DOI: https://doi.org/10.1016/j.bdr.2019.04.001

Anomaly Detection and Repair for Accurate Predictions in Geo-distributed Big Data

Roberto Corizzo[a,b], Michelangelo Ceci[a,b], Nathalie Japkowicz[c]

[a] *Department of Computer Science, University of Bari Aldo Moro - Bari, Italy*
[b] *National Interuniversity Consortium for Informatics (CINI) - Rome, Italy*
[c] *Department of Computer Science, American University - Washington D.C.*

Abstract

The increasing presence of geo-distributed sensor networks implies the generation of huge volumes of data from multiple geographical locations at an increasing rate. This raises important issues which become more challenging when the final goal is that of the analysis of the data for forecasting purposes or, more generally, for predictive tasks. This paper proposes a framework which supports predictive modeling tasks from streaming data coming from multiple geo-referenced sensors. In particular, we propose a distance-based anomaly detection strategy which considers objects described by embedding features learned via a stacked auto-encoder. We then devise a repair strategy which repairs the data detected as anomalous exploiting non-anomalous data measured by sensors in nearby spatial locations. Subsequently, we adopt Gradient Boosted Trees (GBTs) to predict/forecast values assumed by a target variable of interest for the repaired newly arriving (unlabeled) data, using the original feature representation or the embedding feature representation learned via the stacked auto-encoder. The workflow is implemented with distributed Apache Spark programming primitives and tested on a cluster environment. We perform experiments to assess the performance of each module, separately and in a combined manner, considering the predictive modeling of one-day-ahead energy

Email addresses: `roberto.corizzo@uniba.it` (Roberto Corizzo), `michelangelo.ceci@uniba.it` (Michelangelo Ceci), `japkowic@american.edu` (Nathalie Japkowicz)

production, for multiple renewable energy sites. Accuracy results show that the proposed framework allows reducing the error up to 13.56%. Moreover, scalability results demonstrate the efficiency of the proposed framework in terms of speedup, scaleup and execution time under a stress test.

Keywords: Anomaly Detection, Data Repair, Geo-distributed Big Data, Spatial Autocorrelation, Neural Networks, Gradient-boosting.

1. Introduction

Nowadays, we are witnessing the continuous growth of geo-distributed sensor networks in many application domains such as climate monitoring, ecological modeling, traffic data analysis, energy consumption/production monitoring. These geo-distributed sensor networks produce huge volumes of data at increasing rate. Managing and processing such data, generated from multiple geographical locations, raises important issues which become much more challenging when the final goal is that of the analysis of the data for forecasting purposes or, more generally, for predictive modeling.

One of the most recurrent problems in predictive modeling tasks involving sensor data is the presence of noise in the data. In fact, data acquired by the sensors may not be transmitted due to technical problems or may be affected by measurement errors. Semi-supervised approaches [1] respond to the need to work with missing or unlabelled data, while outlier detection [2] approaches respond to the need to identify data that significantly deviates from the expected trend.

However, such approaches are not particularly well-suited to online learning contexts with data coming in the form of streams from multiple sites. In addition, existing online predictive modeling approaches do not address the problem of missing or noisy data, but assume that data measured by sensors are free from anomalies. Hence, there is a clear separation between anomaly detection or repair approaches, and predictive modeling approaches. In this paper we present a framework that can handle these tasks simultaneously, that is, detecting anomalous data, repairing it and predicting values for a variable of interest and a time horizon of interest, in the context of online learning with data coming from multi-site geo-localized sensor data.

The contributions presented in this paper can be summarized as follows:

- *i*) An anomaly detection technique based on the distance between objects described by embedding features learned via a stacked auto-encoder;
- *ii*) A repair module which repairs the data detected as anomalous exploiting non-anomalous data measured by the sensors in nearby spatial locations;
- *iii*) A prediction module which leverages Gradient Boosted Trees (GBTs) to predict values assumed by a target variable of interest for the repaired newly arriving (unlabeled) data, with an input space that is described either by the original feature representation or the embedding feature representation learned via the stacked auto-encoder;

The anomaly detection task we consider is *unsupervised* and the anomalies we consider in this paper are contextual and based on a *local context* [3]. That is, anomalies are identified on a single geographic position on the basis of a comparison with the time series of data observed at that location (context). We will also show that our method provides reliable predictions in case of *diffused-context* contextual anomalies, where the context includes near-by geographic locations, in addition to time series of data observed at that location. We resort to stacked auto-encoders [4] because of their recognized ability to learn to reconstruct a given input representation with a low reconstruction error [5] and to be used for feature extraction [6]. In fact, the hidden levels of their architecture are usually defined with a lower number of neurons, thus representing a reduced dimensionality representation of the inputs, while the output level returns vectors of the same number of features of input data. This characteristic has been exploited to perform anomaly detection [7] [8] [9] relying on reconstruction error. The general idea behind this approach is that if the auto-encoder is trained with non-anomalous data, obtaining a high reconstruction error for a new instance at testing time means that it belongs to a different distribution than that of the non-anomalous instances, therefore it can be labeled as anomalous.

The feature extraction capability of auto-encoders, which allows to obtain a feature space of reduced dimensionality (see [6] and [10]) is preserved in deep neural networks such as stacked auto-encoders, as shown in recent works [11] [12]. In deep neural networks, hidden layers can be thought of latent structures incorporating features of increasing levels of abstraction, learned from unlabeled data. The whole set of hidden layers thus constitute a feature hierarchy of increasing complexity.

Both capabilities of Stacked auto-encoders are exploited in the framework we propose in this paper for learning embedding features and for data repair. Moreover, the individual components of the framework (auto-encoder training, anomaly detection, data repair, feature extraction, prediction) have been implemented in accordance with the distributed programming primitives for Apache Spark, with the purpose of enabling large-scale data processing in cluster computing [13], as well as to take advantage of data locality in geo-distributed sensor networks.

The paper is structured as follows. Section 2 reviews related work in the literature. Section 3 presents the proposed methods for each single task considered in our framework. Section 4 describes the experimental setting, the datasets and the experimental results. Finally, we draw the main conclusions and give directions for further work in Section 5.

2. Related Work

Our work is rooted in sensor data analysis and, more specifically, in anomaly detection, data repair and predictive modeling for sensor (network) data.

In anomaly detection, anomalies are classified in three categories: point anomalies, contextual anomalies and collective anomalies [14]. In this paper we focus on contextual anomalies, where the main goal is to identify anomalous data instances with respect to usual data coming from a specific context, which depends on the specific problem and structure of the data analyzed (e.g. spatio-temporal context).

One example of contextual anomaly is an abrupt value measurement observed in a temperature time series. More generally, it looks clear that contextual anomalies are naturally present in scenarios where spatio-temporal data are generated, as already explored in previous studies [15] [16] [17]. Contextual anomaly detection can be carried out with supervised, semi-supervised and unsupervised methods [14]. Despite the abundance of approaches based on classical supervised machine learning based models such as neural networks and support vector machines, unsupervised methods are of particular interest in many real-world scenarios where labels are missing. This problem is particularly important in the context of streams of sensor data, when anomaly detection has to be performed in real-time.

In this scenario, auto-encoders and stacked auto-encoders have demonstrated superior performance for different problems in the recent literature [5] [8] [9] [18]. This is theoretically justified since auto-encoders are able to construct representations with a low reconstruction error, based on non-linear combinations of input features [4].

Although auto-encoders have seen particular interest for anomaly detection from images [9] and videos [18], in this work we adopt such models and investigate their effectiveness for the detection of abrupt changes in multivariate time-series data.

Concerning data repair approaches for sensor data, although there is much interest motivated by the opportunity to improve the accuracy of predictive models, the approaches available in the literature are still at an initial stage with respect to the emerging and increasingly complex applications. Some approaches rely on smoothing the time series, by a linear interpolation/regression-based mechanism [19]. The study in [20] proposes an approach for cleaning RFID data in a mobile environment based on Bayesian inference, whereas the authors in [21] propose a method applied to photovoltaic data, which recognizes three types of outliers and repairs the time series with a time transform function which considers the rated capacity of a PV station. However, the major issue of smoothing approaches is that they also affect correct data points, compromising the overall data quality.

Other algorithms in the literature exploit the violation graph approach, which is based on user-defined rules, to perform data cleaning, such as the equivalence class algorithm [22] or the holistic data cleaning algorithm [23]. In [24], a stream data cleaning system for categorical and numerical data is proposed, which relies on compact data structures to maintain the necessary state to repair data. Dirty data are repaired using the concept of distributed violation graph, which is an extension of the violation graph approach, aimed to improve the scalability performances.

The study in [25] considers speed constraints to recognize abnormal spikes of values in a stream, and proposes the median principle to repair abnormal data by identifying the local optimum. The main limitation of constraints-based approaches is the intrinsic dependence on constraints, which could be domain-dependent and hard to define, especially in presence of concept drift. Moreover, both smoothing and constraints-based approaches lack the exploitation of the spatial autocorrelation[1] in data coming from geo-referenced sensors.

Shifting the focus on predictive models for streams of sensor data, a series of approaches have been recently developed for solving analytic problems in different domains. These approaches are, for instance, based on Bayesian regression [27], time series classification via logistic models [28] and convolutional neural networks [29].

Considering, specifically, predictive approaches tailored for smart grids, one of the most important predictive tasks is that of renewable energy forecasting for a network of plants. In this context, machine learning approaches typically

[1] According to the Tobler's first law of geography (1970), spatial autocorrelation is defined as "everything is related to everything else, but near things are more related than distant things" [26].

aim to find a mapping between historical and forecasted variables [30] [31]. Along the same stream of research, recently, there has been particular interest among methods that carry out the predictive task for multiple plants at the same time, with adaptive models that exploit the spatial information from neighboring plants [32] [33]. This approach has been proved beneficial in terms of predictive performance of the models. From the methodological perspective, some methods are based on neural networks [30], while others involve quantile regression [34], Gaussian processes [35], vector autoregression (VAR) [32] [36] or predictive clustering trees [33]. However, the recent adoption of Gradient Boosted Trees (GBTs) has shown promising results, outperforming many other approaches [37] [31].

3. Method

The scenario considered in this work considers multivariate sensor data observations (e.g. temperature, pressure, humidity, etc.), also known as *Geodata streams*, observed at regular time points, by sensors which are distributed in different geographic locations. More formally: let P be the set of locations where sensors collect data and $x_{t,p}$ be the vector of observations at time t and location $p \in P$. The multivariate geodata streams is then a sequence of sets $\langle \{x_{1,1}, \ldots, x_{1,|P|}\}, \{x_{2,1}, \ldots, x_{2,|P|}\}, \ldots \rangle$. This definition extends to a multivatiate context the definition of geodata stream provided in [38].

In this section we discuss in detail how the anomaly detection, repair, feature extraction and prediction tasks are performed on multivariate Geodata streams. Each task is implemented with programming primitives in Apache Spark and, thus, is carried out in a distributed manner, exploiting data locality on the different nodes as much as possible.

3.1. Anomaly detection

In our framework we consider stacked auto-encoders as a reference model to carry out the anomaly detection task. Stacked auto-encoders are neural networks consisting of various layers, which aim to reconstruct input data with the lowest possible reconstruction error. They can be considered in the branch of deep neural networks, thus they benefit from a noteworthy expressive power. In fact, the first layer of a stacked auto-encoder learns first level features (for example, edges in a picture), while the second layer learns second-level features (for example, edges occurring together to form contour or corner detectors), and so on. In general, deeper layers of the stacked auto-encoder will learn higher-level concepts. Each auto-encoder has an encoding function γ and a decoding function δ such that:

$$\gamma : \mathcal{X} \to \mathcal{F}, \qquad \delta : \mathcal{F} \to \mathcal{X}$$
$$\gamma, \delta = \arg\min_{\gamma,\delta} \|X - \delta(\gamma(X))\|^2 \tag{1}$$

A suitable way to learn a stacked auto-encoder consists in layer-wise back-propagation learning. In this way, the first layer is learned supplying raw data, which is transformed into new vectors (most often of lower dimensionality with respect to input data) using the activation function of the hidden neurons. Such vectors will represent the training data for the second layer, and the output of this layer will be a second level encoding of the input data. This process can be repeated to learn deeper layers. The final layer of the stacked auto-encoder can be a layer of the same size of input data, if the purpose is that of reconstructing the input, or a softmax layer, if the purpose is that of classification. Obviously, in our case, the final layer represents the decoding stage and, thus, has the same size of the input layer. A representation of an auto-encoder and a stacked-auto encoder is shown in Figure 1.

With one hidden layer, the encoding stage of an auto-encoder takes the input $\mathbf{x} \in \mathbb{R}^d = \mathcal{X}$ and maps it to an hidden representation $\mathbf{z} \in \mathbb{R}^p = \mathcal{F}$:

$$\mathbf{z} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2}$$

Where σ is a sigmoid or a rectified linear unit activation function, W is a weight matrix and b is a bias vector.

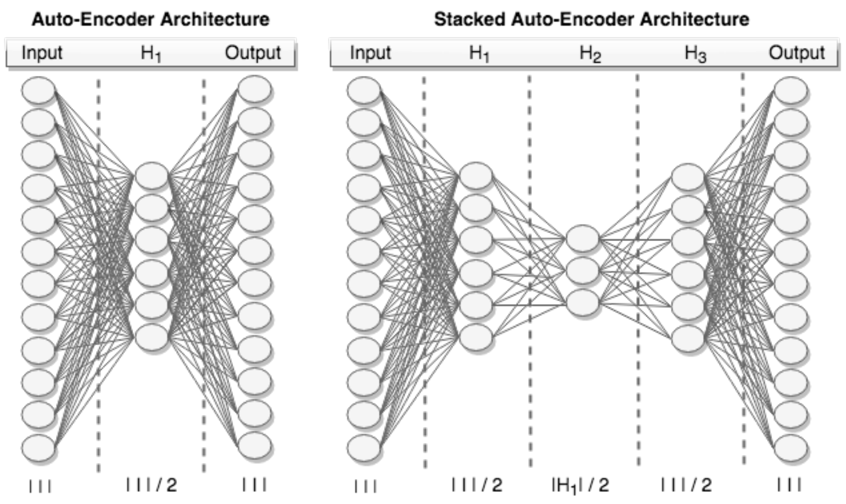

Figure 1: Architecture of a standard auto-encoder (left) and a stacked auto-encoder (right).

The decoding stage reconstructs x from z as:

$$\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{z} + \mathbf{b}) \tag{3}$$

such that the following loss is minimized:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2 \tag{4}$$

In this study, we propose two approaches: the former is based on the reconstruction error, whereas the latter is based on a k-Nearest Neighbors approach which works in the feature space defined by the deepest hidden layer of the auto-encoder.

For the case of anomaly detection approach based on reconstruction error, at each learning session, once the auto-encoder has been learned with training instances, a maximum distance threshold is calculated. This threshold represents a maximum error bound used to determine whether an instance belongs to the normal data distribution, or not. More specifically, if an instance has a reconstruction error greater than the threshold, it is considered as anomalous.

It is noteworthy that the threshold depends on the data distribution of the reconstruction errors. Since this distribution is subject to change over time, due to concept drift, we calculate this dynamically as: $[\overline{e} + 3 \cdot \sigma]$, according to a one-tailed 3-sigma rule [39], where $\overline{e}$ is the average reconstruction error of training instances and σ is the standard deviation of such errors. This feature can be of great practical value, since it is often difficult to select a distance threshold a-priori which allows us to perform anomaly detection tasks with good constant performance over time. The pseudo-code of the algorithms for the anomaly detection and automatic determination of the threshold are described in Algorithm 1 and Algorithm 2, in which we resort to some basic functions not described in the code:

- *learn*: Learns an auto-encoder model given a training dataset, a desired architecture, the maximum number of iterations allowed and the tolerance threshold;
- *encode*: Given a pre-learned auto-encoder model and a new instance with the feature representation of the original training dataset, it returns the embedding version of the instance in the deepest hidden layer of the auto-encoder model;

The auto-encoder architecture adopted consists of two hidden layers. The first one with a number of neurons equal to half the number of neurons of the input layer (independent features), and a second hidden layer with half neurons than the first hidden layer (see Figure 1).

The distributed implementation of stacked auto-encoders for Apache Spark adopted in this work is available at the following link: `https://github.com/avulanov/scalable-deeplearning`. This implementation exploits, in addition to the standard multilayer perceptron, new deep learning features that are still not available in the official Spark MLlib library.

For the latter alternative (k-Nearest Neighbors approach), once the auto-encoder is learned with available instances, the average distance between each instance and the nearest k instances in the hidden layer is calculated. When we have a new instance, this instance is encoded, and its average distance w.r.t. the nearest k instances in the hidden layer is calculated. If this distance is greater than a previously defined threshold, then this instance is considered as anomalous, otherwise it is considered as non-anomalous. This strategy is represented graphically in Figure 2 and 3.

In this case, the maximum distance allowed to classify an instance as non-anomalous is set equal to $[\overline{d} + 3 \cdot \sigma]$, where $\overline{d}$ is the average of the distances observed between each training instance and the nearest instances in the embedding, and σ is the standard deviation of the observed distances.

The distributed k-Nearest Neighbors implementation used in our framework is available at the following link: `https://github.com/saurfang/spark-knn`. It employs hybrid spill trees [40], a distributed data structure (variant of metric trees) for high-dimensional indexing, that allows to achieve high search efficiency. This aspect is particularly suited to the problem of k-Nearest Neighbors that, otherwise, could incur in computational bottlenecks.

The pseudo-code of the algorithms for the anomaly detection and automatic determination of threshold are described in Algorithm 3 and 4, in which we resort to some basic functions not described in the code:

- $getNeighbors$: Given a pre-learned k-NN model and a new instance, it returns its k-nearest instances;

- $dist$: Given two instances with the same feature representation, it returns the Euclidean distance between the two vectors;

3.2. Data repair

After identifying anomalous instances, by means of the anomaly detection module, it is important to address the data repair issue, since noisy data could affect the quality of subsequent tasks (e.g. prediction).

The approach proposed in this paper is to repair such anomalous instances by exploiting non-anomalous instances at nearby spatial locations. The moti-

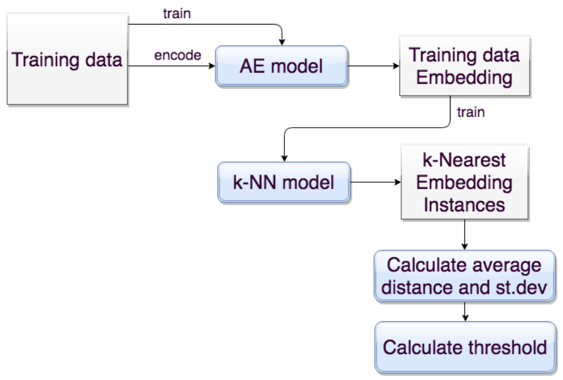

Figure 2: Anomaly detection (k-NN strategy): Learning phase.

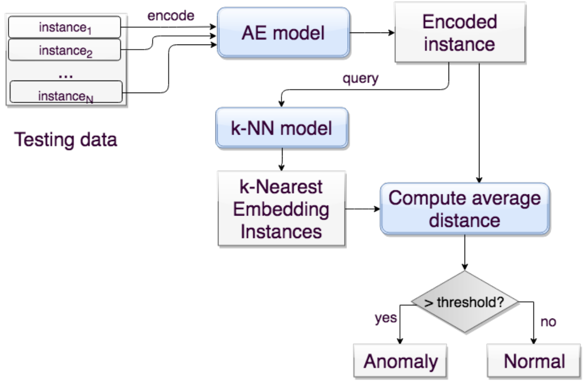

Figure 3: Anomaly detection (k-NN strategy): Anomaly detection phase.

```
Data: $T_R$ : Training data
      $maxIter$ : Maximum iterations for auto-encoder training
      $tol$ : Max tolerance for auto-encoder training
$model \leftarrow learn(T_R, maxIter, tol)$
forall $t_r \in T_R$ do
    $projTr = encode(t_r, model)$
    $recErr.add(MAE(t_r, projTr))$
end
$avgRec = recErr.average()$
$stDevRec = recErr.stdDev()$
$thresh = avgRec + (3 \cdot stDevRec)$
return $model, thresh$
```

Algorithm 1: Anomaly Detection Scheme (Reconstruction Error strategy): Learning stage

```
Data: $T_S$ : New data
      $model$ : Pre-learned auto-encoder model
      $thresh$ : Upper bound error threshold
forall $t_s \in T_S$ do
    $projTs = encode(t_s, model)$
    $error = MAE(t_s, projTs)$
    if $error > thresh$ then
        $A.add(t_s)$
    end
end
return $A$: Anomalous instances.
```
Algorithm 2: Anomaly Detection Scheme (Reconstruction Error strategy): Detection stage

```
Data: $T_R$ : Training data
      $k$ : Number of neighbors
      $maxIter$ : Maximum iterations for auto-encoder training
      $tol$ : Max tolerance for auto-encoder training
$model \leftarrow learn(T_R, maxIter, tol)$
forall $t_r \in T_R$ do
    $projTr = encode(t_r, model)$
    $trainEmb.add(projTr)$
end
$kNNmodel \leftarrow kNNtrain(trainEmb)$
forall $e \in trainEmb$ do
    $e_{NB} = getNeighbors(e, kNNmodel, k)$
    forall $nb \in e_{NB}$ do
        $distances.add(dist(e, nb))$
    end
end
$avgDist = distances.average()$
$stDevDist = distances.stdDev()$
$thresh = avgDist + (3 \cdot stDevDist)$
return $model, kNNmodel, thresh$
```
Algorithm 3: Anomaly Detection Scheme (k-NN strategy): Learning stage

```
Data: $T_S$ : New data
      $k$ : Number of neighbors
      $model$ : Pre-learned auto-encoder model
      $knnModel$ : Pre-learned $k$-NN model
      $thresh$ : Upper bound error threshold
$model \leftarrow learn(T_R, maxIter, tol)$
forall $t_s \in T_S$ do
    $projTs = encode(t_s, model)$
    $ts_{NB} = getNeighbors(projTs, kNNmodel, k)$
    $avgDistTS = 0.0$
    forall $nb \in ts_{NB}$ do
        $avgDistTS+ = dist(projTs, nb)$
    end
    $avgDistTS = \frac{avgDistTS}{k}$
    if $avgDistTS > thresh$ then
        $A.add(t_s)$
    end
end
return $A$: Anomalous instances.
```
Algorithm 4: Anomaly Detection Scheme (k-NN strategy): Detection stage

vations for this solution have roots in the concept of spatial autocorrelation. Spatial autocorrelation is the correlation among the values of a single variable (i.e., object property) strictly attributable to the relatively close position of objects on a two-dimensional surface, introducing a deviation from the independent observations assumption of classical statistics [41]. Intuitively, it is a property of random variables taking values, at pairs of locations a certain distance apart, that are more similar (positive autocorrelation) or less similar (negative autocorrelation) than expected for pairs of observations at randomly selected locations [42]. Positive autocorrelation is common in geophysical phenomena where temperature, wind speed, irradiance, etc. are similar at close locations.

For this purpose, we propose two data repair mechanisms:

- **Non-selective**: The entire instance $\mathbf{x}_{p,t}$ (all features) is repaired by exploiting the non-anomalous instances of other spatial sites by a weighted average. The weight is defined by a pairwise closeness function (in km) between the locations:

$$\mathbf{x'}_{(p,t)} = \frac{\displaystyle\sum_{p' \in N(p)} \left[\mathbf{x}_{(p',t)} \cdot \left(1 - \frac{dist(p,p')}{\max Dist(P)}\right) \right]}{\displaystyle\sum_{p' \in N(p)} \left(1 - \frac{dist(p,p')}{\max Dist(P)}\right)}, \tag{5}$$

where $N(p)$ represents the subset of neighboring locations of p, for which data instances are not detected as anomalous.

- **Selective**: For each feature v of an anomalous instance $\mathbf{x}$, it is detected whether the observed value $\mathbf{x}_{(p,t)}[v]$ is anomalous, querying for each site the historical values of that particular spatial location at the same time of the same month. Specifically, we consider for each plant p, the mean vector $\overline{\mathbf{x}}_{(\mathbf{p},\cdot)}$ and the standard deviation vector $\sigma_{(\mathbf{p},\cdot)}$. If the value of the feature v is out of the range $[\overline{\mathbf{x}}_{(p,\cdot)}[v] - 3 \cdot \sigma_{(p,\cdot)}[v],\ \overline{\mathbf{x}}_{(p,\cdot)}[v] + 3 \cdot \sigma_{(p,\cdot)}[v]]$, the value of feature v is repaired by exploiting the value of the same feature in the spatial locations $p' \in N(p)$, representing neighboring sites whose instances have not been identified as anomalous at time point t. The replaced value is calculated as a weighted average, where the weight is defined by a pairwise closeness function (in km) between the locations:

$$\mathbf{x'}_{(p,t)}[v] \leftarrow \frac{\displaystyle\sum_{p' \in N(p)} \left[\mathbf{x}_{(p',t)}[v] \cdot \left(1 - \frac{dist(p,p')}{\max Dist(P)}\right) \right]}{\displaystyle\sum_{p' \in N(p)} \left(1 - \frac{dist(p,p')}{\max Dist(P)}\right)} \tag{6}$$

3.3. Feature extraction

The framework, as already anticipated, allows us to extract features exploiting exclusively the encoding function of the auto-encoder learned in the previous steps. In more detail, given an auto-encoder with two hidden layers, the encoding process allows us to encode the input data I, with $|I|$ features, in a new

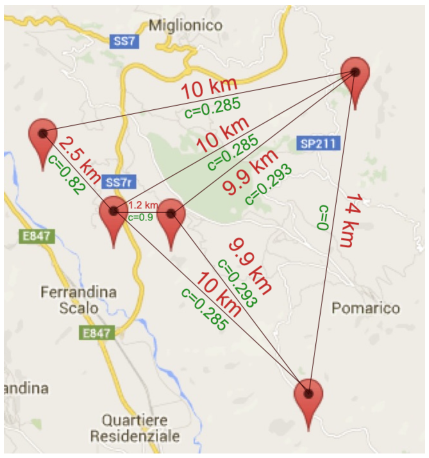

Figure 4: Example of geo-distributed locations with corresponding pairwise distances in kilometers and closeness measure proposed for the data repair step ($maxDist = 14$ km).

feature space H_1 of dimensionality $\frac{|I|}{2}$ and, subsequently, in a new feature space H_2 of dimensionality $\frac{|H_1|}{2}$. This because in this study we exploit the second hidden layer of the stacked auto-encoder in order to achieve a new feature space at a higher level of abstraction and reduced dimensionality than the input feature representation. The main purpose is that of mitigating the collinearity effect between features [43] [44] and, consequently, improve prediction effectiveness.

The feature extraction step is represented in Figure 5. Section 4 includes a comparison of results in terms of prediction error, using the original feature space and the reduced feature space, obtained using the feature extraction module.

3.4. Prediction

This module allows the prediction of a target variable of interest for new data, given the independent (input) feature values, previously processed through the anomaly detection, repair and, optionally, feature extraction processes. The prediction of the target variable of interest is obtained for multiple geographical locations and for multiple time points in the future. An example, taken for the experiments we present in Section 4, concerns the prediction of the energy produced the next day by multiple renewable energy plants, given the forecasted

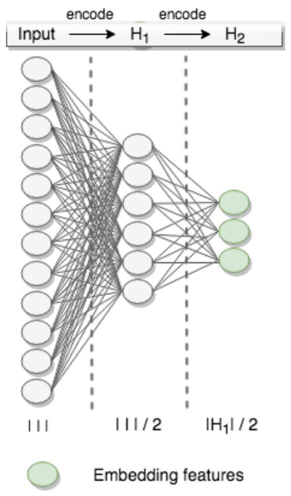

Figure 5: Feature extraction process, performed using the encoding function of the trained auto-encoder.

weather conditions of the day of interest. Data provided to the predictive module can be represented according to the original feature space of the input data (as depicted in Figure 6) or according to the embedding features extracted with the feature extraction module described above (as depicted in Figure 7).

In our framework, we adopt Gradient Boosted Trees (GBTs) as prediction model, which is an ensemble of decision trees, iteratively learned minimizing a loss function. At each iteration, the algorithm uses the current ensemble to predict the target attribute value of each training instance, and then compares the prediction with the actual target attribute value. The dataset is then relabeled, according to a specified loss function, in order to help the next decision tree improve the performance of the ensemble, taking into account previous mistakes (poor predictions). In fact, after each iteration, GBTs reduce this loss function on the training data. Like decision trees, GBTs present the advantage of handling categorical features. Moreover, they do not require normalization for each data feature, and they are able to learn from non-linear interactions between independent and dependent variables.

GBTs have been chosen since they demonstrated exceptional performances

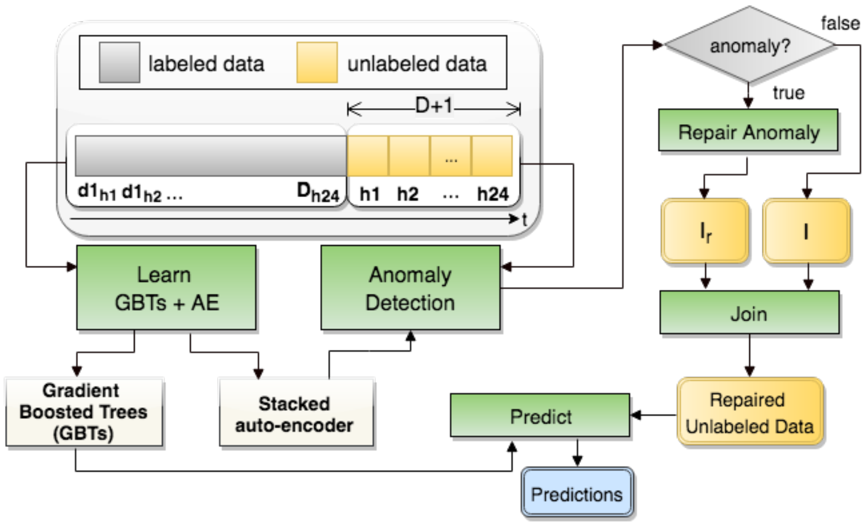

Figure 6: Proposed framework (basic formulation). At training time, a stacked auto-encoder and Gradient Boosted Trees (GBTs) models are trained on historical data. When new data arrive, the detection module performs anomaly detection on these data exploiting the learned auto-encoder model and using either the reconstruction error strategy or the k-NN strategy. The repair module performs data repair adopting either the non-selective or the selective strategy. The repaired anomalous instances and the non-anomalous instances are joined and given as input to the prediction module, which exploits the previously trained GBTs model to extract predictions.

in predictive modeling tasks. This has been shown, in particular, also in the context of energy forecasting [37] [31], which is the reference use case considered in this work. The distributed GBTs implementation used in this work is that available in the Apache Spark Mllib library[2].

3.5. Geo-distributed data processing

One important aspect in geo-distributed contexts is the consideration of how the data processing strategy affects the overall execution time of distributed jobs, in presence of heterogeneous computational resources and network links.

Although the focus of this paper is on the specific analytic scenario, inspired

[2]`https://spark.apache.org/docs/latest/mllib-ensembles.html`

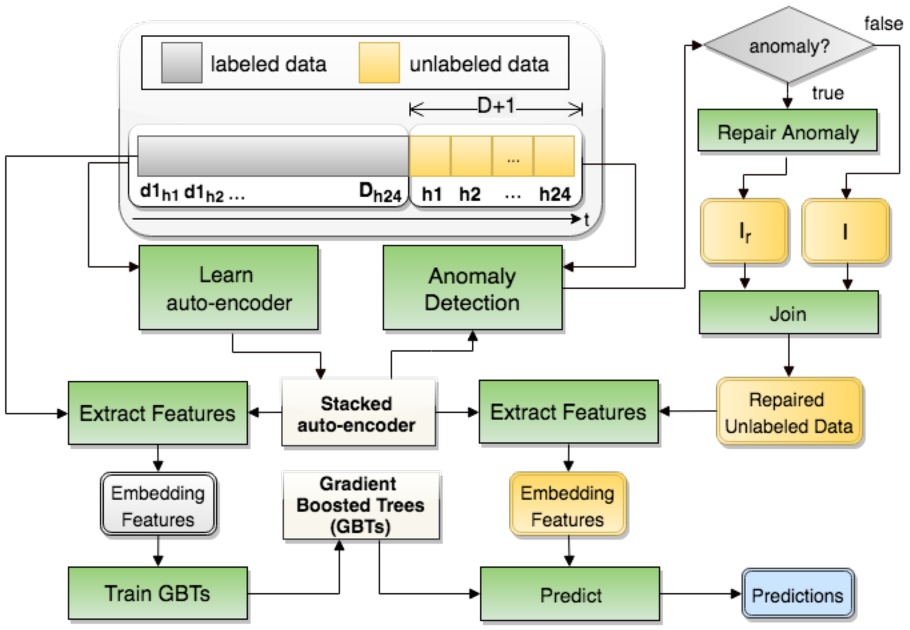

Figure 7: Proposed framework (including the feature extraction module). At learning time, the stacked auto-encoder model is learned with historical training data and the Gradient Boosted Trees (GBTs) model is trained on historical data in the feature space derived from the embedding features of the auto-encoder model. The repair module performs data repair as in the basic framework formulation. Once repaired instances are joined with non-anomalous instances, the embedding feature representation of the data is derived exploiting the auto-encoder model trained before. New data in the embedding feature representation is given as input to the prediction module, which returns the predictions extracted by the previously trained GBTs model.

by the works in [45] and [46], we exploit data fragmentation as much as possible in the deployment of our methodology. In this respect, we exploit one characteristic of our context, that is, the fact that data are directly observed by sensors at each site, and at the same time granularity. This means that data partitions are naturally balanced.

We now discuss the impact of data processing in the geo-distributed context for each step of the methodology proposed.

Considering the anomaly detection step, this is carried out on time-based aggregated historical data (aggregation is performed at one hour granularity in our experiments) collected on all geographical locations, hence the data required for this step are obtained as the union of data partitions. However, this union is only logical (see Figure 8). Technically speaking, the Spark DataFrame is built by considering data locality, that is, data are collected and processed by computational nodes "close" to sensors. The DataFrame, if used for training, is then directly used for learning autoencoders. The same happens in the (anomaly) detection step, which operates lazily in the Apache Spark framework, meaning that it does into incur in data partition shuffle, which would result in a dramatic network overload.

As for data repair, in contrast with [45] and [46], where computation can be carried out independently on different partitions, with a final aggregation step on the top level cluster (the operations performed are: word count, inverted index construction and arithmetic mean), our method introduces additional complexity in terms of the data required for performing the analytics task at hand. More specifically, this step requires non-anomalous data observed in multiple geographical locations, in order to repair anomalous data in a specific location (see Formulas (5) and (6)). However, data repair is performed on new data only and new data, as historical data, are aggregated at each location. For example, in the case study reported in Section 4, spot observations (1-15 minutes measurements) are averaged on a hourly basis and for each new day for which we have to predict the produced energy, the repair step needs to process 24 instances for each location, which results in a light network traffic, and does not affect the overall execution time.

As for the feature extraction step, the auto-encoder model trained during the anomaly detection step is broadcast and available for all the computational nodes. The model, which represents the only input to perform feature extraction, is a light-weight data structure of double precision weights. Thus, features can be extracted locally on each location, without impacting in data transfer via the network infrastructure.

Finally, considering the predictive task, it exploits either repaired data in its original feature representation, or repaired data obtained after feature extraction. The dataset required for this step is, as in the anomaly detection step, the (logical) union of data partitions available on computational nodes. Similarly to the phase of stacked auto-encoders learning, the phase of GBTs learning (which is distributed) uses a Apache Spark DataFrame which is built by considering data locality. The same happens in the prediction step, which operates lazily in the Apache Spark framework (this step only requires the usage of GBTs for associating a prediction to a new example).

4. Experiments

The datasets considered consist of a set of weather variables (such as temperature, humidity, etc.) monitored at hourly granularity by sensors placed on renewable energy plants, located in different geographical areas. Once the weather observations (independent variables) of the testing day have been repaired, the task consists in predicting the production of each renewable energy plant for the next 24 hours, at a hourly granularity.

In this work, two datasets are considered: PV Italy and Wind NREL:

- **PV Italy.** The data are collected at regular intervals of 15 minutes (measurements start at 2:00 and stop at 20:00 every day) by sensors located on 17 plants in Italy. The time period spans from January 1$^{\text{st}}$, 2012 to May 4$^{\text{th}}$, 2014. More details about data preparation steps performed on this dataset can be found in [33].

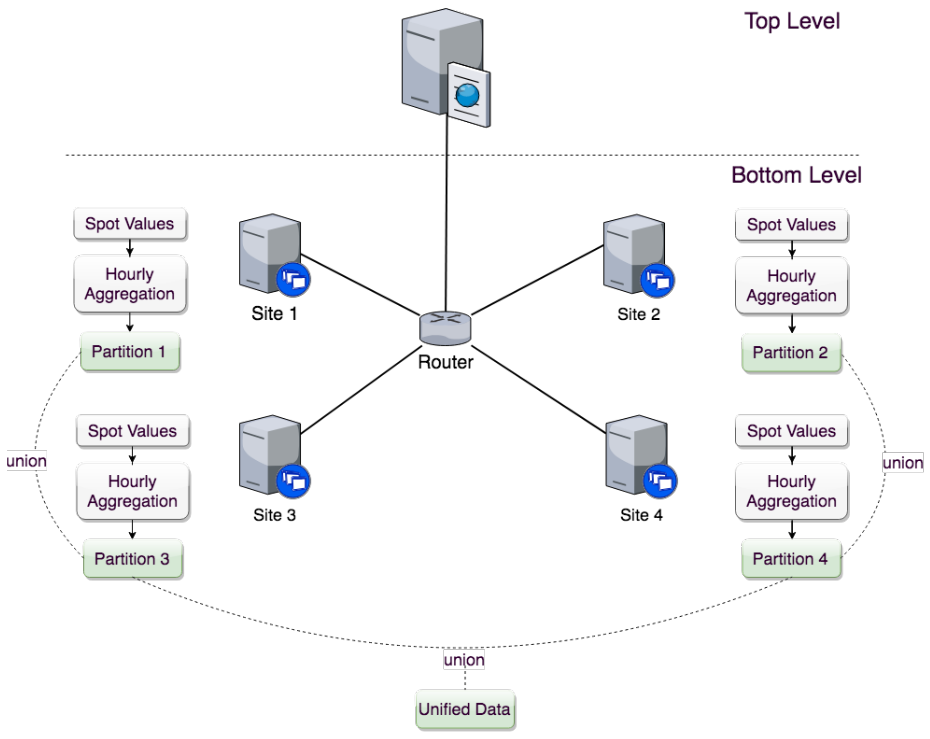

Figure 8: Outline of the geo-distributed data processing environment with hierarchical model.

- **Wind NREL.** This dataset was modeled by 3TIER using the Weather Research & Forecasting (WRF) model (`https://www.nrel.gov/wind/`). Five plants with the highest rated production have been selected, obtaining the time series of wind speed and production observed every 10 minutes, for a time period of two years (from January 1st, 2005 to December 31st, 2006). Hourly aggregation was performed. The data was not affected by outliers or missing values.

For both the datasets the following input features are represented: latitude, longitude of the *i*-th plant; day and hour, respectively; altitude and azimuth; plant ID; weather parameters, such as ambient temperature, irradiance, pressure, wind speed, wind bearing, humidity, dew point, cloud cover, descriptive weather summary. Weather parameters are either measured (training phase) or forecast (testing phase).

Weather data are extracted from Forecast.io (`http://forecast.io/`), the expected altitude and azimuth are extracted from SunPosition (`http://www.susdesign.com/sunposition/index.php`), whereas the expected irradiance (PV Italy dataset only) is extracted from PVGIS (`http://re.jrc.ec.europa.eu/pvgis/apps4/pvest.php`).

In order to evaluate the two anomaly detection strategies, the two data repair strategies and the prediction module described above, experiments have been conducted by injecting different levels of noise. We quantify noise according to two dimensions: *Instance Noise Rate* (INR), which defines the rate of instances affected by noise in the data window considered, and *Feature Noise Rate* (FNR), which defines the rate of features affected by noise for each anomalous instance. We consider two configurations in order to evaluate the sensitivity of the proposed framework with respect to different levels of noise:

- $N1 = \{INR = 0.25, FNR = 0.50\}$;
- $N2 = \{INR = 0.50, FNR = 0.50\}$.

The two perturbations are performed simultaneously. That is, in $N1$, 25%

of instances have been perturbed on 50% of the features. The way noise is introduced into data instances follows two different approaches: *i)* anomalies with local context (default approach), which correspond to random selection of locations to be perturbed *ii)* anomalies with diffused context, which correspond to anomalies introduced at a specific geographical area which contains several sensors/locations. While the first approach corresponds to the main setting of this study, the second case corresponds to diffused weather events and will be discussed only in Section 4.4.

This data perturbation is necessary in order to evaluate:

- Anomaly detection performances with the two strategies proposed in this paper (Reconstruction Error based vs. k-NN based);
- Data repair performances with the two strategies (Non Selective vs Selective);
- Predictive performance for the one-day-ahead renewable energy forecasting task, exploiting two feature representations of data (original and extracted using auto-encoder models), assessing the contribution of the repair step in terms of forecasting error reduction for the task at hand.

We conduct a sensitivity analysis considering three different sliding window sizes: 30, 60 and 90 days. For each dataset, we select randomly 10% of days, which are considered as testing days. For each day, we train the model using historical data (30, 60 or 90 days) preceding the testing day considered, and the goal is to perform prediction on the current day.

In order to evaluate the effectiveness of our approach in terms of anomaly detection (also considering the impact on the subsequent repair and predictive modeling tasks) we compared it with a baseline approach. The baseline approach considered in this work detects whether a data instance is anomalous, based on the consensus of neighboring locations. In case the consensus exists, the data instance will not be repaired, otherwise it will be subject to selective repair. In principle, the main property of this (spatially-aware) baseline approach is that it is able to not consider as anomalous weather events that do not significantly change the data distribution in the current location and in nearby locations as well.

4.1. Anomaly detection

The training procedure, which follows the minimization of the reconstruction error using training data (non-anomalous instances), stops if the maximum number of iterations is reached (800) or a stopping criterion is reached ($tol < 10e - 5$).

Concerning the anomaly detection strategy based on k-Nearest Neighbors, we perform a grid search to identify the best value for the k parameter in the following set of choices: $k = \{5, 10, 50, 100, 150, 200\}$. Anomaly detection results in terms of Precision, Recall and F-Score using the reconstruction error strategy and the k-NN strategy are reported in Table 1 and Table 2 (see Figures 9 and 10 from a more compact view).

The performances of the anomaly detection module leveraging the k-Nearest Neighbors (k-NN) strategy are more robust than the strategy based on reconstruction error, especially as noise increases in input data. In particular, results in terms of F-Score for the k-NN strategy are significantly better for the PV Italy dataset, and slightly better for the Wind NREL dataset. Moreover, the k-NN strategy always outperforms the reconstruction error strategy in terms of Recall, especially with high noise rate. It is important to highlight that low Recall impacts in a high amount of false negatives, which affect the subsequent repair step in the framework. More specifically, since the repair module needs to exploit instances identified as non-anomalous in order to repair instances identified as anomalous, if an high number of anomalous instances are wrongly classified as non-anomalous, this will reduce the repair capability for the anomalous instances, resulting in a high repair error. Therefore, an anomaly detection strategy with high Recall performances is strongly favorable in our scenario.

Concerning the configuration of the k parameter in the $k-$NN strategy, it can be observed that small values of k obtain the best results for the PV Italy

Table 1: Anomaly detection results for the PV Italy dataset (anomalies with local context) with the two variants of the proposed framework and the baseline method, using different distance thresholds ($T \in 15, 30, 45$), considering varying noise rates and training window sizes. Best results in terms of F-Score for each Window size configuration are marked in bold.

Noise rate	**Window size**								
$\{INR = 0.25,$ $FNR = 0.50\}$	30 days			60 days			90 days		
Reconstruction error variant	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
	0.9923	0.7303	0.8364	0.9936	0.7288	0.8362	0.9919	0.7297	0.8356
k-NN variant	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
k=5	0.8765	0.8358	**0.8458**	0.8819	0.8434	0.8532	0.8729	0.8384	0.8471
k=10	0.8912	0.8254	0.8442	0.8962	0.8404	**0.8563**	0.8910	0.8373	**0.8528**
k=50	0.9248	0.7729	0.8250	0.9161	0.7948	0.8344	0.9118	0.7965	0.8337
k=100	0.9517	0.7528	0.8261	0.9323	0.7687	0.8260	0.9228	0.7711	0.8236
k=150	0.9651	0.7427	0.8277	0.9451	0.7553	0.8246	0.9349	0.7600	0.8229
k=200	0.9733	0.7365	0.8291	0.9543	0.7487	0.8256	0.9445	0.7537	0.8239
Baseline method	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
$T = 15$	0.6787	0.5048	0.5008	0.6787	0.5048	0.5008	0.6787	0.5048	0.5008
$T = 30$	0.9814	0.7401	0.8336	0.9814	0.7401	0.8336	0.9814	0.7401	0.8336
$T = 45$	0.9997	0.7227	0.8379	0.9997	0.7227	0.8379	0.9997	0.7227	0.8379

Noise rate	**Window size**								
$\{INR = 0.50,$ $FNR = 0.50\}$	30 days			60 days			90 days		
Reconstruction error based	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
	0.9877	0.4575	0.6084	0.9890	0.4562	0.6080	0.9886	0.4560	0.6080
k-NN based	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
k=5	0.8225	0.6996	**0.7104**	0.8271	0.7148	**0.7245**	0.8201	0.7134	**0.7217**
k=10	0.8392	0.6606	0.6826	0.8404	0.6881	0.7053	0.8375	0.6882	0.7041
k=50	0.8928	0.5504	0.6202	0.8741	0.5864	0.6389	0.8696	0.5909	0.6402
k=100	0.9315	0.5041	0.6076	0.9023	0.5382	0.6166	0.8928	0.5458	0.6185
k=150	0.9523	0.4832	0.6050	0.9200	0.5134	0.6085	0.9106	0.5221	0.6104
k=200	0.9644	0.4711	0.6045	0.9338	0.4988	0.6056	0.9238	0.5073	0.6069
Baseline method	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
$T = 15$	0.7421	0.5788	0.6109	0.7421	0.5788	0.6109	0.7421	0.5788	0.6109
$T = 30$	0.9973	0.4462	0.6085	0.9973	0.4462	0.6085	0.9973	0.4462	0.6085
$T = 45$	1.0000	0.4448	0.6093	1.0000	0.4448	0.6093	1.0000	0.4448	0.6093

Table 2: Anomaly detection results for the Wind NREL dataset (anomalies with local context) with the two variants of the proposed framework and the baseline method, using different distance thresholds ($T \in 15, 30, 45$), considering varying noise rates and training window sizes. Best results in terms of F-Score for each Window size configuration are marked in bold.

Noise rate	**Window size**								
$\{INR = 0.25,$ $FNR = 0.50\}$	30 days			60 days			90 days		
Reconstruction error based	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
	0.9962	0.7514	0.8554	0.9954	0.7539	0.8558	0.9955	0.7526	0.8555
k-NN based	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
k=5	0.8992	0.7790	0.8223	0.9050	0.7876	0.8285	0.9072	0.7888	0.8305
k=10	0.9289	0.7780	0.8340	0.9256	0.7855	0.8363	0.9301	0.7892	0.8401
k=50	0.9768	0.7586	0.8478	0.9585	0.7664	0.8421	0.9590	0.7700	0.8439
k=100	0.9930	0.7547	0.8548	0.9781	0.7559	0.8474	0.9772	0.7647	0.8503
k=150	0.9970	0.7526	0.8564	0.9874	0.7529	0.8511	0.9884	0.7581	0.8537
k=200	0.9982	0.7521	0.8568	0.9955	0.7543	**0.8561**	0.9928	0.7566	**0.8553**
Baseline method	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
T = 15	0.9996	0.7505	0.8570	0.9996	0.7505	0.8570	0.9996	0.7505	0.8570
T = 30	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**
T = 45	1.0000	0.7500	0.8571	1.0000	0.7500	0.8571	1.0000	0.7500	0.8571

Noise rate	**Window size**								
$\{INR = 0.50,$ $FNR = 0.50\}$	30 days			60 days			90 days		
Reconstruction error variant	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
	0.9933	0.5051	0.6651	0.9935	0.5058	0.6654	0.9931	0.5068	0.6656
k-NN variant	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
k=5	0.8705	0.5992	**0.6669**	0.8762	0.6120	**0.6761**	0.8673	0.6144	**0.6754**
k=10	0.9056	0.5789	0.6654	0.8993	0.5922	0.6711	0.8915	0.5963	0.6707
k=50	0.9650	0.5243	0.6603	0.9501	0.5413	0.6621	0.9370	0.5476	0.6600
k=100	0.9876	0.5100	0.6644	0.9766	0.5217	0.6643	0.9658	0.5301	0.6628
k=150	0.9947	0.5057	0.6661	0.9854	0.5147	0.6652	0.9801	0.5180	0.6640
k=200	0.9971	0.5031	0.6662	0.9898	0.5102	0.6654	0.9873	0.5122	0.6648
Baseline method	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
T = 15	0.9979	0.5022	0.6663	0.9979	0.5022	0.6663	0.9979	0.5022	0.6663
T = 30	1.0000	0.5000	0.6667	1.0000	0.5000	0.6667	1.0000	0.5000	0.6667
T = 45	1.0000	0.5000	0.6667	1.0000	0.5000	0.6667	1.0000	0.5000	0.6667

dataset $k = 5, 10$). The same can be observed for the Wind NREL dataset, but only with the high noise rate setting, whereas with smaller noise rate, high values of k are preferred (k=200).

Tables 1 and 2 also report the comparison with the spatially-aware baseline method. The results obtained in terms of Precision, Recall and F-Score show that our k-NN based anomaly detection approach performs favorably in terms of F-Score in the majority of cases (9 of the 12 configurations obtained considering the 2 datasets, the 3 window size values and the 2 noise rates). Moreover, it is always capable to obtain a much higher Recall, which, as discussed before, is a highly preferred property. In fact, low Recall impacts in a high amount of false negatives, which affect the subsequent repair step in the framework. These results are motivated by two aspects: 1) Our method uses, as features, the spatial coordinates of the considered sites, this means that spatial distance is still (although indirectly) considered. 2) This analysis investigates the effectiveness of the anomaly detection step in presence of anomalies with local context in the data; in Section 4.4, we will discuss the case of anomalies with diffused context.

4.2. Data repair

The results in Table 3 and Table 4 show the performances in terms of RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) for the non-selective and selective repair schemes, considering the best value of k obtained for the anomaly detection task.

Comparing the results obtained with the two repair strategies we can see that the selective repair strategy is capable to obtain a consistent margin of improvement (lower RMSE) than the non-selective one for the PV Italy dataset, and similar (or slightly better) performances for the Wind NREL dataset. The motivation can be found in the higher degree of freedom of the selective approach, that can correct single feature values and does not try to repair the whole data instance when some feature values are possibly correct.

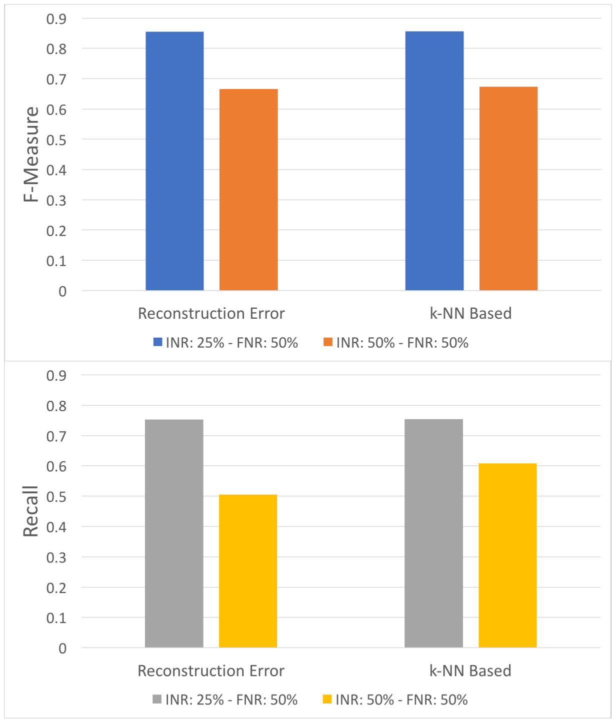

Figure 9: Anomaly Detection accuracy results obtained with two variants (Reconstruction Error and k-NN based) and two configurations of Instance Noise Rate (INR) and Feature Noise Rate (FNR). The results are represented in terms of F-Score and Recall for the PV Italy dataset (anomalies with local context). For k-NN based anomaly detection, the average (with different values of k) is reported.

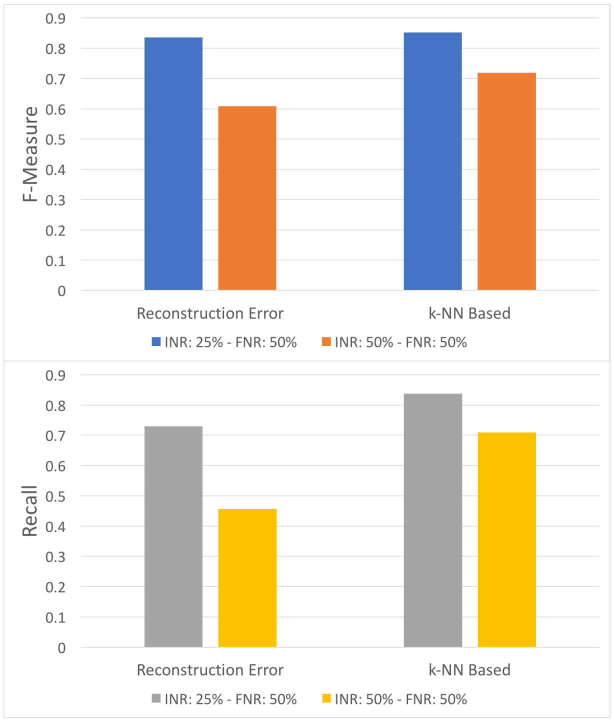

Figure 10: Anomaly Detection accuracy results obtained with two variants (Reconstruction Error and k-NN based) and two configurations of Instance Noise Rate (INR) and Feature Noise Rate (FNR). The results are represented in terms of F-Score and Recall for the Wind NREL dataset (anomalies with local context). For k-NN based anomaly detection, the average (with different values of k) is reported.

Table 3: Repair results for the PV Italy dataset (anomalies with local context), carried out with the two variants proposed, considering varying noise rates and training window sizes. The best value of the k parameter in the anomaly detection task adopting the k-NN strategy is depicted for each window size.

Noise rate	**Window size**					
$\{INR = 0.25,$ $FNR = 0.50\}$	30 days		60 days		90 days	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.0406	0.0179	0.0393	0.0170	0.0401	0.0172
(best k)	5		10		10	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.0329	0.0120	0.0334	0.0122	0.0344	0.0125
(best k)	5		10		10	
Noise rate	**Window size**					
$\{INR = 0.50,$ $FNR = 0.50\}$	30 days		60 days		90 days	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.0692	0.0292	0.0678	0.0294	0.0676	0.0294
(best k)	5		5		5	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.0583	0.0204	0.0563	0.0198	0.0562	0.0198
(best k)	5		5		5	

Table 4: Repair results for the Wind NREL dataset (anomalies with local context), carried out with the two variants proposed, considering varying noise rates and training window sizes. The best value of the k parameter for the anomaly detection task adopting the k-NN strategy is depicted for each window size.

Noise rate	**Window size**					
$\{INR = 0.25,$ $FNR = 0.50\}$	30 days		60 days		90 days	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.0384	0.0155	0.0377	0.0153	0.0367	0.0150
(best k)	200		200		200	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.0384	0.0155	0.0376	0.0152	0.0366	0.0149
(best k)	200		200		200	
Noise rate	**Window size**					
$\{INR = 0.50,$ $FNR = 0.50\}$	30 days		60 days		90 days	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.0553	0.0242	0.0533	0.0236	0.0521	0.0231
(best k)	5		5		5	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.0540	0.0226	0.0520	0.0220	0.0514	0.0217
(best k)	5		5		5	

4.3. One-day-ahead renewable energy forecasting

Results for the prediction task using Gradient Boosted Trees (GBTs) with non repaired data, repaired data, and repaired data after feature extraction are reported in Table 6 and 7 (see Figures 11 and 12 for a compact view).

The performance of GBTs for the multi-plant predictive modeling of one-day ahead energy production , using repaired data, allows the framework to gain, in average, up to 4.98% in terms reduction of the prediction error (RMSE). This improvement reaches 13.56% in the case of the embedding feature representation, when working on noisy testing data. In general, it can be observed that the benefit of using the repair scheme, in terms of error reduction, increases when the noise rate increases.

The feature extraction strategy allows to achieve a consistent RMSE reduction with the Wind NREL dataset, on which the extracted feature space performs better than the original feature space representation in all 12 configurations, whereas on the PV Italy dataset it obtains better performances in just 2 over 12 configurations.

In order to better clarify the real contribution of each single step of the framework, in Table 5 we report the results of the Wilcoxon Signed Rank tests. The results show that both the repair strategy and the feature extraction are clearly beneficial.

4.4. Anomalies with diffused context

In this section, we address a different scenario in which the anomalies are not observed randomly and at a specific location, but contextually distributed over a geographic area. This can happen in reality in presence of rare weather events, such as windstorms, thunderstorms, etc. For this purpose, we introduced noise in the data, according to the "diffused" approach, that operates in the following way: for each testing day, a number of instances are randomly selected as seeds, and the instances observed at neighboring locations in the same day and hour are perturbed. The features perturbed are the following: $temperature, windspeed, windbearing, pressure$. In order to model anomalies, once a seed is selected, the standard deviation observed for each feature in each local neighborhood is calculated. Then, the standard deviation is multiplied by a random factor between 4 and 8, and this amount is added to the value observed at each plant of the local neighborhood. The process continues until the desired instance noise rate is reached. Coherently with the setup defined for the anomalies with local context, we propose the same values for the instance noise rate (INR) parameter: 0.25 and 0.50, corresponding respectively to 25% and 50% of the instances.

The results in Table 8 and Table 9 show the results obtained with the (spatially-aware) baseline method and our proposed method. For our method we use the anomaly detection strategy based on k-Nearest Neighbors, which is the most robust according to the results reported in Section 4.1 (the best values for the k parameter have been selected according to the grid search executed before). The results show that the baseline method for anomaly detection exhibits the best performances in terms of F-Score in the majority of cases (10 of the 12 configurations obtained considering the 2 datasets, the 3 window size values and the 2 noise rates). Moreover, the baseline method is comparable to ours in terms of Recall (in 7 cases out of the 12 configurations our method outperforms the baseline in terms of recall).

This behavior of our approach (lower precision and comparable recall with respect to the baseline) is motivated by the fact that it is able to identify anomalies with both local and diffused contexts, whereas the baseline is able to only identify anomalies with a local context that are not anomalies in a diffused context, resulting in a much more conservative approach for this specific dataset, that contains artificially-generated contextual anomalies with diffused context.

However, from a repair viewpoint, lower anomaly detection performances do not necessarily mean worse predictions. In fact, the anomaly detection step of our proposed method identifies anomalies with local context, and repairs them exploiting normal instances observed in neighboring locations. Since in case of anomalies with diffused context most of the data associated to neighboring locations are also likely to be considered anomalous, our approach prefers to be

Pairwise comparison	p-value	winner
RMSE criterion		
No repair *vs* Repair	**1.82E-05**	Repair
Repair *vs* Repair with Feature Extraction	**1.29E-02**	Repair with Feature Extraction
No repair *vs* Repair with Feature Extraction	**8.55E-05**	Repair with Feature Extraction

Table 5: Wilcoxon Signed Rank Tests (all datasets, anomalies with local context). Bold: improvement is statistically significant when the p-value is smaller than 0.05.

conservative and not to replace values (due to absence of information).

This reflects on the experimental results obtained for the subsequent prediction step reported in Table 10 and Table 11. In fact, comparing the two methods used for the anomaly detection step, it is possible to observe that, even in those cases in which the baseline method outperforms our method in terms of F-Score, our method exhibits the best predictive performance in terms of RMSE (in 11 out of 12 configurations, either adopting the simple repair scheme or the repair scheme with feature extraction). Moreover, it is important to highlight that the baseline method cannot offer feature extraction capabilities, considering its model-less nature. Especially for the Wind NREL dataset, taking advantage of the feature extraction step leads to a consistent margin of improvement of the predictive performance. The beneficial effect of the feature extraction step is shown in Table 12, where we can see that our method with feature extraction significantly outperforms the spatially-aware baseline method. In the same table, we can see that without feature extraction the difference between our method and the baseline method is not statistically significant.

Overall, the results demonstrate the robustness for predictive tasks of our method also in case of anomalies with diffused context, although the method we propose is not specifically designed for such specific type of anomalies.

4.5. Scalability evaluation

In this section we introduce scalability results obtained with the PV Italy dataset. The original version of the dataset (∼250K instances) has been horizontally replicated by multiple factors (up to 512x), in order to test the perfor-

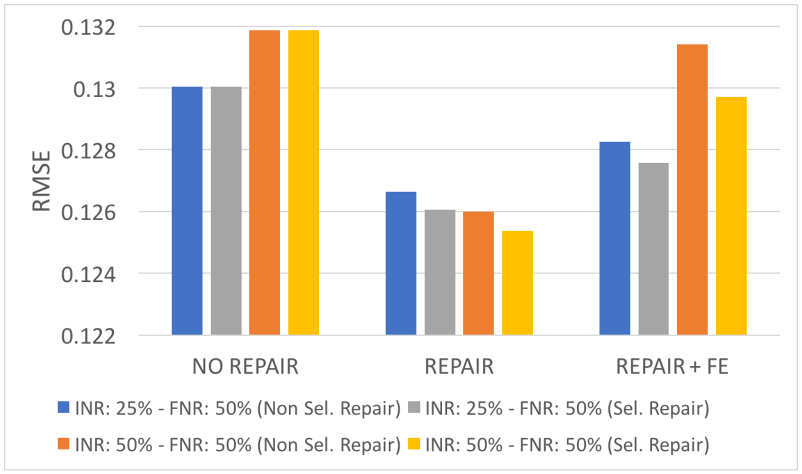

Figure 11: One day ahead forecasting error obtained without repairing noisy data (No Repair), after repair with Non-selective and Selective repair strategies (Repair), and performing repair in combination with feature extraction (Repair + FE), considering two configurations of Instance Noise Rate (INR) and Feature Noise Rate (FNR). Results are represented in terms of RMSE for the PV Italy dataset (anomalies with local context).

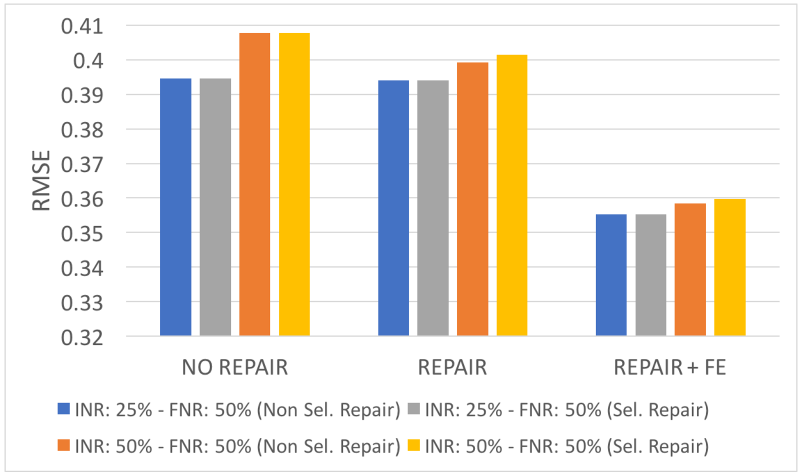

Figure 12: One day ahead forecasting error obtained without repairing noisy data (No Repair), after repair with Non-selective and Selective repair strategies (Repair), and performing repair in combination with feature extraction (Repair + FE), considering two configurations of Instance Noise Rate (INR) and Feature Noise Rate (FNR). Results are represented in terms of RMSE for the Wind NREL dataset (anomalies with local context).

Table 6: Prediction results for the PV Italy dataset (anomalies with local context) with the different strategies proposed, considering varying noise rates and training window sizes. The percentage of improvement of RMSE for the strategies involving repaired data is reported for each configuration.

Noise rate	**Window size**					
$\{INR = 0.25, FNR = 0.50\}$	30 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1330	0.0808	0.1297	0.0790	0.1287	0.0852
RMSE impr.			2.45%		3.17%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1330	0.0808	0.1290	0.0782	0.1274	0.0845
RMSE impr.			3.01%		4.15%	

Noise rate	**Window size**					
$\{INR = 0.50, FNR = 0.50\}$	30 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1349	0.0822	0.1291	0.0786	0.1339	0.0887
RMSE impr.			4.33%		0.75%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1349	0.0822	0.1282	0.0777	0.1315	0.0874
RMSE impr.			4.97%		2.50%	

Noise rate	**Window size**					
$\{INR = 0.25, FNR = 0.50\}$	60 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1295	0.0783	0.1263	0.0765	0.1269	0.0837
RMSE impr.			2.50%		1.98%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1295	0.0783	0.1256	0.0760	0.1267	0.0838
RMSE impr.			3.05%		2.14%	

Noise rate	**Window size**					
$\{INR = 0.50, FNR = 0.50\}$	60 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1315	0.0796	0.1254	0.0764	0.1293	0.0856
RMSE impr.			4.64%		1.67%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1315	0.0796	0.1251	0.0759	0.1279	0.0846
% impr.			4.87%		2.73%	

Noise rate	**Window size**					
$\{INR = 0.25, FNR = 0.50\}$	90 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1276	0.0778	0.1239	0.0757	0.1292	0.0869
RMSE impr.			2.90%		-1.30%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1276	0.0778	0.1236	0.0754	0.1286	0.0864
RMSE impr.			3.15%		-0.82%	

Noise rate	**Window size**					
$\{INR = 0.50, FNR = 0.50\}$	90 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1292	0.0790	0.1235	0.0755	0.1310	0.0880
RMSE impr.			4.44%		-1.42%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.1292	0.0790	0.1228	0.0748	0.1297	0.0873
RMSE impr.			4.98%		-0.40%	

Table 7: Prediction results for the Wind NREL dataset (anomalies with local context) with the different strategies proposed, considering varying noise rates and training window sizes. The percentage of improvement of RMSE for the strategies involving repaired data is reported for each configuration.

Noise rate	**Window size**					
$\{INR = 0.25, FNR = 0.50\}$	30 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.3945	0.3087	0.3942	0.3085	0.3481	0.2824
RMSE impr.			0.07%		11.77%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.3945	0.3087	0.3942	0.3085	0.3479	0.2823
RMSE impr.			0.06%		11.79%	
Noise rate	**Window size**					
$\{INR = 0.50, FNR = 0.50\}$	30 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.4074	0.3169	0.4001	0.3117	0.3522	0.2859
RMSE impr.			1.79%		13.56%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.4074	0.3169	0.4023	0.3127	0.3531	0.2863
RMSE impr.			1.26%		13.33%	
Noise rate	**Window size**					
$\{INR = 0.25, FNR = 0.50\}$	60 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.3945	0.3082	0.3941	0.3079	0.3514	0.2889
RMSE impr.			0.09%		10.92%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.3945	0.3082	0.3939	0.3078	0.3514	0.2890
RMSE impr.			0.13%		10.91%	
Noise rate	**Window size**					
$\{INR = 0.50, FNR = 0.50\}$	60 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.4080	0.3169	0.3992	0.3103	0.3557	0.2923
RMSE impr.			2.16%		12.81%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.4080	0.3169	0.4019	0.3125	0.3566	0.2926
% impr.			1.49%		12.59%	
Noise rate	**Window size**					
$\{INR = 0.25, FNR = 0.50\}$	90 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.3948	0.3082	0.3938	0.3076	0.3665	0.3040
RMSE impr.			0.24%		7.16%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.3948	0.3082	0.3938	0.3076	0.3665	0.3040
RMSE impr.			0.24%		7.16%	
Noise rate	**Window size**					
$\{INR = 0.50, FNR = 0.50\}$	90 days					
	No Repair		Repair		Repair + Feat.Extr.	
Non-Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.4081	0.3168	0.3986	0.3101	0.3673	0.3062
RMSE impr.			2.34%		9.99%	
Selective	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
repair	0.4081	0.3168	0.4004	0.3114	0.3696	0.3079
RMSE impr.			1.89%		9.44%	

Table 8: Anomaly detection results for the PV Italy dataset (anomalies with diffused context) with the baseline method, using different distance thresholds ($T \in 15, 30, 45$) and the proposed framework (Auto-Encoder model with 1 and 2 hidden layers), considering varying noise rates and training window sizes. Best results in terms of Recall and F-Score for each Window size configuration are marked in bold.

Noise rate	**Window size**								
$\{INR = 0.25\}$	30 days			60 days			90 days		
k-NN based	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
Proposed method (k=5)	0.8306	0.7509	0.7797	0.8411	0.7569	0.7882	0.8427	0.7618	0.7920
Baseline method	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
$T = 15$	0.6005	0.5135	0.4906	0.6005	0.5135	0.4906	0.6005	0.5135	0.4905
$T = 30$	0.9604	**0.8036**	0.8662	0.9604	**0.8036**	0.8662	0.9604	**0.8036**	0.8662
$T = 45$	0.9905	0.7965	**0.8778**	0.9905	0.7965	**0.8778**	0.9905	0.7965	**0.8778**
Noise rate	**Window size**								
$\{INR = 0.50\}$	30 days			60 days			90 days		
k-NN based	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
Proposed method (k=5)	0.8246	**0.6953**	0.7295	0.8564	**0.7163**	0.7551	0.8455	**0.7181**	0.7514
Baseline method	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
$T = 15$	0.6716	0.4994	0.5196	0.6717	0.4994	0.5196	0.6716	0.4994	0.5196
$T = 30$	0.8492	0.6932	0.7453	0.8492	0.6931	0.7453	0.8492	0.6932	0.7453
$T = 45$	0.9010	0.6760	**0.7557**	0.9010	0.6760	**0.7557**	0.9009	0.6759	**0.7557**

Table 9: Anomaly detection results for the Wind NREL dataset (anomalies with diffused context) with the baseline method, using different distance thresholds ($T \in 15, 30, 45$) and the proposed framework (Auto-Encoder model with 1 and 2 hidden layers), considering varying noise rates and training window sizes. Best results in terms of Recall and F-Score for each Window size configuration are marked in bold.

Noise rate	**Window size**								
$\{INR = 0.25\}$	30 days			60 days			90 days		
k-NN based	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
Proposed method (k=200)	0.9992	0.7508	0.8569	0.9925	**0.7525**	0.8539	0.9913	**0.7567**	0.8543
Baseline method	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
$T = 15$	0.9980	**0.7520**	0.8566	0.9980	0.7520	0.8566	0.9980	0.7520	0.8566
$T = 30$	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**
$T = 45$	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**
Noise rate	**Window size**								
$\{INR = 0.50\}$	30 days			60 days			90 days		
k-NN variant	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
Proposed method (k=200)	0.9961	0.5192	0.6788	0.9907	**0.5250**	**0.6783**	0.9804	**0.5400**	**0.6811**
Baseline method	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
$T = 15$	0.9864	**0.5211**	0.6755	0.9864	0.5211	0.6755	0.9864	0.5211	0.6755
$T = 30$	0.9903	0.5167	0.6757	0.9903	0.5167	0.6757	0.9903	0.5167	0.6757
$T = 45$	0.9977	0.5174	**0.6790**	0.9977	0.5174	0.6790	0.9977	0.5174	0.6790

Table 10: Prediction results for the PV Italy dataset (anomalies with diffused context) with the baseline method, using different distance thresholds ($T \in 15, 30, 45$) and the proposed framework (Auto-Encoder model with 1 and 2 hidden layers), considering varying noise rates and training window sizes. All the configurations use the best performing repair variant (selective repair).

Noise rate {$INR = 0.25$}	**Window size** 30 days					
	No Repair		Repair		Repair + Feat.Extr.	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
Baseline method (*T*=15)	0.1366	0.0859	0.1429	0.0926	NA	NA
Baseline method (*T*=30)	0.1366	0.0859	0.1368	0.0856	NA	NA
Baseline method (*T*=45)	0.1366	0.0859	0.1365	0.0856	NA	NA
Proposed method ($k = 5$)	0.1366	0.0859	**0.1348**	0.0841	0.1446	0.0945

Noise rate {$INR = 0.50$}	**Window size** 30 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline T=15	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
Baseline method (*T*=15)	0.1371	0.0865	0.1410	0.0909	NA	NA
Baseline method (*T*=30)	0.1371	0.0865	0.1374	0.0862	NA	NA
Baseline method (*T*=45)	0.1371	0.0865	0.1371	0.0865	NA	NA
Proposed method ($k = 5$)	0.1371	0.0865	**0.1350**	0.0842	0.1463	0.0952

Noise rate {$INR = 0.25$}	**Window size** 60 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline method (*T*=15)	0.1376	0.0845	0.1455	0.0920	NA	NA
Baseline method (*T*=30)	0.1376	0.0845	0.1377	0.0847	NA	NA
Baseline method (*T*=45)	0.1376	0.0845	**0.1372**	0.0842	NA	NA
Proposed method ($k = 5$)	0.1376	0.0845	0.1376	0.0841	0.1377	0.0937

Noise rate {$INR = 0.50$}	**Window size** 60 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline method (*T*=15)	0.1379	0.0851	0.1431	0.0899	NA	NA
Baseline method (*T*=30)	0.1379	0.0851	0.1387	0.0856	NA	NA
Baseline method (*T*=45)	0.1379	0.0851	0.1383	0.0850	NA	NA
Proposed method ($k = 5$)	0.1379	0.0851	**0.1368**	0.0841	0.1396	0.0943

Noise rate {$INR = 0.25$}	**Window size** 90 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline method (*T*=15)	0.1355	0.0847	0.1424	0.0923	NA	NA
Baseline method (*T*=30)	0.1355	0.0847	0.1355	0.0848	NA	NA
Baseline method (*T*=45)	0.1355	0.0847	0.1349	0.0843	NA	NA
Proposed method ($k = 5$)	0.1355	0.0847	0.1329	0.0830	**0.1321**	0.0882

Noise rate {$INR = 0.50$}	**Window size** 90 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline T=15	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
Baseline method (*T*=15)	0.1360	0.0850	0.1404	0.0901	NA	NA
Baseline method (*T*=30)	0.1360	0.0850	0.1366	0.0852	NA	NA
Baseline method (*T*=45)	0.1360	0.0850	0.1360	0.0850	NA	NA
Proposed method ($k = 5$)	0.1360	0.0850	0.1337	0.0837	**0.1331**	0.0889

Table 11: Prediction results for the Wind NREL dataset (anomalies with diffused context) with the baseline method, using different distance thresholds ($T \in 15, 30, 45$) and the proposed framework (Auto-Encoder model with 1 and 2 hidden layers), considering varying noise rates and training window sizes. All the configurations use the best performing repair variant (selective repair).

Noise rate $\{INR = 0.25\}$	**Window size** 30 days					
	No Repair		Repair		Repair + Feat.Extr.	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
Baseline method (T=15)	0.4433	0.3430	0.4431	0.3427	NA	NA
Baseline method (T=30)	0.4433	0.3430	0.4433	0.3430	NA	NA
Baseline method (T=45)	0.4433	0.3430	0.4433	0.3430	NA	NA
Proposed method (k = 200)	0.4433	0.3430	0.4430	0.3428	**0.3057**	0.2433

Noise rate $\{INR = 0.50\}$	**Window size** 30 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline T=15	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
Baseline method (T=15)	0.4652	0.3626	0.4649	0.3625	NA	NA
Baseline method (T=30)	0.4652	0.3626	0.4651	0.3624	NA	NA
Baseline method (T=45)	0.4652	0.3626	0.4651	0.3625	NA	NA
Proposed method (k = 200)	0.4652	0.3626	0.4654	0.3630	**0.3227**	0.2547

Noise rate $\{INR = 0.25\}$	**Window size** 60 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline method (T=15)	0.4371	0.3371	0.4369	0.3368	NA	NA
Baseline method (T=30)	0.4371	0.3371	0.4371	0.3371	NA	NA
Baseline method (T=45)	0.4371	0.3371	0.4371	0.3371	NA	NA
Proposed method (1 - k = 200)	0.4371	0.3371	0.4361	0.3359	0.3522	0.2797
Proposed method (k = 200)	0.4371	0.3371	0.4359	0.3362	**0.3214**	0.2553

Noise rate $\{INR = 0.50\}$	**Window size** 60 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline method (T=15)	0.4589	0.3550	0.4581	0.3543	NA	NA
Baseline method (T=30)	0.4589	0.3550	0.4584	0.3543	NA	NA
Baseline method (T=45)	0.4589	0.3550	0.4584	0.3544	NA	NA
Proposed method (k = 200)	0.4589	0.3550	0.4589	0.3549	**0.3321**	0.2651

Noise rate $\{INR = 0.25\}$	**Window size** 90 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline method (T=15)	0.4383	0.3389	0.4382	0.3386	NA	NA
Baseline method (T=30)	0.4383	0.3389	0.4383	0.3389	NA	NA
Baseline method (T=45)	0.4383	0.3389	0.4383	0.3389	NA	NA
Proposed method (k = 200)	0.4393	0.3396	0.4390	0.3391	**0.3179**	0.2507

Noise rate $\{INR = 0.50\}$	**Window size** 90 days					
	No Repair		Repair		Repair + Feat.Extr.	
Baseline T=15	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
Baseline method (T=15)	0.4563	0.3517	0.4549	0.3507	NA	NA
Baseline method (T=30)	0.4563	0.3517	0.4552	0.3507	NA	NA
Baseline method (T=45)	0.4563	0.3517	0.4553	0.3508	NA	NA
Proposed method (k = 200)	0.4571	0.3526	0.4571	0.3524	**0.3293**	0.2594

Pairwise comparison	p-value	winner
RMSE criterion		
Repair (Baseline) *vs* Repair (Proposed)	0.6949	Repair baseline
Repair (Baseline) *vs* Repair + Feat.Extr. (Proposed)	**0.0499**	Repair + Feat.Extr. (Proposed)

Table 12: Wilcoxon Signed Rank Tests (all datasets, anomalies with diffused context). The best performing configuration is selected for the Baseline method ($T = 45$). Bold: improvement is statistically significant when the p-value is smaller than 0.05.

mances of our method with stress tests and speedup/scaleup tests.

All the experiments have been conducted on a Azure HDInsight Spark cluster consisting of one driver node (6 cores, 32GB RAM) and 6 worker nodes (8 cores each, 192GB RAM in total) and SSD hard drives.

The running times, with the different samples, are shown in Fig. 13. The results highlight the linear complexity of the anomaly detection algorithm considering enough neighbors for each object in the embedding ($k = 25$). This is a clear indication that the algorithm is not affected by computational bottlenecks such as complex operations performed on the driver node.

The running times, when compared with the non-distributed approach, show a significant reduction margin, which becomes increasingly higher as the number of instances to be processed increases. This aspect becomes clear when we analyze speedup and scaleup results.

Figure 14 (left), reports the speedup factor obtained when processing up to 32M instances with an increasing number of cores. Moreover, Figure 14 (right) shows the scaleup performances obtained with an increasing number of data instances and cores (2M - 8 cores, 4M - 16 cores, 6M - 24 cores, 8M - 32 cores, 10M - 40 cores, 12M - 48 cores). Both, speedup and scaleup curves are quite close to the ideal curves (linear and constant curves, respectively). This confirms that our anomaly detection method can be profitably used in a cluster environment with large datasets.

Finally, Figure 15 reports the execution time for the k-NN based anomaly detection method ($k = 25$) observed with different levels of data parallelism, that is, different number of partitions for the Spark dataset. The results show that the setting with 100 partitions is the best performing for our cluster configuration.

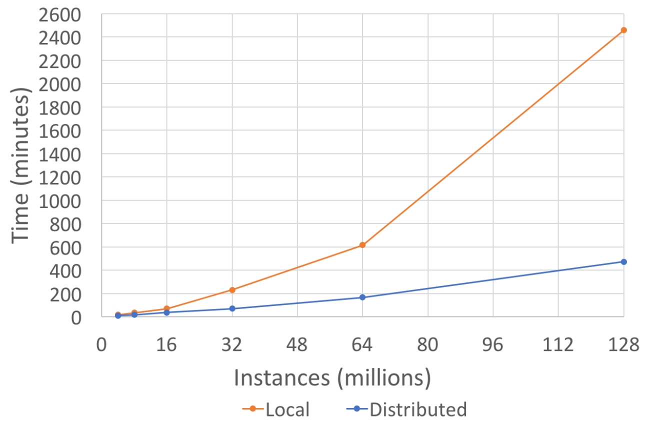

Figure 13: Stress test results (PV Italy dataset) for the k-NN based anomaly detection method ($k = 25$).

4.6. Availability

The system and the datasets are available to replicate the experiments at the following URL: `http://www.di.uniba.it/~ceci/ad-repair-framework`.

5. Conclusions

In this paper we presented a framework which supports predictive modeling tasks involving streaming data coming from multiple geo-referenced sensors. We have formulated a novel anomaly detection strategy which trains a stacked auto-encoder using non-anomalous data and then compares the projections of testing data (encoded data) in the auto-encoder embedding with their closest projections. This strategy is capable to detect which instances are anomalies, adopting a distance-based approach and a dynamically learned threshold. Moreover, we proposed two repair schemes which, as new data arrive, automatically and dynamically repair the instances identified as anomalies, adopting a spatial weighting which simultaneously considers spatial autocorrelation from nearby

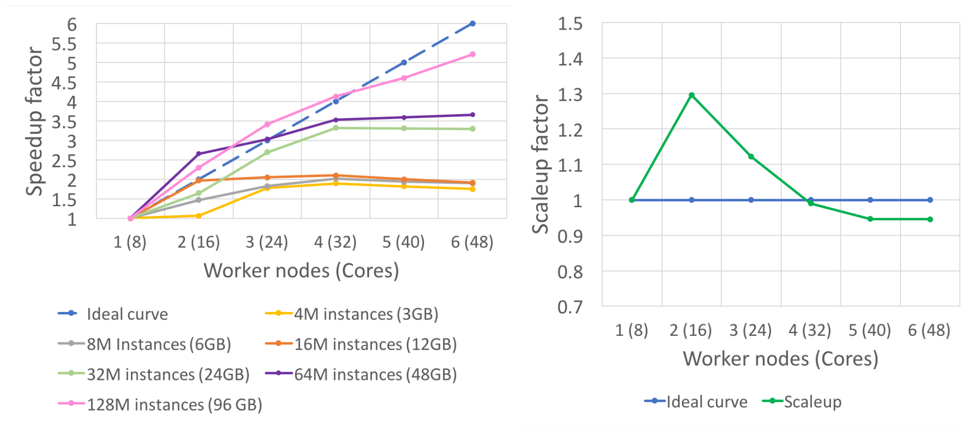

Figure 14: Speedup and scaleup results (PV Italy dataset) for the k-NN based anomaly detection method ($k = 25$).

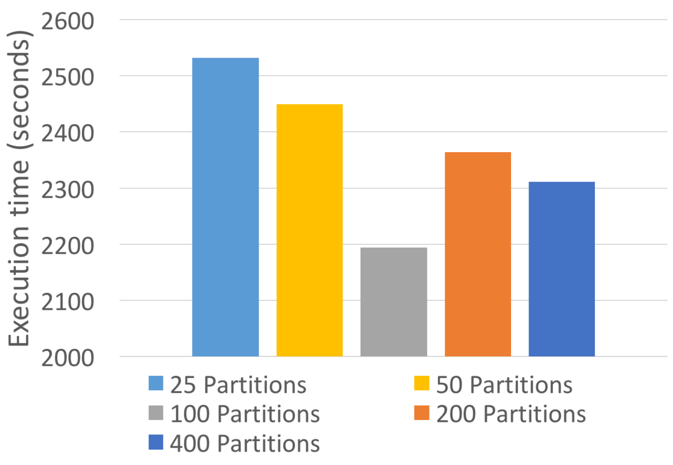

Figure 15: Execution time for the k-NN based anomaly detection method ($k = 25$) with different number of partitions for the Spark dataset. Results are obtained in cluster mode with six worker nodes, using the 16M instances version of the dataset.

locations. In addition, we carried out the predictive task using Gradient Boosted Trees with two different feature representations of the input data: the original data representation, after the data repair procedure, and an embedding feature space extracted from the auto-encoder embedding.

Experiments have been performed for the one-day-ahead predictive modeling of energy production for multiple renewable energy sites, carried out on two datasets. The results have shown that adopting the framework proposed, allows us to obtain up to 13.56% of RMSE reduction, compared to the baseline scenario which directly applies the predictive model discarding the anomaly detection and repair steps. This demonstrates that the combination of stacked auto-encoder and Gradient Boosted Trees models, complemented with effective anomaly detection, data repair and feature extraction strategies, can be of valuable support for predictive modeling tasks in geo-distributed sensor networks.

As future work we aim to investigate the adoption of statistical indicators in the learning process, in order to explicitly consider the spatial autocorrelation phenomenon in the models. Moreover, we aim to extend the learning setting to multi-target regression to predict time series.

Acknowledgement

The research described in this paper has been funded by the Ministry of Education, Universities and Research (MIUR) through the project "ComESto - Community Energy Storage: Gestione Aggregata di Sistemi d'Accumulo dell'Energia in Power Cloud" (Grant N. ARS01_01259). We also acknowledge the support of the European commission through the projects "MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data" (Grant ICT-2013-612944) and "TOREADOR - TrustwOrthy model-awaRE Analytics Data platform" (Grant 988797). The computational work has been carried out on the resources provided by the projects Microsoft Azure for Research, ReCaS (PONa3_00052), PRISMA (PON04a2_A). The authors also wish to thank Lynn Rudd for her help in reading the manuscript.

References

[1] X. Zhu, Semi-supervised learning literature survey, Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison (2005).
URL `http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`

[2] H. Kriegel, P. Kröger, A. Zimek, Outlier detection techniques, Tutorial at KDD 10.

[3] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 15:1–15:58. `doi:10.1145/1541880.1541882`.
URL `http://doi.acm.org/10.1145/1541880.1541882`

[4] Y. Y. Bengio, et al., Learning deep architectures for ai, Foundations and trends® in Machine Learning 2 (1) (2009) 1–127.

[5] M. Najafabadi, F. Villanustre, T. Khoshgoftaar, N. Seliya, R. Wald, E. Muharemagic, Deep learning applications and challenges in big data analytics, Journal of Big Data 2 (1) (2015) 1.

[6] G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, science 313 (5786) (2006) 504–507.

[7] N. Japkowicz, Supervised versus unsupervised binary-learning by feedforward neural networks, Machine Learning 42 (1/2) (2001) 97–122. `doi:10.1023/A:1007660820062`.

[8] M. Sakurada, T. Yairi, Anomaly detection using autoencoders with nonlinear dimensionality reduction, in: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, ACM, 2014, p. 4.

[9] C. Zhou, R. Paffenroth, Anomaly detection with robust deep autoencoders, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 665–674.

[10] N. Japkowicz, S. J. Hanson, M. A. Gluck, Nonlinear autoassociation is not equivalent to PCA, Neural Computation 12 (3) (2000) 531–545. `doi:10.1162/089976600300015691`.

[11] J. Gehring, Y. Miao, F. Metze, A. Waibel, Extracting deep bottleneck features using stacked auto-encoders, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2013, pp. 3377–3381.

[12] J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, Artificial Neural Networks and Machine Learning–ICANN 2011 (2011) 52–59.

[13] S. Sakr, A. Liu, D. M. Batista, M. Alomari, A survey of large scale data management approaches in cloud environments, IEEE Communications Surveys & Tutorials 13 (3) (2011) 311–336.

[14] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM computing surveys (CSUR) 41 (3) (2009) 15.

[15] A. S. Weigend, M. Mangeas, A. N. Srivastava, Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting, International Journal of Neural Systems 6 (04) (1995) 373–399.

[16] Y. Kou, C.-T. Lu, D. Chen, Spatial weighted outlier detection, in: Proceedings of the 2006 SIAM international conference on data mining, SIAM, 2006, pp. 614–618.

[17] S. Shekhar, C.-T. Lu, P. Zhang, Detecting graph-based spatial outliers: algorithms and applications (a summary of results), in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 371–376.

[18] Y. S. Chong, Y. H. Tay, Abnormal event detection in videos using spatiotemporal autoencoder, in: International Symposium on Neural Networks, Springer, 2017, pp. 189–196.

[19] E. Keogh, S. Chu, D. Hart, M. Pazzani, An online algorithm for segmenting time series, in: IEEE International Conference on Data Mining, IEEE, 2001, pp. 289–296.

[20] Z. Zhao, W. Ng, A model-based approach for rfid data stream cleansing, in: Proceedings of the 21st ACM international conference on Information and knowledge management, ACM, 2012, pp. 862–871.

[21] J. Zhang, S. Zhang, J. Liang, B. Tian, Z. Hou, B. Liu, Photovoltaic generation data cleaning method based on approximately periodic time series, in: IOP Conference Series: Earth and Environmental Science, Vol. 63:1, IOP Publishing, 2017, p. 012008.

[22] P. Bohannon, W. Fan, M. Flaster, R. Rastogi, A cost-based model and effective heuristic for repairing constraints by value modification, in: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, 2005, pp. 143–154.

[23] X. Chu, I. Ilyas, P. Papotti, Holistic data cleaning: Putting violations into context, in: IEEE 29th International Conference on Data Engineering (ICDE), IEEE, 2013, pp. 458–469.

[24] Y. Tian, P. Michiardi, M. Vukoli, Bleach: A distributed stream data cleaning system, in: Big Data (BigData Congress), 2017 IEEE International Congress on, IEEE, 2017, pp. 113–120.

[25] S. Song, A. Zhang, J. Wang, P. Yu, Screen: Stream data cleaning under speed constraints, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, 2015, pp. 827–841.

[26] H. J. Miller, Tobler's first law and spatial analysis, Annals of the Association of American Geographers 94 (2) (2004) 284–289.

[27] M. M. Sohrabi, R. Benjankar, D. Tonina, S. J. Wenger, D. J. Isaak, Estimation of daily stream water temperatures with a bayesian regression approach, Hydrological processes 31 (9) (2017) 1719–1733.

[28] W. Pei, H. Dibeklioğlu, D. M. Tax, L. van der Maaten, Multivariate time-series classification using the hidden-unit logistic model, IEEE transactions on neural networks and learning systems 29 (4) (2018) 920–931.

[29] F. Karim, S. Majumdar, H. Darabi, S. Chen, Lstm fully convolutional networks for time series classification, IEEE Access 6 (2018) 1662–1669.

[30] A. Rashkovska, J. Novljan, M. Smolnikar, M. Mohorčič, C. Fortuna, Online short-term forecasting of photovoltaic energy production, in: Innovative Smart Grid Technologies Conference (ISGT), 2015 IEEE Power & Energy Society, 2015, pp. 1–5.

[31] C. Persson, P. Bacher, T. Shiga, H. Madsen, Multi-site solar power forecasting using gradient boosted regression trees, Solar Energy 150 (2017) 423–436.

[32] R. J. Bessa, A. Trindade, V. Miranda, Spatial-temporal solar power forecasting for smart grids, IEEE Transactions on Industrial Informatics 11 (1) (2015) 232–241.

[33] M. Ceci, R. Corizzo, F. Fumarola, D. Malerba, A. Rashkovska, Predictive modeling of PV energy production: How to set up the learning task for a better prediction?, IEEE Trans. Industrial Informatics 13 (3) (2017) 956–966. `doi:10.1109/TII.2016.2604758`.

[34] B. Liu, J. Nowotarski, T. Hong, R. Weron, Probabilistic load forecasting via quantile regression averaging on sister forecasts, IEEE Transactions on Smart Grid 8 (2) (2017) 730–737.

[35] S. Fang, H. Chiang, A high-accuracy wind power forecasting model, IEEE Transactions on Power Systems 32 (2) (2017) 1589–1590.

[36] J. Dowell, P. Pinson, Very-short-term probabilistic wind power forecasts by sparse vector autoregression, IEEE Transactions on Smart Grid 7 (2) (2016) 763–770.

[37] J. Huang, M. Perry, A semi-empirical approach using gradient boosting and k-nearest neighbors regression for gefcom2014 probabilistic solar power forecasting, International Journal of Forecasting 32 (3) (2016) 1081–1086.

[38] A. Appice, A. Ciampi, F. Fumarola, D. Malerba, Data Mining Techniques in Sensor Networks - Summarization, Interpolation and Surveillance, Springer Briefs in Computer Science, Springer, 2014. `doi:10.1007/978-1-4471-5454-9`.
URL `https://doi.org/10.1007/978-1-4471-5454-9`

[39] F. Pukelsheim, The three sigma rule, The American Statistician 48 (2) (1994) 88–91.

[40] T. Liu, A. W. Moore, A. Gray, K. Yang, An investigation of practical approximate nearest neighbor algorithms, in: Proceedings of the 17th International Conference on Neural Information Processing Systems, NIPS'04, MIT Press, Cambridge, MA, USA, 2004, pp. 825–832.
URL `http://dl.acm.org/citation.cfm?id=2976040.2976144`

[41] D. Stojanova, M. Ceci, A. Appice, D. Malerba, S. Dzeroski, Dealing with spatial autocorrelation when learning predictive clustering trees, Ecological Informatics 13 (2013) 22–39. `doi:10.1016/j.ecoinf.2012.10.006`.
URL `https://doi.org/10.1016/j.ecoinf.2012.10.006`

[42] P. A. Moran, Notes on continuous stochastic phenomena, Biometrika 37 (1/2) (1950) 17–23.

[43] C. H. Mason, W. D. Perreault Jr, Collinearity, power, and interpretation of multiple regression analysis, Journal of marketing research (1991) 268–280.

[44] D. A. Belsley, E. Kuh, R. E. Welsch, Regression diagnostics: Identifying influential data and sources of collinearity (2005).

[45] M. Cavallo, G. Di Modica, C. Polito, O. Tomarchio, Fragmenting big data to boost the performance of mapreduce in geographical computing con-

texts, in: Big Data Innovations and Applications (Innovate-Data), 2017 International Conference on, IEEE, 2017, pp. 17–24.

[46] M. Cavallo, C. Polito, G. D. Modica, O. Tomarchio, H2f: a hierarchical hadoop framework for big data processing in geo-distributed environments, in: Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, ACM, 2016, pp. 27–35.