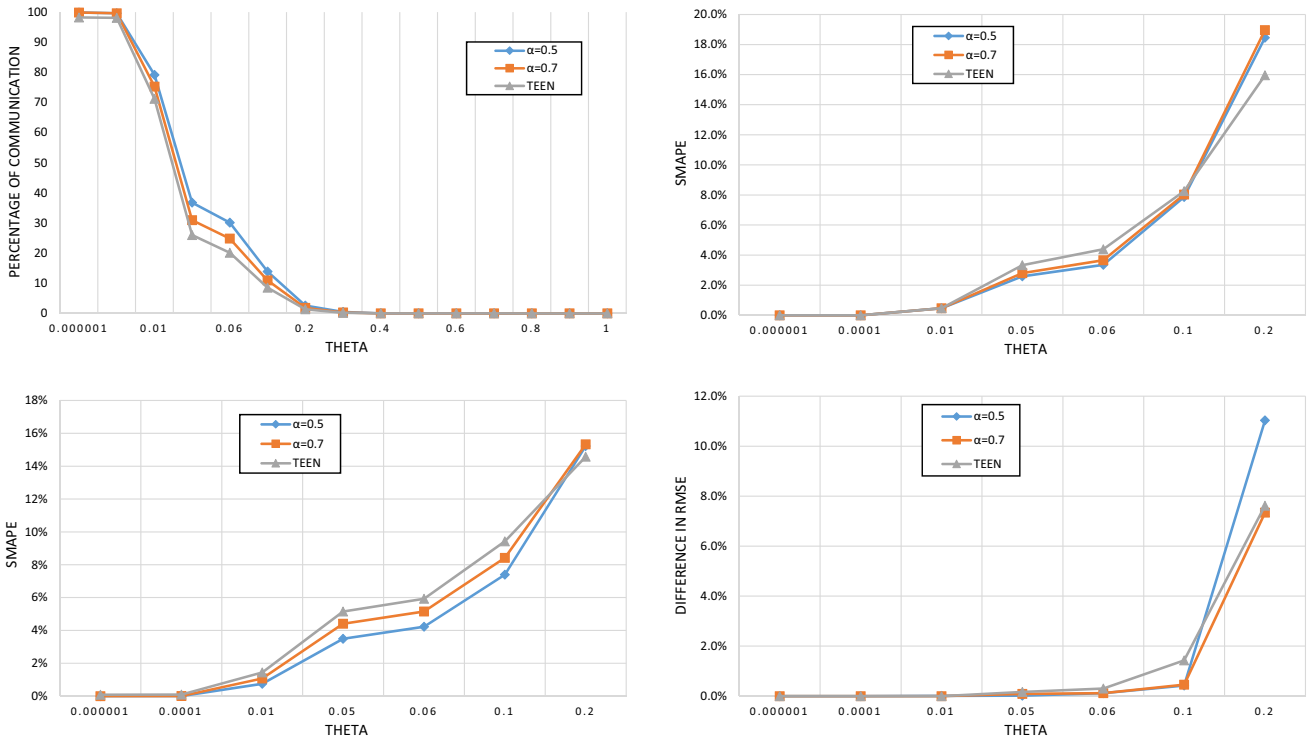

Fig. 7 For DS1: *top left* percentage of remaining communication against θ using $\alpha \in (0.5, 0.7)$ and TEEN; *top right* re-construction difference SMAPE against θ for Policy 1 using $\alpha \in (0.5, 0.7)$ and TEEN; *lower left* Aggregation Analytics Difference (AVG) SMAPE against θ for Policy 1 using $\alpha \in (0.5, 0.7)$ and TEEN; *lower right* Predictive Analytics Difference against θ for Policy 1 using $\alpha \in (0.5, 0.7)$ and TEEN

that $p(\tilde{x})$ and $p(x)$ are the probability distribution functions, respectively. KL is defined as:

$$KL(p(\tilde{x})||p(x)) = \int_0^1 p(\tilde{x}) \log \frac{p(\tilde{x})}{p(x)} dx. \tag{28}$$

In our case, KL indicates is the amount of information lost when EN j approximates the actual time series at SAN i based on our Policies 1, 2 or 3. Ideally, we would like $p(\tilde{x})$ to be as close to the real/actual $p(x)$ given certain policies and for given θ values. We are examining the impact of θ and the applicability of TEEN w.r.t. our proposed mechanism in terms of approximating the actual probability distribution of the time series reconstructed in EN.

By performing the introduced sensitivity analysis metrics in Eqs. (26), (27) and (28), we obtain the Figs. 2, 3, 4, 5 and 6. The purpose of analyzing these metrics is to identify the upper bound of θ for selecting one dimensional time series from DS1; similar sensitivity results are obtained using DS2. Not only the sensitivity metrics are important for the appropriate and reasonable θ values, but also to consider the relationship between increase of θ and the percentage of communication overhead occur between SAN and EN (please refer to Fig. 7 top left,which shows this relationship). By combining Fig. 7 with Fig. 2, it is clearly applicable that given a small θ value, the reconstructed time series $\tilde{x}$ follows the actual time series x. This is caused by the fact that still high communication occur between SAN and EN. By increasing the values of θ, it leads to less communication overhead, while with a $\theta > 0.2$ one can observe that only a couple of values are sent from SAN to EN. This is represented by the green lines in Fig. 2. It should also be mentioned that the number of communication is highly depending on the chosen α value in the SAN. Looking at Fig. 2 with $\alpha = 1$, which is equivalent to the TEEN model in SAN, it leads to the fact that only one value is sent from SAN to EN with $\theta = 0.3$. Whereas for the same θ value and $\alpha = 0.5$, more measurements are sent in between. Closely related to the number of communication between SAN and EN as well as the comparison of the reconstructed time series with the actual time series, is the probability density function $p(x)$ for indicators of the behavior of high θ values. Figure 3 shows the probability density functions (shown as histograms) of the actual and the reconstructed time series with fixed $\alpha = 0.5$ for different $\theta \in (0.05, 0.1, 0.3, 0.7)$. We can observe that an increase of θ decreases the possibility to reconstruct the actual distribution in the EN, i.e., the reconstructed time series follows a significantly different probability density function with that of the actual time series, especially when $\theta > 0.2$.

The aforementioned relationship between small θ values and tight-fitting of the distribution is undermined with the analysis of the sensitivity metrics. Not only is it impossible to follow the probability density function of the actual time series with increase of θ, further the sensitivity metric of SSR clearly identifies in Fig. 4 that using $\theta > 0.2$ causes a loose-fitting of the EN reconstruction model to the actual SAN model. While $\theta \leq 0.2$, the SSR is increasing over time but still in a reasonable range which is applicable as SSR growing in an exponential way. Besides the SSR metric, the coefficient of variation (CV) metric indicates the development of θ and its impact on the probability distribution. Figures 5 (right) and 6 and show that by using $\theta > 0.5$ produces an CV value of 0, which holds true for all policies and aggregation methods as well for the reconstruction case. From CV in Eq. (27) it can be seen that either the mean or the variance has to be zero if the CV is zero. This can clearly been observed in Fig. 5 (left) where variance is zero for $\theta > 0.5$. This finding and the CV effect can be explained by the knowledge and insights we gained from the sensitivity metrics KL and SSR. After $\theta > 0.5$ limited communication between SAN and EN occurs and, therefore, a low or zero variance time series is reconstructed in the EN. Moreover, from the behavior of the CV metric, we can observe that a reconstruction of the actual CV depends on (1) the policy adopted for reconstruction, (2)exponential smoothing or TEEN adopted on SAN. A comprehensive comparison between those cases is provided in Sect. 5.5.

Lastly, by evaluating the KL divergence of the reconstructed probability distribution in Fig. 4 (right), it is observed that, depending on α, the KL is only slightly increasing up to a θ value of approximately 0.1. After having $\theta > 0.1$, the loss of information is growing linearly. By adopting the TEEN in the SAN, it is applicable that the loss of information is higher for a lower value of θ than using exponential smoothing with $\alpha = 0.5$. It can be observed that the maximum loss of information is 3.5, which indicates a loss of 350% of communication. This maximum is reached for TEEN with $\theta > 0.3$, and $\alpha = 0.5$ for $\theta > 0.5$. In order to obtain information loss of less than 100%, then θ should be less than 0.2.

After analyzing a range of sensitivity metrics for different policies, α values and the TEEN, we draw our conclusion that the range for $\theta \in (0.2, 1]$ is not useful for achieving high quality data and, therefore, in the experimental and comparative assessment we adopt the range of $\theta \in (0, 0.2]$.

5.5 Performance assessment

Based on the results obtained from our sensitivity analysis of our model in Sect. 5.4, in this section we evaluate the performance of our proposed method over elucidated datasets DS1 and DS2 and the defined performance metrics to illustrate the trade-off between communication and error for the reconstruction, aggregation analytics and regression analytics differences. Independent of the specific differences, our aim is to reduce the percentage of communication only with a slight increase of the error.

Generally, over all evaluated differences, increasing the value of θ is decreasing the number of communications towards the EN which is applicable from Fig. 7. The reason behind this is that θ is demonstrating the tolerance for a change of the expected value and the actual/sensed one. Therefore high values of θ are indicating that values can vary between a larger range before they are sent towards the EN (refer also to sensitivity analysis in Sect. 5.4). Furthermore, it is worth noting that the number of communications is highly dependent on the exponential smoother parameter α. Given the same θ value, the number of communications is decreasing with higher values for α at which $\alpha = 1$ is equivalent to the TEEN model under comparison. Increasing α means reducing the influence of previous/historical measured data. Having $\alpha = 1$ denotes that the current measured value is compared only against the previous for a forwarding decision inside the SAN. The following assessments are achieved by adopting the three reconstruction Policies in the EN with values for $\alpha \in \{0.5, 0.7, 1\}$ and θ values up to the upper bound of 0.2 resulted from Sect. 5.4, i.e., $\theta \in \{10^{-5}, 10^{-3}, 10^{-2}, 0.05, 0.06, 0.1, 0.2\}$. In further evaluation the shown figures represent only some results of our evaluation of the proceed 315 experiments for each dataset. They are representing the general outcome. Moreover, it should be note that the displayed dots inside the figures are values for θ with the relation towards its difference and its communication.

5.5.1 Re-construction difference assessment

Evaluating the re-construction difference for DS1 and DS2 it is applicable that the re-construction difference is depending on the chosen α value this is seen in Fig. 7 (top right). Generally using TEEN inside the SAN is creating a higher error for re-construction than using the exponential smoothing with $\alpha = 0.5$. However, the trade-off between communication and re-construction difference is of importance for evaluating the efficiency of our model. Evaluating this trade-off we use the three proposed policies and values for $\alpha \in \{0.5, 0.7\}$ and $\theta \in \{10^{-5}, 10^{-4}, 10^{-2}, 0.05, 0.06, 0.1, 0.2\}$. Further we compare our model using TEEN in SAN. Applicable from Fig. 8 Policy 2 is creating the highest re-construction difference trade-off over both datasets. Whereas Policy 1 and 3 are nearly identical for DS1 but for DS2 Policy 1 is identified as best solution. This perception is applicable over all α values. Therefore, we can conclude that the chosen reconstruction policy in the EN is highly influencing the produced SMAPE and its efficiency towards the communication.

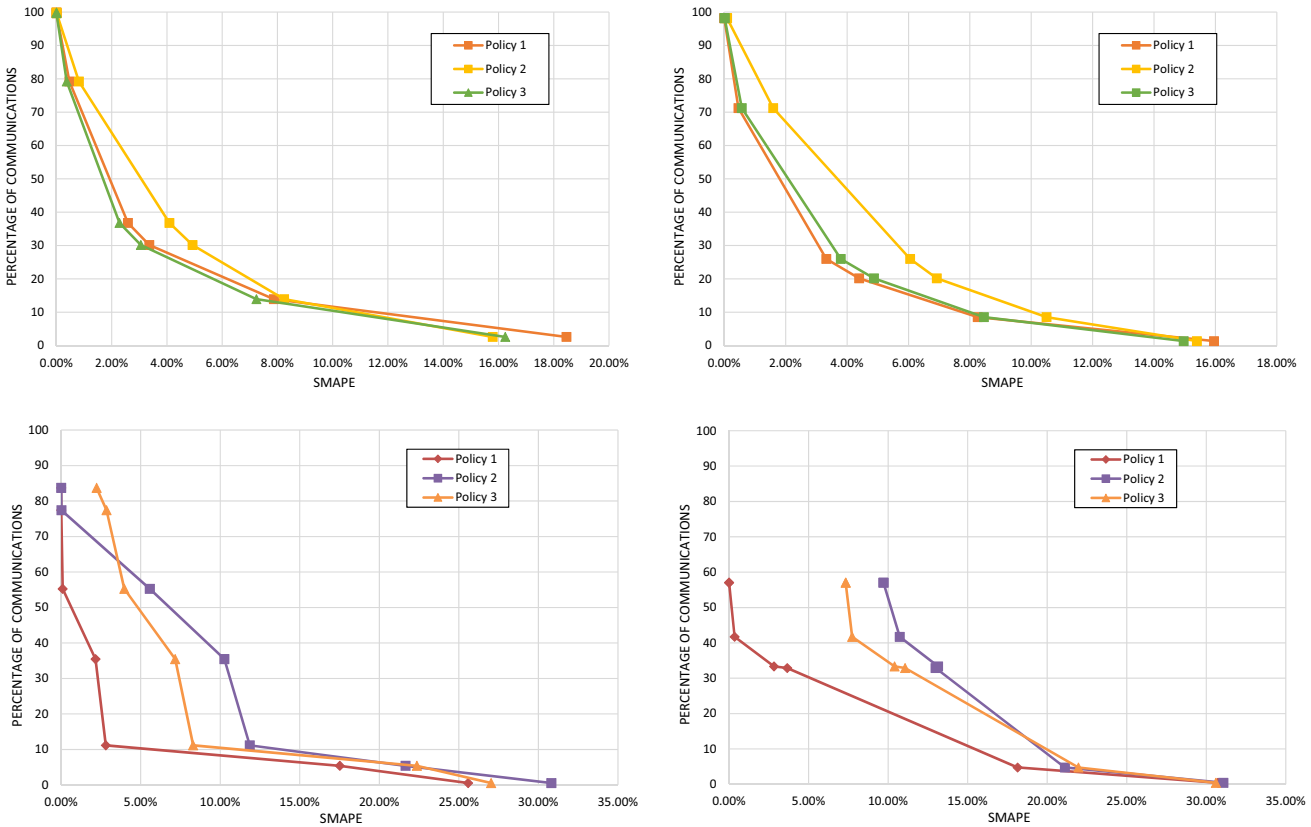

Fig. 8 Re-construction difference trade-off for Policy 1,2,3; percentage of communication against SMAPE with *top left* DS1 with $\alpha = 0.5$, *top right* DS1 with TEEN, *lower left* DS2 with $\alpha = 0.5$, *lower right* DS2 with TEEN

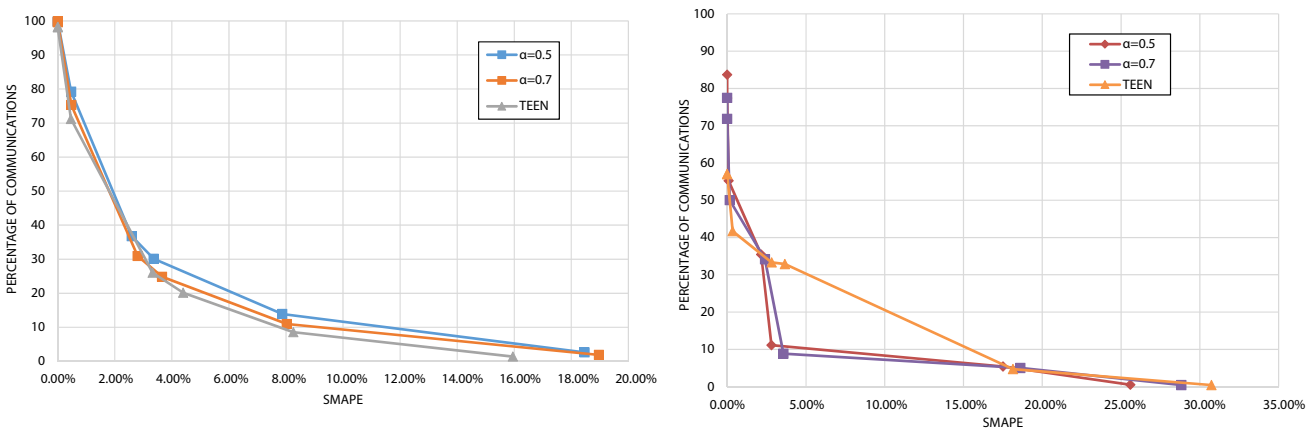

Fig. 9 Re-construction difference trade-off for Policy 1; percentage of communication against SMAPE with *left* DS1 and *right* DS2

Using policy 1 to identify the influence of the chosen α value it can be seen that in Fig. 9 that decreasing the communication the re-construction difference in all α values is increasing up to around 20%. The maximum of 20% is explained by the sensitivity assessment previously where it was shown that after a certain value of θ only some values are send and the distribution $p(x)$ is mostly around the mean. This is leading to a maximum difference of around 20%. Besides this realisation, it is applicable that all chosen α values are producing a similar trade-off between SMAPE and percentage of communication. However, for DS1 TEEN in SAN is producing a slightly lower error with less communication that $\alpha = 0.5$ or $\alpha = 0.7$. For DS2 this holds only for communication savings up to 40% whereas decreasing

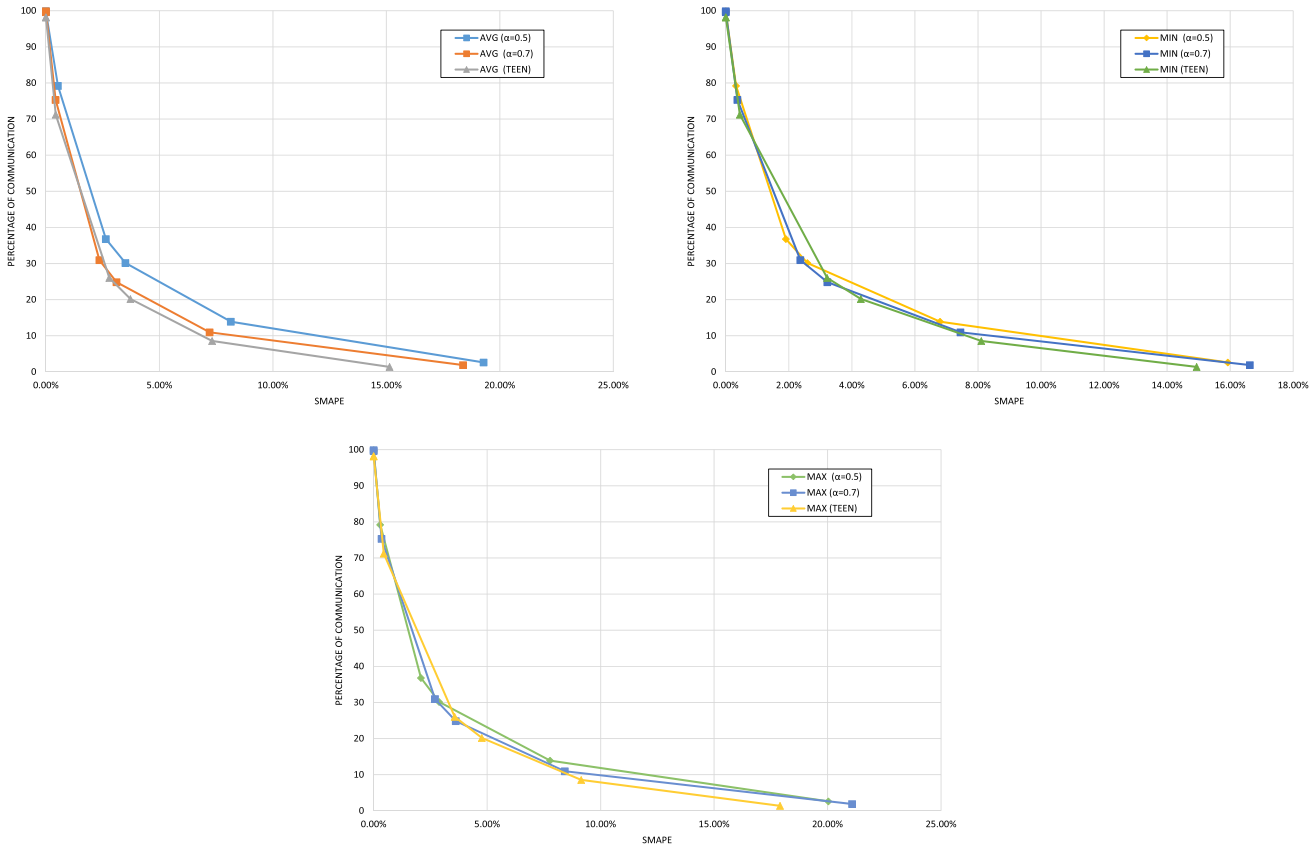

Fig. 10 Aggregation analytics difference trade-off for DS1 with Policy 1; percentage of communication against SMAPE for DS1 *left top* AVG, *right top* MIN, *lower* MAX

the communication further using our proposed model with $\alpha \in (0.5, 0.7)$ is significantly better than TEEN in SAN.

It is worth noting that, as shown in Figs. 8 and 9, using our proposed predictive intelligence for edge analytics on DS1, a communication overhead of 30% can be saved by tolerating an re-construction error of less than 1%. If IoT applications can tolerate up to 2% error in their analytics accuracy, it is possible to save between 50 and 60% of communication. Saving this amount of communication with a given tolerated error would increase the lifetime of an edge network between 30 and 50%. Moreover, in Fig. 9, it is applicable that for DS2 an even extreme trade-off is performed. Depending on the chosen value for α, without any error produced, up to 50% communication can be saved. Tolerating a relatively slight 2.5% error, it is possible to save even up to 70% of communication. This phenomenon can be explained by considering that DS2 is measuring every 10 min, which is causing similar or even identical measurements because of slightly changes in the indoor environment of the School of Computing Science. In the contrary, DS1 is measuring every hour, thus, the surrounding environment could change significantly towards the previous measure.

5.5.2 Aggregation analytics assessment

Besides the re-construction difference, the aggregation analytics difference produced by our method is an important metric by many analytics IoT applications. The introduced figures in 8 for the re-construction difference are similar to the comparison of the policies for the aggregation analytics differences. Therefore no further figure is shown in this assessment. From our experiments it is applicable for both datasets and all three aggregation functions, `AVG`, `MIN` and `MAX`, that Policy 2 is producing the highest SMAPE for the aggregation analytics difference. Similar to the re-construction difference, Policy 1 and 3 are generating the lowest average error per SAN over the entire time frame T. However it should be note that in our experiments a value of $\alpha = 0.5$ is producing the lowest error over all three aggregation functions by employing the re-construction Policy 3. Comparing Policy 1 over all three aggregation functions, Fig. 10 shows this for DS1 and Fig. 11 for DS2. Specifically, one can observe from Figs. 10 and 11 that, similar to the re-construction difference, the aggregation analytics difference depends on α. Higher values of α producing a better trade-off. In both

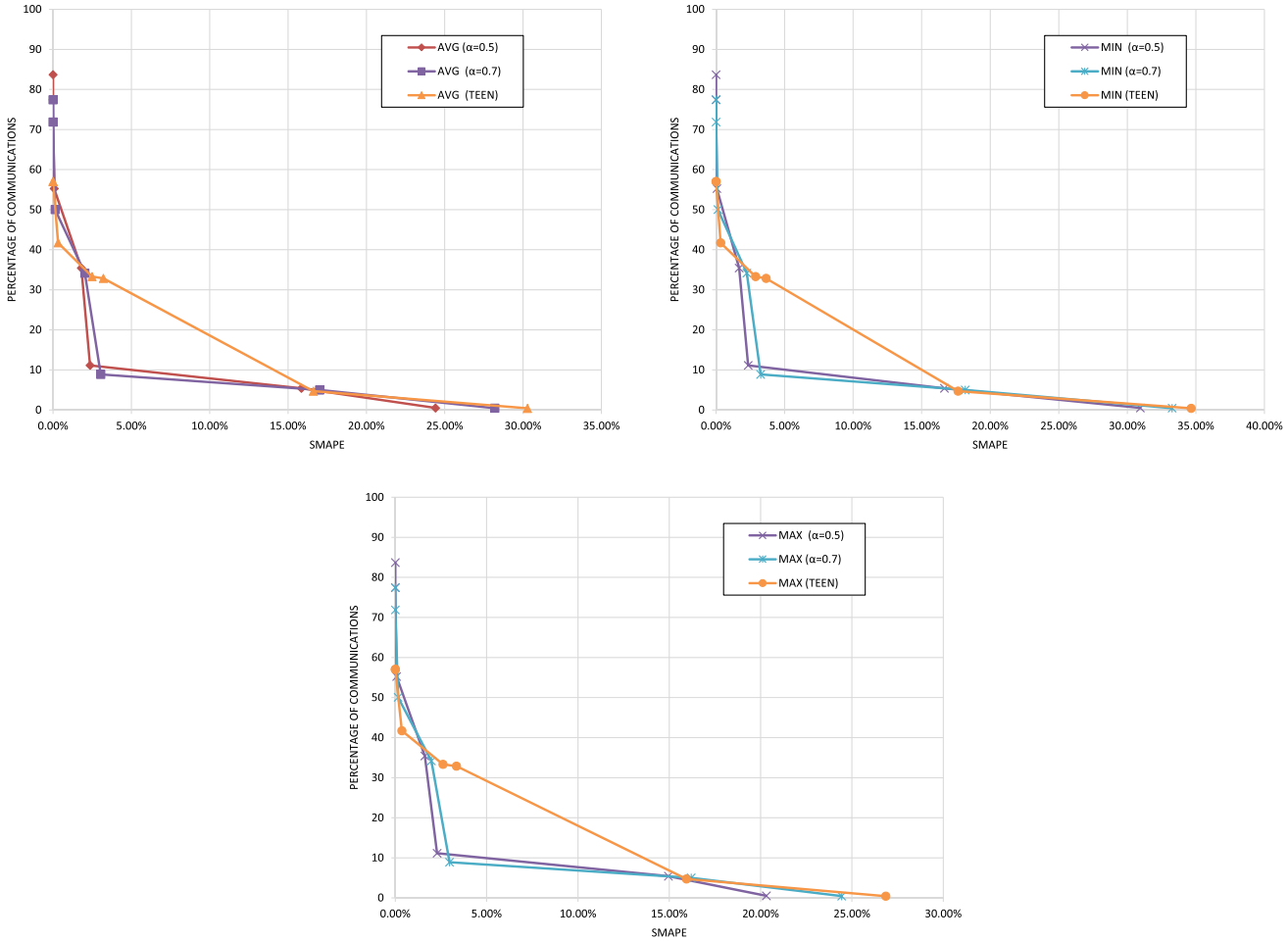

Fig. 11 Aggregation analytics difference trade-off for DS2 with Policy 1; percentage of communication against SMAPE for DS2 *left top* AVG, *right top* MIN, *lower* MAX

datasets for MIN and MAX this reverses after a SMAPE of around 1.5%. AVG in DS1 continuously produces a better trade-off for high α but for DS2 a reverse order appears too after 1.5% SMAPE.

For DS1, it is observed in Fig. 10, that a reduction of 20% by communication for MIN, MAX and 30% for AVG is only generating an error of 0.5%. Therefore, it is possible to increase the lifetime of a network up to 30% with tolerating a slightly difference towards the true result. For IoT applications that can tolerate a higher discrepancy for this kind of aggregation functions, they can save up to 50% with an error of 1.5–2%. In comparison with DS2, as illustrated in Fig. 11 for all three aggregation functions, it is possible to use only 60% of the communication *without any difference* towards the aggregation analytics output. A further 20% can be saved by tolerating an accuracy difference of maximum 0.5%.

5.5.3 Predictive analytics assessment

The Regression Analytics Difference is evaluated by using the Air Quality chemical compounds SANs: PT08.S1 (CO) and PT08.S5 (O_3) for DS1. For the DS2, both temperature SANs from rooms F121 and S123 are chosen for performing the linear regression analytics function. For DS1 the first 6000 data points are used for on-line training the linear regression coefficient whereas the last 3357 measured values are used for testing the prediction accuracy derived from the linear approximation. DS2 was split into 700 training and 300 testing pairs. The testing pairs were used to calculate the Regression Analytics Difference γ. In both cases, the EN trains the linear regression model in an on-line/incremental mode (through SGD) provided in Algorithm 1, with learning rate $\eta = 0.1$ (Bottou 2010). The idea is to demonstrate that even by not forwarding all sensed values from the SANs to the ENs, we can extract the *same* linear regression models with the baseline mechanism. And, more interestingly, *the prediction accuracy of the regression models based on our*

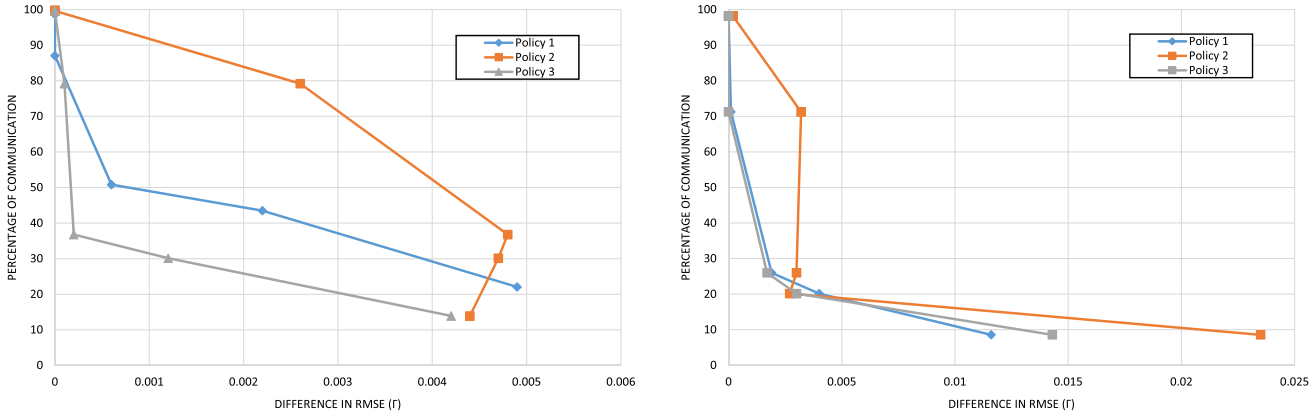

Fig. 12 Regression analytics difference for DS1; percentage of communication against *difference* in RMSE (i.e., γ metric) for *left* $\alpha = 0.5$, *right* TEEN

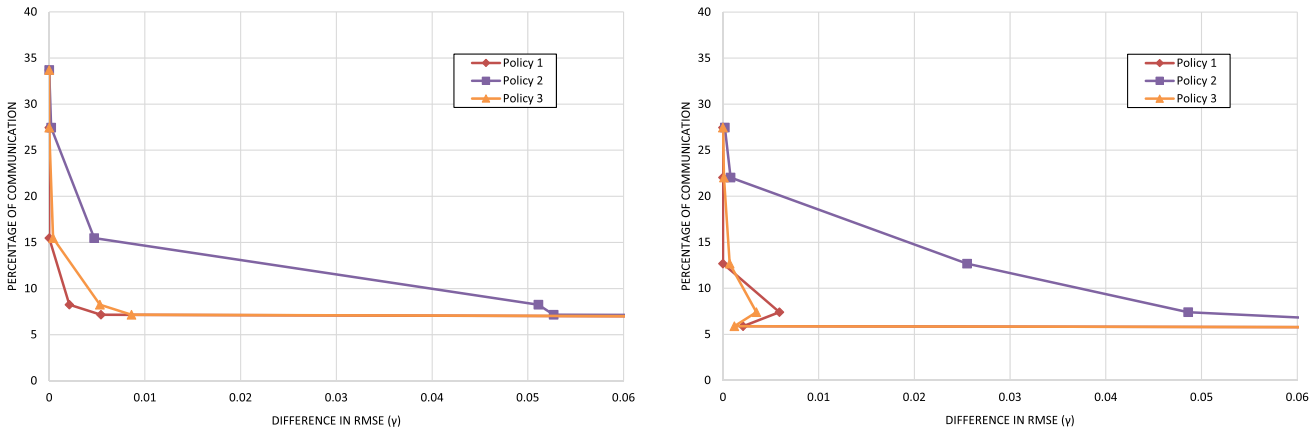

Fig. 13 Regression analytics difference for DS2; percentage of communication against *difference* in RMSE (i.e., γ metric) for *left* $\alpha = 0.5$, *right* $\alpha = 0.7$

Table 1 Overall performance of Policy 1, 2, 3 and Policy 1 with TEEN in SAN for re-construction, aggregation, and predictive analytics

	Re-construction	Aggregation analytics	Regression analytics
Policy 1	+	++	+
Policy 1 (TEEN)	+	+	– –
Policy 2	– – –	–	– –
Policy 3	++	++	+++

mechanism is very close to that of the regression models based on the baseline mechanism. In this case, IoT application and predictive analytics services can proceed with the same quality of analytics in a communication efficient way.

Figure 12 is representing the Regression Analytics Difference γ for $\alpha \in \{0.5, 1\}$ over all three different reconstruction policies inside the EN for DS1. Respectively, in Fig. 13 γ is shown for DS2 and $\alpha \in \{0.5, 0.7\}$. For DS2 it is not possible to illustrate TEEN as no change in γ occur with increasing θ. For better illustration the figures showing the Regression Analytics Difference only showing the θ values up to 0.06. Increasing θ over this threshold is increasing the error and decreasing the communication. For better readability of smaller values this is hidden in the figures above.

However, applicable from both Figures Policy 3 is creating the best trade-off between communication saving and regression analytics quality. This is independent on the chosen α value in the SAN as seen when comparing the left and right figures of each dataset. Moreover, it is applicable for DS1 that using our mechanism the same linear regression model is produced even with only 80% of the communication. Considering a communication saving of 50% around 0.0002 for γ needs to be tolerated from the IoT applications site for DS1 using $\alpha = 0.5$. For DS2 an identical regression can be produced by using only 15–20% of the complete communication, which is shown in Fig. 13. Considering these

both outputs, our proposed mechanism is increasing the lifetime of an edge network for predictive analytics tasks.

Overall, Table 1 summarizes the performance of our mechanism in re-construction, aggregation, and predictive analytics tasks and TEEN model adopted in SAN for Policy 1. We can conclude that Policy 3 is preferable to be adopted for predictive/regression analytics due to the smoothing component on the EN. That is, by adopting an exponential smoother in EN, the re-constructed values decreases the induced variance on the EN site. Hence, the linear regression approximation deals with less variance training pairs, which leads to a better representative of the regression plane. On the other hand, Policy 1 is recommended for re-construction or aggregation analytics since it retains the variance of the delivered contextual data stream, which plays significant role for aggregation functions. However, in the case of TEEN adopted in SAN for Policy 1, the reconstruction error is higher than the adoption of the exponential smoothing in SAN. Finally, Policy 2 exhibits poor performance in all analytics tasks due to the extreme generalization property of the average of the most recent values, especially, in the cases we encounter a significant number of undelivered values.

6 Conclusions and future work

We focus on the edge computing paradigm where pushing aggregation and predictive analytics to the edge of the IoT network allows the complexity of analytics tasks to be distributed into many smaller and more manageable pieces and to be physically located at the source of the contextual information. We introduce a lightweight, distributed, predictive intelligence mechanism that supports communication efficient aggregation and predictive modeling within the edge network of SANs and ENs. The mechanism is following the evolving nature of the multivariate time series (context vectors) based on the idea of locally deciding whether to deliver contextual data or not in light of minimizing the induced communication overhead and providing high quality analytics tasks. Based on splitting this intelligence into: prediction (through exponential smoothing) and decision making at the SANs and context re-instruction at ENs (by proposing three policies), we eliminate data transfer at the edge of the network, by exploiting the predictability of the captured contextual data. We provide fundamental theoretical analyses of the upper bounds of the reconstructed data quality and a comprehensive sensitivity analysis with the most important model parameter. We provide comprehensive comparative (theoretical and experimental) assessment with baseline solutions found in the literature and experimental evaluation of the proposed mechanism over two real multidimensional contextual datasets for aggregation and linear regression analytics tasks. We show the benefits stemmed from its adoption in edge computing environments and experiment with the trade-off between accuracy (quality) of edge analytics tasks and communication overhead. Our mechanism demonstrated its efficiency in supporting high quality of edge analytics by tolerating a relatively low error in light of decreasing significantly the communication overhead in an edge network.

Our future agenda includes investigating intelligent delay tolerant mechanisms for further minimizing the induced analytics errors in favor of saving communication. Moreover, future work is focused on certain modifications of our mechanism to support advanced analytics tasks including outliers detection, non-linear predictive models, and concept drifts in multidimensional contextual data streams in edge computing environments.

References

Abadi DJ, Carney D, Cetintemel U, Cherniack M, Convey C, Lee S, Stonebraker M, Tatbul N, Zdonik S (2003) Aurora: a new model and architecture for data stream management. VLDB J 12(2):120–139

Ahmad Y, Berg B, Cetintemel U, Humphrey M, Hwang J-H, Jhingran A, Maskey A, Papaemmanouil O, Rasin A, Tatbul N, Xing W, Xing Y, Zdonik S (2005) Distributed operation in the Borealis stream processing engine. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data (SIGMOD '05). ACM, New York, NY, USA, pp 882–884. doi:10.1145/1066157.1066274

Anagnostopoulos C, Hadjiefthymiades S, Georgas P (2012) PC3: principal component-based context compression. J Parallel Distrib Comput 72(2):155–170

Anagnostopoulos C, Hadjiefthymiades S, Katsikis A, Maglogiannis I (2014) Autoregressive energy-efficient context forwarding in wireless sensor networks for pervasive healthcare systems. Pers Ubiquitous Comput 18(1):101–114

Anagnostopoulos, C, Triantafillou P (2014) Scaling out big data missing value imputations: Pythia vs. Godzilla'. In: 20th ACM SIGKDD international conference on knowledge discovery and data mining (KDD '14), New York, pp 651–660

Anagnostopoulos C, Hadjiefthymiades S, Kolomvatsos K (2016) Accurate, dynamic, and distributed localization of phenomena for mobile sensor networks. ACM Trans Sensor Netw 12(2). doi:10.1145/2882966

Anagnostopoulos C, Hadjiefthymiades S (2014) Advanced principal component-based compression schemes for wireless sensor networks. ACM Trans Sensor Netw 11(1). doi:10.1145/2629330

Anagnostopoulos C, Anagnostopoulos T, Hadjiefthymiades S (2010) An adaptive data forwarding scheme for energy efficiency in wireless sensor networks. In: 5th IEEE international conference intelligent systems, London, pp 396–401

Anagnostopoulos C, Triantafillou P (2017a) Query-driven learning for predictive analytics of data subspace cardinality. ACM Trans Knowl Discov Data 11(4):46. doi:10.1145/3059177

Anagnostopoulos C, Triantafillou P (2015) Learning set cardinality in distance nearest neighbours. In: IEEE international conference on data mining (IEEE ICDM 2015), Atlantic City, pp 691–696

Anagnostopoulos C, Triantafillou P (2017b) Efficient scalable accurate regression queries in In-DBMS analytics. In: IEEE international conference on data engineering (ICDE), San Diego

Arasu A, Babu S, Widom J (2006) The CQL continuous query language: semantic foundations and query execution. VLDB J 15(2):121–142

Awang A, Suhaimi MH (2007) RIMBAMON©: A forest monitoring system using wireless sensor networks. In: International conference on intelligent and advanced systems 2007, Kuala Lumpur, pp 1101–1106. doi:10.1109/ICIAS.2007.4658555

Bottou L (2016) Large-Scale machine learning with stochastic gradient descent. In: Proceedings of the 19th international conference on computational statistics (COMPSTAT'2010), Springer, Paris, pp 177–187

Bottou L, Curtis FE, Nocedal J (2017) Optimization methods for large-scale machine learning. arXiv:1606.04838. [stat.ML]

Box GEP, Jenkins G (1990) Time series analysis, forecasting and control. Holden-Day, Incorporated

Cheng B, Papageorgiou A, Bauer M (2016) Geelytics: enabling on-demand edge analytics over scoped data sources. In: IEEE international congress on big data (BigData Congress), San Francisco, pp 101–108

Chowdappa VP, Botella C, Beferull-Lozano B (2015) Distributed clustering algorithm for spatial field reconstruction in wireless sensor networks. In: IEEE 81st vehicular technology conference (VTC Spring), Glasgow, pp 1–6

Chu D, Deshpande A, Hellerstein JM, Hong W (2006) Approximate data collection in sensor networks using probabilistic models. In: Proceedings of the 22nd international conference on data engineering (ICDE '06). IEEE Computer Society, Washington, DC, USA, p 48. doi:10.1109/ICDE.2006.21

Dallachiesa M, Jacques-Silva G, Gedik B, Wu K.-L, Palpanas T (2015) Sliding windows over uncertain data streams. Knowl Inf Syst 45(1):159–190. doi:10.1007/s10115-014-0804-5

De Vito S, Massera E, Piga M, Martinotto L, Di Francia G (2008) On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. Sens Actuators B Chem 129(2):750–757 **(ISSN 0925-4005)**

Durbin J, Jan Koopman S (2012) Time series analysis by state space methods. Oxford Statistical Science Series

Eidson GW et al (2010) The South Carolina digital watershed: end-to-end support for real-time management of water resources, vol 2010. In: Proc. 4th intl. symposium on innovations and real-time applications of distributed sensor networks (IRADSN 09), USA

Ganti R, Ye F, Lei H (2011) Mobile crowdsensing: current state and future challenges. IEEE Commun Mag 49(11):32–39

Gemulla R, Nijkamp E, Haas PJ, Sismanis Y (2011) Large-scale matrix factorization with distributed stochastic gradient descent. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11). ACM, New York, pp 69–77

Goel S, Imielinski T (2001) Prediction-based monitoring in sensor networks: taking lessons from MPEG. ACM SIGCOMM Comput Comm Rev 31(5):82–98

Gray J, Chaudhuri S et al (1997) Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub totals. Data Min Knowl Discov 1(1):29–53

Hentschel K, Jacob D, Singer J, Chalmers M (2016) Supersensors: Raspberry Pi devices for smart campus infrastructure. In: 4th IEEE international conference on future internet of things and cloud, FiCloud, Vienna, pp 58–62

Jiang H, Jin S, Wang C (2011) Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks. IEEE Trans Parallel Distrib Syst 22(6):1064–1071

Kang J et al (2013) High-fidelity environmental monitoring using wireless sensor networks. Article 67. In: Proc. 11th ACM conference on embedded networked sensor systems (SenSys '13), USA

Kejela G, Esteves RM, Rong C (2014) Predictive analytics of sensor data using distributed machine learning techniques. In: IEEE 6th International Conference on cloud computing technology and science, Singapore, 2014, pp 626–631. doi:10.1109/CloudCom.2014.44

Kim J-J et al (2010) Wireless monitoring of indoor air quality by a sensor network. Indoor Built Environ 19(1):145–150

Kolomvatsos K, Anagnostopoulos C, Hadjiefthymiades S (2017) Data fusion and type-2 fuzzy inference in contextual data stream monitoring. IEEE Trans Syst Man Cybern Syst 47(8):1839–1853. doi:10.1109/TSMC.2016.2560533

Kuhn M, Johnson K (2013) Applied predictive modeling. Springer, Berlin. doi:10.1007/978-1-4614-6849-3**(ISBN 9781461468493)**

Lane N, Miluzzo E, Lu H, Peebles D, Choudhury T, Campbell A (2010) A survey of mobile phone sensing. IEEE Commun Mag 48(9):140–150

Manjeshwar A, Agrawal DP (2001) TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In: Proceedings of the 15th international parallel and distributed processing symposium (IPDPS '01). IEEE Computer Society, Washington, DC, p 189

McConnell SM, Skillicorn DB (2005) A distributed approach for prediction in sensor networks. In: Proceedings of 1st international workshop on data mining in sensor networks as part of the SIAM International Conference on data mining, pp 28–37

Muth J (1960) Optimal properties of exponentially weighted forecasts. J Am Stat Assoc 55(290):299–306

Nguyen N et al (2010) A real-time control using wireless sensor network for intelligent energy management system in buildings. In: Proc. IEEE workshop on environmental energy and structural monitoring systems (EESMS 10), pp 87–92

Nittel S (2009) A survey of geosensor networks: advances in dynamic environmental monitoring. Sensors 9:5664–5678

Oliveira LM, Rodrigues JJ (2011) Wireless sensor networks: a survey on environmental monitoring. J Commun 6(2):143–151

Papithasri K, Babu M (2016) Efficient multihop dual data upload clustering based mobile data collection in wireless sensor network. In: 2016 3rd international conference on advanced computing and communication systems (ICACCS), Coimbatore, pp 1–6

Patroumpas K, Sellis T (2011) Maintaining consistent results of continuous queries under diverse window specifications. Inf Syst 36(1):42–61

Patroumpas K, Sellis T (2010) Multi-granular time-based sliding windows over data streams. Temporal representation and reasoning (TIME). In: 2010 17th international symposium, pp 146–153

Patroumpas K, Sellis T (2006) Window specification over data streams. In: Proc. international conference on current trends in database technology (EDBT'06). Springer, Berlin, pp 445–464

Satyanarayanan M et al (2015) Edge analytics in the internet of things. IEEE Pervasive Comput 14(2):24–31

Silberstein A, Braynard R, Filpus G, Puggioni G, Gelfand A, Munagala K, Yang J (2007) Data-driven processing in sensor networks. In: Proc. Conf, innovative data systems research (CIDR) 3rd Biennial Conference on innovative data systems research (CIDR) Jan 7–10, 2007, Asilomar, California, USA

Simoens P, Xiao Y, Pillai P, Chen Z, Ha K, Satyanarayanan M (2013) Scalable crowd-sourcing of video from mobile devices. In: Proceeding of the 11th annual international conference on mobile

systems, applications, and services (MobiSys '13). ACM, New York, pp 139–152
Simonetto A, Leus G (2014) Distributed maximum likelihood sensor network localization. IEEE Trans Signal Process 62(6):1424–1437
Stojmenovic I, Wen S (2014) The fog computing paradigm: scenarios and security issues. In: 2014 Federated conference on computer science and information systems, Warsaw, pp 1–8
The mobile-edge computing initiative. http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing **(Online)**
Tofallis C (2015) A better measure of relative prediction accuracy for model selection and model estimation. J Oper Res Soc 66(8):1352–1362
Tulone D, Madden S (2006) An energy-efficient querying framework in sensor networks for detecting node similarities. In: Proceedings of the 9th ACM international symposium on modeling analysis and simulation of wireless and mobile systems (MSWiM '06). ACM, New York, NY, USA, pp 191–300. doi:10.1145/1164717.1164768
Vulimiri A, Curino C, Godfrey PB, Jungblut T et al (2015) WANalytics: geo-distributed analytics for a data intensive world. In: Proceedings of the 2015 ACM SIGMOD international conference on management of data, pp 1087–1092
Xu G et al (2014) Applications of wireless sensor networks in marine environment monitoring: a survey. Sensors 14(9):16932–16954
Xu Y, Lee W-C (2003) On localized prediction for power efficient object tracking in sensor networks. In: Proceeding of the 23rd international conference on distributed computing systems workshops, 2003, pp 434–439. doi:10.1109/ICDCSW.2003.1203591
Yi S, Li C, Li Q (2015) A survey of fog computing: concepts, applications and issues. In: Proceedings of the 2015 workshop on mobile big data, pp 37–42
Zervas E et al (2011) Multisensor data fusion for fire detection. Inf. Fusion 12(3):1566–2535. Elsevier