CHALMERS
UNIVERSITY OF TECHNOLOGY

Survey: Time-Series Data Preprocessing: A Survey and an Empirical Analysis

Journal of Engineering Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of Engineering Research

journal homepage: www.journals.elsevier.com/journal-of-engineering-research

Survey:Time-series data preprocessing: A survey and an empirical analysis

Amal Tawakuli [a,*], Bastian Havers [b], Vincenzo Gulisano [c], Daniel Kaiser [a], Thomas Engel [a]

[a] University of Luxembourg, Esch-sur-Alzette, Luxembourg
[b] Volvo Car Corporation, Gothenburg, Sweden
[c] Chalmers University of Technology, Gothenburg, Sweden

ARTICLE INFO

Keywords:
Data preprocessing
Data quality

ABSTRACT

Data are naturally collected in their raw state and must undergo a series of preprocessing steps to obtain data in their input state for Artificial Intelligence (AI) and other applications. The data preprocessing phase is not only necessary to fit input requirements but also effective in improving AI training efficiency and output accuracy. Data preprocessing is a time consuming and complex phase that lacks a unified and structured approach. We survey data preprocessing techniques under different categories to provide an extended and structured scope of data preprocessing relevant to numerical time-series data. We also provide an empirical analysis of the impact of preprocessing techniques on the quality of the data and on the performance of AI algorithms. In addition, we discuss the feasibility of distributing some of the surveyed techniques to the edge. Leveraging edge computing to distribute data preprocessing reduces the workload on central systems, creates more manageable data lakes, reduces the consumption of resources (e.g., energy) and enables EdgeAI.

Introduction

One can identify from literature a common understanding that data preprocessing is the set of operations that transform raw data into quality input data. It includes operations such as dimensionality reduction, normalization and outlier detection [3,59,92,167]. Preprocessing data is necessary to obtain quality input and ultimately quality output, particularly for AI models and networks. To list a few challenges, raw data may be incompatible (in terms of size, format, etc.), biased, or may consist of outliers and redundancies. Even though they are not considered as quality issues, the heterogeneity and high dimensionality of data may impose further challenges. To perform analysis on data, it must first undergo several preprocessing operations. The required preprocessing tasks, their sequence, optimal location of execution, and parameters depend on many factors, including the type of data, the data source, system context, application, available resources, and the type of algorithm consuming the data. There are also "individual factors", such as the user's experience in preparing data [107]. Data preprocessing is also typically tailored for a specific problem or application [54,110,139] and is initiated after the accumulation of raw data (batched), thus lacking standardization and being prone to errors and delays. This approach could result in repeating common and shared preprocessing tasks on the same data but for different applications. Accumulating raw data in a central system collectively wastes bandwidth (used to transfer raw data from sensors), storage, and other resources (time, energy, etc.), as raw data often include anomalies that fruitlessly consume these valuable resources. This may be tolerated to some extent, however, with the wide spread of IoT applications and devices, the accumulation of data collected from the edge and transferred to central systems have considerably higher impact on available resources and infrastructure. The automotive sector is a good example, where there is an imbalance between the volume of data sensed from vehicles and infrastructure's capacities, in addition to the inherent limitations of batch technology [81]. Batch preprocessing raw data in a central system may take up to 80% of available resources [92,110]. While preprocessing data streams (i.e., as data is collected) close to the data source could improve efficiency and standardize shared preprocessing requirements across different applications. Edge preprocessing is also an intuitive approach to address private or biased data to fulfill regulations concerning privacy protection (e.g., GDPR) and fairness.

This survey is a broad account of different types of preprocessing based on a holistic notion and an extended scope of data preprocessing. It is not limited to addressing data cleaning but also covers techniques that address other aspects such as sensor fusion and data compression. Fig. 4 introduces our holistic and standardized taxonomy of data

* Corresponding author.
E-mail address: amal.tawakuli@uni.lu (A. Tawakuli).

https://doi.org/10.1016/j.jer.2024.02.018
Received 19 November 2023; Received in revised form 18 February 2024; Accepted 26 February 2024
Available online 8 March 2024

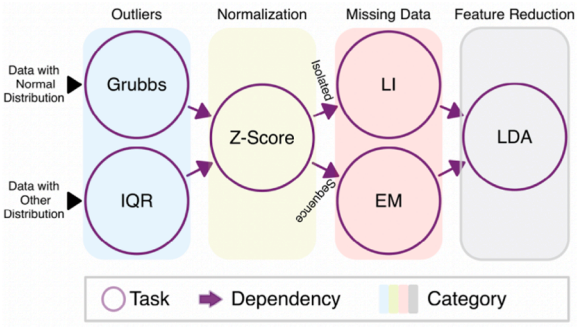


Fig. 1. The Preprocessing DAG.

preprocessing. Also worth highlighting about the scope of the survey is that it focuses on techniques applicable to numerical time-series data. Our objective is to provide a practical data preprocessing guide for practitioners and academics alike. It is also the first survey to include experimental analyses of the impact of the tested preprocessing techniques on both the quality of the data (model input) and the quality and accuracy of the extracted information (model output). For the empirical analysis, we used a real-world dataset and a use-case with controlled experimental setup and configuration. We also briefly discuss the feasibility of distributing some of the techniques to the edge for the aforementioned benefits.

The survey is organized as follows: In Section 2, we define quality data and its characteristics. We describe our experimental setup and the selected real-world dataset in addition to our evaluation methods and metrics in Section 3. In section 4, we propose a broader definition of data preprocessing and present data preprocessing techniques under different identified categories. We also highlight dependencies between different data preprocessing categories in Section 5. Lastly, we conclude and present future research opportunities in Section 6.

Characteristics of data quality

Some of the characteristics of quality data mentioned in relevant literature include: clean, compatible, accurate, reliable, interpretable, complete, trustworthy, unbiased, secure, useful, valuable, easy to access, traceable, and timely [7,78,100,188]. Data with such characteristics contribute to the application's performance and integrity and prevent errors from propagating through the dataflow pipeline, causing losses and delays. From a technical perspective, quality data contribute to better predictions and classifications and faster convergence (more efficient learning). From a business perspective, quality data result in more reliable decision-making, achieve legal compliance, and boost operations' efficiency [141]. To obtain the aforementioned characteristics, factors that hinder data quality must be addressed. These factors and challenges include outliers, missing instances, high dimensional data, variant scales, biased data, and sensitive or private data. Many techniques have been proposed by the research community to address each factor, with some challenges attracting more attention, such as missing data. This survey is comprehensive in the sense that it presents techniques for addressing different data anomalies and challenges. Thus, it is not limited to data cleaning but rather broadens the meaning of data preprocessing to cover categories that achieve wider characteristics of quality in data by addressing extended challenges found in the data. Each category is self-contained, with many categories having an extensive research history and a wide range of proposed solutions by active research communities.

Dataset and experimental details

The preprocessing techniques are evaluated based on their impact on the data and their impact on the prediction model. Our selection criteria are based on choosing techniques of diverse types (univariate, multivariate, statistical ML, etc.). We used the same dataset, applied the same overall data preprocessing pipeline, and used the same evaluation strategy across all experiments within the categories covered in the survey. This was important to ensure consistency and control variations in the experimental environment. The evaluation strategy also involved training Long Short Term Memory (LSTM) networks using the dataset produced by each preprocessing experiment to evaluate the impact of the preprocessing technique on prediction quality and not just the data quality and characteristics. To control experimental variation, the same training configuration and evaluation metrics were used in all of our experiments. The following are the details of each of these empirical components:

The dataset

We used the real-world time-series **AirQuality**[193] dataset from the UCI Machine Learning repository. The dataset includes 9358 instances (1.3 MB) of hourly averaged values of 15 variables, including temperature, humidity, and air pollutant chemicals, that were collected from sensors located in a polluted area in an Italian city between March 2004 and February 2005. We selected the dataset because it consists of continuous sensor data and can be used to evaluate a wide range of regression preprocessing techniques, which is our focus, and classification preprocessing techniques. The only limitation of the dataset for the purpose of this survey is that it cannot be used to evaluate privacy preservation techniques and debiasing techniques as it does not include protected or sensitive features. The dataset is a good example of scenarios where deploying preprocessing to the edge is feasible and intuitive. The dataset was divided so that 90% of the data is used for training while the rest is used for testing. From the 15 features in the dataset, we allocated the feature Carbon Monoxide (*CO*) as the response variable and the rest of the features as predictors. Models or networks trained would attempt to predict the values of *CO* from the predictors, which are the input used for training and predicting. As a future work, we wish to perform similar tests on several other datasets from different industries to confirm and generalize our conclusions.

Standardized preprocessing of the dataset across all experiments

To control variations in our experiments and for comparable results, we standardized the required preprocessing tasks across all experiments. As a default and for any dataset, the presence of anomalies, such as missing data and outliers, must be checked and handled. To avoid overfitting, we also normalized the scales of the different features and performed feature selection to select the most informative features. We tested multiple techniques for each category on the selected dataset and chose the techniques that yielded the best results. The preprocessing pipeline starts with replacing the reserved value ($-$ 200) for missing data in the dataset with *NaN*. This is followed by detecting outliers in each feature using the Grubbs outlier detection technique for normally distributed features and the Interquartile Range outlier detection technique otherwise. The outliers are replaced with estimated values using the imputation technique Cubic Spline Interpolation. The next anomaly addressed is the missing data anomaly. As proposed in [186], we differentiated between isolated and sequence missing instances and used an optimal technique for each. For isolated missing instances, we used Cubic Spline Interpolation to estimate the missing instances. For sequence missing data, we used Expectation Maximization (EM) to impute the missing data. For our dataset, the different attributes (features) have already been fused into one dataset; however, in a real-world IoT scenario, a sensor fusion technique must be applied prior to EM as it is the first multivariate technique in the preprocessing plan. Finally, we applied multiple feature selection techniques, including Neighborhood Component Analysis and Laplacian Scores, and selected the top most common features. The Directed Acyclic Diagram (DAG) in Fig. 1 (see

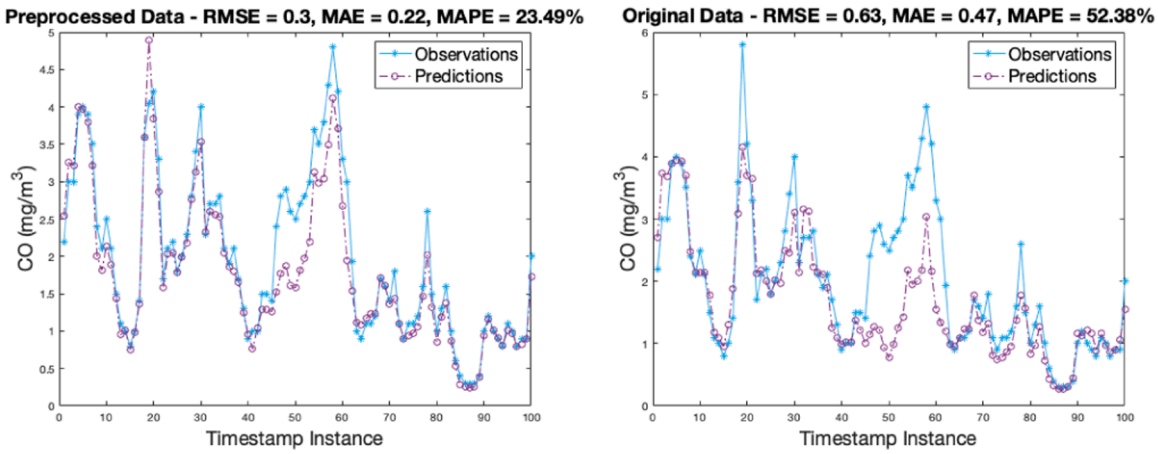


Fig. 2. LSTM Performance Trained on Preprocessed Data (Left) and Trained on the Original Data.

appendix) represents the preprocessing plan used across the different experiments of the survey. At each category represented in the DAG (e.g., Missing Data Imputation), the default technique for all the experiments is replaced with the tested techniques for evaluation and comparison, while the rest of the preprocessing categories of the DAG remain unchanged (with the same default technique). Categories not represented in the DAG (e.g., compression) are applied after the last node, after Feature Selection, as further preprocessing.

The preprocessing pipeline presented in Fig. 1 is specific to this particular dataset and to our experiments. It was derived based on our experimental evaluations of different preprocessing techniques, including their applicability, and yielded results. For example, data debiasing and privacy preserving techniques were not applied as the dataset does not contain attributes that are explicitly or implicitly related to individuals. Fig. 1 does not represent the scope of data preprocessing as presented in this survey, nor does it imply the order or depth of our survey and experimentations. We have covered more preprocessing categories and techniques for both the survey and experimental analysis. The preprocessing pipeline presented in Fig. 1 is merely for controlling and standardizing variations in our experiments.

Empirical evaluation methods and metrics

Evaluating the effectiveness of a preprocessing technique in addressing an anomaly in the data and its impact on the quality of the data (Input Quality) is not sufficient as the only evaluation component. For example, detecting potential outliers in a dataset using an outlier detection technique based on local or global value ranges that highlight extreme values at both ends is only one part of determining the performance of the technique or the quality of the produced data. An extreme value may not necessarily be an outlier but rather provide important information about the real-world. The bottom line is that we want quality data because we want accurate predictions and valuable information obtained from the data. For such reasons and to help us assess the amount of information loss or accuracy gains that resulted from applying a preprocessing technique, we also evaluate the technique based on its impact on the prediction accuracy (Output Quality). For such purpose, we trained LSTM networks [87] that predict the values of Carbon Monoxide *CO*. LSTM is a recurrent neural network (RNN) that models sequential or time-series data with memory capacity to capture long term dependencies. LSTM was proposed to address the diminishing gradients issue suffered by RNN [87,176,178]. LSTM networks are proven powerful solutions for speech recognition, handwriting recognition, and making predictions based on time-series data [24,31,62,70,133,202,208]. In addition, LSTM can be used for non-linear, non-monotonic, and fluctuating data where historic information maintains its value [87]; making the LSTM algorithm a plausible choice for our experiments.

We used MATLAB R2020b to conduct our experiments on two local machines. The LSTM network architecture consists of a sequence input layer, an LSTM layer with 200 hidden units, a fully connected layer, and a regression output layer. We trained the network using the Adaptive Moment Estimation (Adam) optimization algorithm with an initial learning rate of 0.005 and a maximum of 100 full passes through the entire dataset.

We evaluated trained LSTM networks using three error metrics, namely Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE), which are commonly used by the research community [75,105,127,146,188]. We denormalized the predictions and observations of the response variable to present the results on the original *CO* scale and thus obtain comparable results. We repeated each experiment at least three times. The averaged error metric results (the average result of the last three tests) for each experiment are presented in the tables that summarize the tested techniques. We also present the results visually using suitable plots for the different categories, which are included in the appendix.

Figure 2 contains two plots representing the results of two experiments. The one on the left plots the predictions of an LSTM network trained on the AirQuality dataset preprocessed according to the DAG shown in Fig. 1 and the one on the right plots the predictions of an LSTM network trained on data with minimum preprocessing to remove the NaN values using Cubic Spline Interpolation. The later experiment required the removal of the NaN values to avoid errors in training; otherwise, the data are kept in their original state. The values of the error metrics on the plots are from the last test conducted. The average errors of the last three tests for the experiment where the complete DAG is applied are: 0.32 for the RMSE metric, 0.23 for the MAE metric, and 25.26% for the MAPE metric. While the average errors of the last three tests for the experiment where the DAG was not applied are: 0.60 for the RMSE metric, 0.45 for the MAE metric, and 51.41% for the MAPE metric. The prediction accuracy difference between the two LSTM networks is significant, with the network trained on the DAG preprocessed data yielding error reductions of $\sim$ 46.66% in the RMSE value, $\sim$ 48.88% in the MAE value, and of $\sim$ 50.87% in the MAPE value. The observations (blue lines in 1) between the two experiments vary slightly because the data in the experiment represented by the left plot underwent several preprocessing steps (see 1) that included replacing outliers.

Data preprocessing

Our objective is to cover as many techniques as possible under each identified category and sub-category. Given the time constraint,

Table 1
Sample of Preprocessing Techniques Covered in The Survey.

#	Technique	Category	Input Type	Tested
1	Min-Max Scaling	Normalization	Univariate	Yes
2	Z-score	Normalization	Univariate	Yes
3	Robust Standardization	Normalization	Univariate	Yes
4	P-Norm	Normalization	Univariate	Yes
5	Decimal Scaling	Normalization	Univariate	Yes
6	Log-scaling	Normalization	Univariate	Yes
7	Box-Cox Transform	Normalization	Univariate	Yes
8	Ignore Missing	Missing Data	Univariate	No
9	Reserved Value Imputation	Missing Data	Univariate	No
10	Mean or Median Imputation	Missing Data	Univariate	Yes
11	Piecewise linear interpolation (LI)	Missing Data	Univariate	Yes
12	piecewise cubic spline interpolation	Missing Data	Univariate	Yes
13	Autoregressive (AR, MA, ARMA & ARIMA)	Missing Data	Univariate	No
14	Vector Autoregressive (VAR)	Missing Data	Multivariate	No
15	Hot Deck and Cold Deck Imputation	Missing Data	Multivariate	Yes
16	Expectation Maximization	Missing Data	Multivariate	Yes
17	Linear regression and Stochastic Linear Regression	Missing Data	Multivariate	No
18	K-Nearest Neighbor (kNN) Imputation	Missing Data	Multivariate	No
19	Support Vector Machines (SVM) Imputation	Missing Data	Multivariate	Yes
20	Long Short Term Memory (LSTM) Imputation	Missing Data	Multivariate	No
21	Multiple Imputation (MI)	Missing Data	Multivariate	Yes
21	Hybrid Missing Type Based Approach	Missing Data	Dependent	Yes
22	Median Absolute Deviation (MAD)	Outlier Detection	Univariate	Yes
23	Grubbs Test	Outlier Detection	Univariate	Yes
24	Generalized Extreme Studentized Deviate (GESD)	Outlier Detection	Univariate	Yes
25	Interquartile Range (IQR)	Outlier Detection	Univariate	Yes
26	Minimum Covariance Determinant	Outlier Detection	Multivariate	No
27	Olive-Hawkins estimate	Outlier Detection	Multivariate	No
28	Local Outlier Factor (LOF)	Outlier Detection	Multivariate	No
29	Density-Based Spatial Clustering of Applications with Noise (DBSCAN)	Outlier Detection	Multivariate	Yes
30	Isolation Forest (iForest)	Outlier Detection	Multivariate	Yes
31	One Class Support Vector Machine (OCSVM)	Outlier Detection	Multivariate	No
32	Kalman Filter (KF) & Unscented KF	Sensor Fusion	Multivariate	No
33	Particle Filter (PF)	Sensor Fusion	Multivariate	No
34	Dempster-Shafer	Sensor Fusion	Multivariate	No
35	Equal-Width Binning	Discretization	Univariate	Yes
36	Equal-Width Binning	Discretization	Univariate	Yes
37	Gaussian Approximation	Discretization	Univariate	Yes
38	K-means Discretization	Discretization	Multivariate	Yes
39	Shared Nearest Neighbor (SNN)	Discretization	Multivariate	No
40	Self Organizing Map (SOM)	Discretization	Multivariate	Yes
41	Class-Attribute Contingency Coefficient (CACC)	Discretization	Univariate	No
42	Chi-Merge	Discretization	Univariate	No

Table 2
Preprocessing Techniques Covered in Survey - Continued.

#	Technique	Category	Input Type	Tested
43	Class-Attribute Interdependence Maximization (CAIM)	Discretization	Univariate	No
44	Low Variance Filter	Feature Selection	Univariate	No
45	Correlation-based Feature Selection (CFS)	Feature Selection	Multivariate	No
46	F-test	Feature Selection	Univariate	Yes
47	Relief & RReliefF	Feature Selection	Multivariate	Yes
48	Laplacian Score	Feature Selection	Univariate	Yes
49	Neighborhood Component Analysis (NCA)	Feature Selection	Multivariate	Yes
50	Minimum Redundancy and Maximum Relevance (mRmR)	Feature Selection	Multivariate	No
51	Genetic Algorithm	Feature Selection	Multivariate	No
52	Principle Component Analysis (PCA)	Feature Extraction	Multivariate	Yes
53	Linear Discriminant Analysis (LDA)	Feature Extraction	Multivariate	Yes
54	Piecewise Aggregate Approximation	Compression	Bivariate	Yes
55	Piecewise Linear Approximation	Compression	Bivariate	Yes
56	Gorilla compression	Compression	Multivariate	Yes
57	ZIP / deflate	Compression	Multivariate	Yes

however, covering all techniques under some categories and sub-categories is not feasible. Our selection criteria prioritize techniques recommended by the research community [4,59,60,137] and recommended or practically validated by industry (site MATLAB, Google, Scikit, etc.). Thus, selected techniques may not necessarily be considered state-of-the-art, but instead they are recognized and proven to be impactful and effective in practice. Our selection criteria are also governed by the availability of the technique in MATLAB or by a third-party library developed for MATLAB, especially for categories and sub-categories that included an empirical analysis. We also selected techniques that are applicable to numerical time-series data. Under each category and sub-category, we start with the simplest techniques or technique types and gradually present more complex techniques. We also mention some state-of-the-art solutions and highlight techniques that are proposed for distribution to the edge. Our search criteria included "missing data imputation", "outlier detection", "data discretization", "feature selection", "feature reduction", "sensor fusion", "data sampling", etc. The research databases used for our search included "IEEE Xplore", "ACM Digital Library", "a-z.lu", "ScienceDirect", etc. We selected techniques based on the following criteria:

(1) Techniques that are thoroughly tested and extensively used by academics, researchers and practitioners.
(2) Techniques applicable on time-series numerical data.
(3) Techniques with the potential of being effectively distributed to the edge.
(4) For the empirical analysis, use techniques with available implementation details or existing libraries in MATLAB.
(5) Techniques that are state-of-the art and at least satisfy point 2 and for the empirical analysis, also satisfy point 4.

Criterion 4 stated above ensures the reproducibility of our test results as the same implementations of different preprocessing techniques are readily available for us and for other researchers and evaluators. Most of the preprocessing techniques used and tested in both parts of the survey, including those that are part of the generalized preprocessing plan are available by default or as libraries in MATLAB R2020b. Using the selected dataset, which is publicly available [193], and applying the generalized preprocessing plan of our empirical analysis, presented in Fig. 1, one can obtain results similar to our results that are presented in this extended survey. Tables 1, 2 list the techniques we cover in depth or with an empirical evaluation and highlight the extended categories we cover in this survey. For a comprehensive view of the data preprocessing

Table 3
Tested Normalization Techniques.

Technique	Formula	Property	Results
Min-Max	$x'_i = \frac{x_i - min(X)}{max(X) - min(X)}$	Gaussian Distributions, Known Minimum and Maximum, Sparse Values	RMSE **0.46** MAE **0.29** MAPE **30.46%**
Z-score	$x'_i = \frac{x_i - \mu}{\sigma}$	Minimum Outliers, Non-gaussian Distributions	RMSE **0.35** MAE **0.25** MAPE **27.26%**
Robust Standardization	$x'_i = \frac{x_i - median}{Q75 - Q25}$	Tolerates Many Outliers	RMSE **0.38** MAE **0.25** MAPE **27.57%**
P-Norm	$x'_i = \frac{x_i}{[\sum_{k=1}^{N} \lvert x_k \rvert^p]^{\frac{1}{p}}}$	Tolerates Outliers	RMSE **0.57** MAE **0.46** MAPE **49.75%**
Decimal Scaling	$x'_i = \frac{x_i}{10^k}$	Known Maximum, Preserves feature values	RMSE **0.45** MAE **0.30** MAPE **32.42%**
Log Scaling	$x'_i = log_a(x_i)$	Power Law distribution, Non-negative Values	RMSE **0.52** MAE **0.36** MAPE **33.77%**
Box-Cox	$x'_i = \begin{cases} \frac{x_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln(x_i) & \text{if } \lambda = 0 \end{cases}$	Skewed and Poisson Distributions, Non-negative Values	RMSE **0.48** MAE **0.31** MAPE **28.55%**

categories, we included Fig. 4 in the appendix, which is the first data preprocessing taxonomy that we have proposed towards normalizing and unifying the definition and practices of the data preprocessing phase.

Normalization

par **Impact**: Normalization is a preprocessing step crucial and effective for distance based algorithms including K-Nearest Neighbors (K-NN), K-means clustering and Support-Vector Machine (SVN) as it prevents biases towards features with higher magnitudes. It also accelerates the convergence of gradient descent in linear regression, logistic regression and artificial neural networks as it generates a cost function with circular contours making the path to a global minimum more direct. Principle Component Analysis (PCA) used for feature extraction is an example where normalizing features before being used as input for the algorithm is essential to the performance of the algorithm. This is because normalization would prevent features with wider scales from dominating the direction of maximum variance and consequently being determined by PCA as more important features.

Normalization equalizes the influence and importance of feature scales and thus prevents wide ranges or higher magnitudes from having greater influence on learning [3,26,78,109]. In [185], empirical results presented indicate that standardization reduces the interquartile range of transmission time and predication time, in other words more stable and persistent transmission and prediction durations.

Definition: Normalization techniques rescale the values of numeric features to produce features with similar range of scale. It is applicable on multiple features datasets that have significantly different scales.

Techniques: Several normalization techniques exist that can be used on numerical sensor data at the edge using local or global data statistics. An overview of tested techniques can be found in Table 3. In *P*-Norm normalization, each element is scaled between the range zero to one using the magnitude measures L1-norm (Manhattan distance) or L2-norm (Euclidean distance) [4]. Min-Max is a scaling technique that can be used when the upper and lower bounds of the dataset are known with few or no outliers and the data are approximately uniformly distributed across the minimum and maximum range. Min/Max scales each feature between the range {0,1} using its minimum and maximum values [4,59,69]. Log scaling is effective with datasets that have

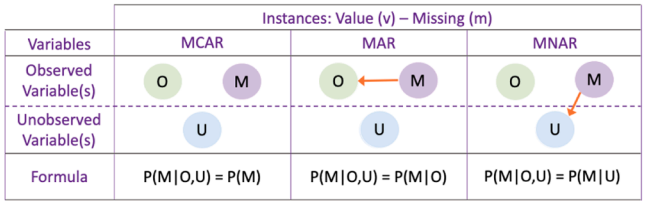


Fig. 3. Dissertation Structure.

dominant values while the rest of the values have few data points. Such dataset is not linear but follow the power law distribution (heavy-tailed distribution). Linearization of the power law distribution is possible by taking the logarithm of variables with skewed distributions to compress the range of large values and expand the range of small values thus transforming the distribution closer to a gaussian distribution [4,69]. Box-Cox transform [17] is a power transform and a generalization of log scaling that creates a monotonic transformation and a distribution more resembling to normal distribution to stabilize variances. Power transforms also improve the validity of association measures between variables such Pearson Correlation. Similar to log scale, Box-Cox transform only works with positive values. The transformation or power parameter (λ) is chosen using maximum likelihood, goodness-of-fit or Bayesian methods. If λ is zero then log scaling is applied on the target variable [4, 17].

The equation demonstrating how the new distribution is obtained can be found in Table 3. Standardization (z-score) transforms data with Gaussian distributions to a standard distribution with zero mean and unit variance (standard deviation of 1) [3,69,78]. It is effective on data distributions that exclude extreme outliers [69] and used for algorithms that assume gaussian distribution of the input data such as linear and logistic regressions. For image data, the mean can be calculated per image or for the whole image set. Z-index uses the mean (μ) and standard deviation (σ) of the dataset to standardize each element. Using the mean absolute deviation instead of the standard deviation provides more robustness to outliers because the deviations from the mean are not squared thus reducing the effects of outliers [59]. Robust standardization is used to standardize datasets with outliers as the technique ignores outliers in the calculations by using the median ($Q50$) and the interquartile range ($Q75 - Q25$) [166]. Decimal scaling reduces decimal values to be less than one by moving decimal points based on the maximum absolute value to the left [4,78]. Decimal scaling transforms the scale range to [-1,1] and keeps the original values of the features but with decimal points shifts. Z-score and robust standardization can be deployed to the edge and executed efficiently due to their low computational and storage requirements. Normalizing streaming data at the edge is challenging as the data are volatile with evolving statistical properties [146]. To overcome this challenge authors in [73,146] proposed adaptive normalization. This approach uses fixed-sized sliding windows, a concept popular in data streaming, to compute statistics within each window and thus consider seasonality and volatility. Most of the normalization techniques discussed (z-score, Min-Max, etc.) are also reversible; meaning the original values can be restored in the cloud if the statistical properties used at the edge are available. Thus, we recommend moving adaptive normalization to the edge to be executed by smart sensors or other smart devices. It is also important to mention that multiple different normalization techniques can be used sequentially on the same data to achieve different effects. For example, using Box-Cox to change the data distribution of features (e.g., to a gaussian distribution) and Z-score to change the scale of features.

Normalization functions are simple. Apart from traversing the content (tuples) of a window, they do not involve looping, recursion or matrix operations. Thus, they can be distributed to small edge components for execution (Raspberry Pi, smart sensors) without overloading the components. The parameters of normalization techniques (e.g., the mean and standard deviation for the z-score) are extracted during AI training. The same parameters from training are then used for testing and operations (on new data). These characteristics render normalization ideal for distribution to the edge where they can be applied immediately as the data are collected. The distributed transformation is perfumed using pre-defined parameters with the possibility of performing regular parameter updates to reflect changes in the data and application. In our experiments [184,185], we distributed min-max and z-score and thus proved the feasibility of distributing normalization techniques to the edge. The experimental results also highlighted the normalization techniques' impact in controlling and reducing variations in transmission time and bandwidth usage.

Empirical Results: The graphs in Figs. 5 and 6 show the impact of the normalization techniques on the distribution and scale of the data while graphs in Fig. 7 demonstrate the impact of the different techniques on the performance of the LSTM networks. The different tested techniques generate different scales. Min-Max produces the same scale (0, 1) for all features while the rest of the techniques produce similar scales for the different features. Log Scale and Box-Cox also change the distribution of features and can be used to obtain gaussian distributions for features with other distributions and where the predictor model assumes gaussian distribution of the data. Standardization (z-score) and robust standardization generated LSTM networks that produced the most accurate predictions as shown in Table 3; compared to an average RMSE of $\sim$ 0.69, average MAE of $\sim$ 0.57 and an average MAPE of $\sim$ 59% when the input data are not normalized. Yielding an improvement in accuracy by $\sim$ 49% for RMSE, $\sim$ 56% for MAE and $\sim$ 54% for MAPE when using z-score to normalize the data compared to unnormalized.

Data cleaning

Impact: Anomalies in data could result in errors in the value extraction process or misleading and incorrect information and inferences about the real-world resulting in costly consequences. In addition, anomalies may propagate the dataflow pipeline and their presence and impact can be amplified in results and models thus the need to address them thoroughly and early in the pipeline.

Definition: Data Cleaning is the process of identifying and handling anomalies that render the data of poor quality including missing data, outliers and noisy data [59,92,116].

Missing data impact

The missing data problem is one of the most common data quality problems found in sensor data. The presence of missing instances hinders the performance of models, causes errors (e.g., NaN propagation) and introduces biases [59,109]. Statistical operations are not feasible with *null* values. Using reserved values and categories to represent missing data leads to biased outputs and misleading results [79]. There are three types of missing data:

- Missing Not At Random (MNAR): the probability of a missing value depends on the unobserved variable value [46,157]. For example, the probability of missing temperature values is dependent on whether the unobserved values are higher than a maximum threshold or not.
- Missing At Random (MAR): the probability of a missing value depends on other observed variable(s) [46,157]. For example, the probability that temperature values are missing in a sensor network is dependent on high humidity values observed.
- Missing Completely At Random (MCAR): The probability of missing value is independent from observed and unobserved values [46, 157]. For example, all the temperature sensors in a sensor network has a missing value rate of 7%.

Figure 3 provide further illustration of the difference between the three missing data types and their mathematical formula.

Table 4
Tested Missing Data Imputation Techniques.

Technique	Formula	Property	Results
Mean	Replace each missing instance with: $\mu = \frac{\sum_{i=1}^{n} x_i}{n}$ where n is the number of instances	Constant, Statistical Based	RMSE **0.54** MAE **0.37** MAPE **39.03%**
LI	Given (x_0, y_0) and (x_1, y_1), find the missing attribute y of instance (x, y) via: $y = \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0}$	Linear Data, Statistical Based	RMSE **0.47** MAE **0.30** MAPE **34.84%**
CS	Given the cubic polynomials $p_k(x)$ on intervals x_k, x_{k+1}, a cubic spline (S) is interpolated under the conditions: 1) Each polynomial pass through its endpoint $p_k(x_k) = a_k x^3 + b_k x^2 + c_k x + d_k = y_k$ AND $p_{k+1}(x_{k+1}) = a_{k+1}x^3 + b_{k+1}x^2 + c_{k+1}x + d_{k+1} = y_{k+1}$ 2) The first derivatives at the middle points match (continuous at S'): $p'_k(x_{k+1}) = p'_{k+1}(x_{k+1})$ 3) The second derivatives at the middle points match (continuous at S''): $p''_k(x_{k+1}) = p''_{k+1}(x_{k+1})$ 4) The second derivatives at the end points are equal to 0 $p''_1(1) = 0$ & $p''_{m-1}(x_m) = 0$ where m is the number of points	Gaussian Distribution, Statistical Based	RMSE **0.49** MAE **0.32** MAPE **35.60%**
Hot Deck	1) Find similar instances $S = s_1, s_2, \ldots, s_n$ in the dataset$D$. E.g., such that If $s_i \& s_j \in S \Rightarrow s_i(Month) = s_j(Month)$ 2) Take the average value of the similar instances to impute the missing: $s_{missing}(Temperature) = \frac{\sum_{i=1}^{n} s_i}{n}$ 3) Repeat for all missing instances.	Nearest Neighbor Approach, Statistical Modeling	RMSE **0.49** MAE **0.33** MAPE **35.46%**
EM	Given the observed values (X), the missing values (Z) are computed via: 1) Initialize (θ): $\theta = \theta^t$ 2) E-step: $Z^{(t)} = E(Z\|\theta^{(t)}, X)$ 3) M-step: $\theta^{t+1} = argmax_\theta Q(\theta, \theta^t)$ When $Q(\theta, \theta^t) = \sum_Z p(Z\|X, \theta^t) \log p(X, Z\|\theta)$ (discrete) or $Q(\theta, \theta^t) = \int p(Z\|X, \theta^t) \log p(X, Z\|\theta)\, dZ$ (continuous) and where (t) is the iteration count. 4) Iterate steps 2) and 3) until $\|\theta^{t+1} - \theta^t\| < threshold$. 5) Impute missing data: $Z = E(Z\|X, \theta_{optimal})$.	Exponential Family Distributions, Statistical Based	RMSE **0.34** MAE **0.23** MAPE **27.17%**

Missing Data Techniques: A plethora of techniques exist to handle missing data making it an extensively researched area. Tables 4 and 5 provide a summary of tested missing data techniques and their empirical results. The theoretical analysis in the next paragraphs cover more techniques. We refer to the paper [1] for a comprehensive survey on Imputation techniques dedicated for IoT.

Univariate techniques only consider the variable to be pre-processed for missing data. The most trivial technique sample dropping or Ignore Missing (IM), which discards entire rows containing missing values [59,78,109]. IM is a simple solution, however similar to random sampling, it is effective when the complete observed sample represents the original or entire dataset. IM is recommended when data are MCAR as removing incomplete samples would not introduce biases [59,78]. The number of variables and the percentage of missing values are also factors in determining the applicability of IM. Dropping rows with one missing variable and several observed variables

(e.g., 20 observed features) results in significant loss of information even when IM would result in the removal of less than 10% of the samples.

Another approach to handle missing data is to recover the missing values via substitution or estimation. This approach is called missing data imputation. Techniques under this category are more effective as they retain all samples and attempt to produce a plausible recovery of the missing instances. Imputing missing values with reserved categorical or continuous values is the simplest univariate imputation technique. It, however, introduces biases and reduces the variability of the dataset [3, 116]. Using the next or previous values are also traditional imputation solutions. Their use is limited as they are not effective in exerting patterns and seasonality.

Using the mean, for normal distributions, or the median, for skewed distributions, to fill missing instances are popular univariate imputation techniques [3,109,116,157]. The mean or median imputation, however, results in an underestimated variance when the dataset consists of many missing instances and adds biases when used on data that are MAR or MNAR [46,157].

Piecewise linear interpolation (LI) is often used for time-series linear data to find a missing value between two points. For non-linear data, piecewise cubic spline interpolation (CS) could provide more accurate results and it is relatively more efficient than polynomial interpolation. The cubic spline and both its first and second derivatives are continuous functions providing for a smoother curves thus better approximations for non-linear data [160].

Another class of univariate imputation techniques is stationarity models for time-series data. To build models (e.g., linear regression) under this category, stationarity in time-series data must be strong or weak. Strong stationarity is when moving a window across the dataset yields similar distributions while a weak stationarity would yield a similar mean and finite covariance. Stationarity models include the Autoregressive (AR) technique which uses observations from a previous time step to predict a missing or unknown value of the next time step. Previous observations are called value lags and can span one or n lags(s). The notation AR(n) indicates the number of previous lags considered. Assuming AR(1), then each observation x_t is dependent on the previous observation x_{t-1} and consecutively, instance x_{t-1} is dependent on x_{t-2}; repeating this calculation recursively until reaching the first observation. This makes AR a long memory model, in which all value lags have an effect on predicting the missing or unknown value but the effect is reduced with older value lags. Moving Average (MA) Models, replace value lags with error lags, other literature call them innovations, and thus predicting a missing or unknown value depends on n previous error lag(s) of previous predictions. MA is a short memory model as older error lags stop having an effect on predictions with time. Autoregressive Moving Average Model (ARMA) includes both AR value lags and MA error lags thus it is a short and long memory model. Data with trends, seasonality and volatility shifts are not stationary and thus AR, MA and ARMA are not applicable. Using differencing removes trends and

Table 5
Tested Missing Data Imputation Techniques - Continued.

Technique	Formula	Property	Results
SVM	**Training**: find the cost function $f(x)$ s.t., regularized hinged loss function using the training dataset. 1) The Primal Formula: $w* = \frac{1}{2}w'w + C\sum_{n=1}^{N}(\xi_n + \xi_n*)$ where w^* is the margin to be optimized, ξ_n and ξ_n^* are the slack variables that allow for soft margins, N is the number of instances and C is the regularization variable that controls sensitivity to errors and avoids overfitting. 2) Dual Formula: A Lagrange function is constructed from the primal function to facilitate the optimization problem by finding the coefficients that minimize: $L(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i*)(\alpha_j - \alpha_j*)G(x_i, x_j) + \varepsilon\sum_{i=1}^{N}(\alpha_i - \alpha_i*)\sum_{i=1}^{N}y_i(\alpha_i* - \alpha_i)$ where α_n and α_n^* are non-negative multipliers for each instance x_n and $G(x_i, x_j) = e^{-\|x_i - x_j\|^2}$ is the kernel function that maps the instances into higher dimensional space to obtain a nonlinear SVM regression model. The dual formula is subject to the following constraints: a) $\sum_{n=1}^{N}(\alpha_n - \alpha_n*) = 0$ b) $\forall\, n$: $0 \leq \alpha_n \leq C$ c) $\forall\, n$: $0 \leq \alpha_n^* \leq C$ 3) The solver algorithms above are computed iteratively until convergence via a convergence criterion: $\Delta = \frac{w* + L(\alpha)}{w* + 1} <$ *threshold* **Predicting**: missing values are imputed via: $f(x) = \sum_{n-1}^{N}(\alpha_n - \alpha_n*)G(x_i, x_j) + b$	Non-linear Modeling technique	RMSE **0.48** MAE **0.33** MAPE **35.24%**
MI with (PMM)	**Imputation**: 1) I-Step: $Y_t*P(Y_{mis}\|Y_{obs}, \theta_{t-1}*)$ 2) P-Step: $\theta_t*P(\theta\|Y_{obs}, Y_{t-1}*)$ Repeat the imputation phase m times **Analysis**: Perform regression analysis on m datasets. **Pooling**: $\tilde{\theta} = \frac{1}{m}\sum_{i=1}^{m}\hat{\theta}_i$, pooling the standard errors requires the following: 1) Within imputation variance: $V_W = \frac{1}{m}\sum_{i=1}^{m}SE_i^2$ 2) Between imputation variance: $V_B = \frac{1}{m-1}\sum_{i=1}^{m}(\hat{\theta}_i - \tilde{\theta})^2$ 3) Pooled standard error: $SE_p = \sqrt{V_W + V_B + \frac{V_B}{m}}$ Apply the above steps to pool the covariance matrices.	Non-Gaussian Distribution	RMSE **0.42** MAE **0.29** MAPE **31.13%**

seasonality, however this requires an additional preprocessing step. Autoregressive Integrated Moving Average Model (ARIMA) includes the additional differencing step to achieve stationarity followed by fitting an ARMA model. Other disadvantages of ARMA and ARIMA models include the challenge of choosing optimal parameters and their tendency to overfit the data [16,192].

Multivariate techniques use multiple features (predictors) to estimate the missing instances of the response variable. Vector Autoregressive (VAR) is a generalization of the AR model for multivariate time-series data. VAR model includes the value lags of the response variable, the value lags of the predictors and the error lags. Several estimators can be used to estimate the regression coefficients including Maximum Likelihood and Bayesian analysis [77,122]. VAR as an imputation technique is only applicable on stationary time-series data.

Hot deck imputation finds complete samples where the predictors have similar values and calculate the mean or median of their response variable values to fill the missing instances [78,107,109]. Given it is a multivariate technique, hot deck imputation is computationally more expensive than univariate techniques, which corresponds to more computational and storage capacities as requirements when choosing a deployment edge device. It underestimates standard errors and produces bias estimates [49]. Cold deck imputation is a variation of Hot deck where samples are taken from a different dataset.

Expectation Maximization (EM) is a traditional multivariate imputation technique that finds the Maximum Likelihood Estimate (MLE) or the maximum a posteriori (MAP) estimate from datasets with missing values or models that depend on latent variables. In this context there are two unknowns, the missing values and the parameters (θ), i.e., mean (μ), covariance (ϵ) or standard deviation (σ), of the probability distribution. EM algorithm tackles the problem by initializing the parameters with plausible values and iterating through two steps; the Expectation (E) Step and the Maximization (M) Step. The E-Step computes the expectation of the posterior distribution of the missing values given the observed values and the initialized parameters [11,37,59]. It uses the initialized parameters to build models (e.g., stochastic regression models) to predict the missing values from observed values [49]. The M-Step maximizes the expected value of the log likelihood based on the complete data obtained from the E-Step [37,49]. EM iterates the E-Step and the M-Step until convergence; that is when changes in θ values are zero or smaller than a threshold. The optimal parameters ($\theta_{optimal}$) are then used to impute the missing values as the last step. EM algorithm is used as an imputation technique for data missing at random (MAR) and performs best on datasets with exponential family probability distributions (e.g., normal, exponential, gamma, etc.). Under these conditions EM provides less biased results compared to the techniques previously discussed. It can be used to impute missing values of discrete and continuous data and it is simple and easy to implement [44,59]. Datasets with large number of variables and missing values result in a slower convergence of the EM algorithm and increased probability of getting a local maximum rather than the global maximum and thus the MLE is not guaranteed. EM can also suffer from overfitting and its performance depends on the initialization of the parameters [44,46,59].

Machine learning (ML) models and Artificial Neural Networks (ANN) as imputation techniques have recently gained the attention of the research community. Linear regression is used to impute missing values of continuous data. It involves using a complete-case subset and estimating the regression coefficients for each covariate. The fitted model is then used to predict the missing values of the response variable. In deterministic regression imputed values have high correlation with the observed values of the predictors. This overestimation of correlation translates to lack of variability and substantial biases that would otherwise not be present should the data be available (not missing) [3,49]. Stochastic regression adds noise to each imputed value to increase variability and reduce biases. The noise is a random selection from a sample of residuals (residual distribution) of the regression model. The residual distribution is a normal distribution with mean of zero and variance equal to the residual variance. Stochastic regression underperforms if applied on datasets with heteroskedasticity, where the variance of errors is not constant along the regression line. Imputation via stochastic regression attenuates standard error as it ignores additional sampling error introduced from missing values [46,49].

K Nearest Neighbor (kNN) imputation has two variations; kNN classification for discrete data and kNN regression for continuous data. A distance metric (e.g., Euclidean, Manhattan, Mahalanobis, Minkowski, etc.) is used to determine feature similarity between samples and together with the k parameter, the neighbors for a given sample is determined. A kNN model is constructed from the training set based on the predictors and the specified k value. The model is then used to predict the missing values of the response variable. The k parameter is key to the prediction accuracy. Small k values render predictions subject to distortion caused by noise and outliers while too large k values diminish categories with small number of samples and increases biases. In kNN regression, the imputed value of a response variable is the average of neighbor values. kNN is a robust and effective imputation technique given the right k value. Its performance, however, suffers with large datasets as it traverses the entire dataset to calculate distance based similarities. A kNN classifier must be created for each variable with missing values that require imputation [46,59].

Support Vector Machines (SVM) is a multivariate and non-linear modeling technique with two variations; SVM regression (SVR) to impute missing instances of continuous variables and SVM classification (SVC) to impute missing instances of categorical data. The technique is recommended when non-linear relationships exist in datasets. SVM attempts to learn the decision boundary of the training data by fitting a partition hyperplane, or hypersphere depending on the dimensionality, that maximizes the margin of separation from the origin. For non-linear classification and regression, the data are implicitly mapped into a higher dimensional feature space by using non-linear kernel functions [172,217]. Examples of Kernel functions include polynomial kernel, gaussian radial basis function kernel and sigmoid kernel. The implicit mapping of data avoids actual transformation to the higher dimensionality feature space but rather computes the inner product in the feature space by evaluating the kernel function thus improving computational efficiency and space usage [78]. The quadratic problem is solved by using Lagrange multipliers and setting the derivates of the equation's variables equal to zero [172]. SVM performance deteriorates with larger number of variables. Choosing the kernel function is dependent on the dataset and application.

Long Short Term Memory (LSTM) is a recurrent neural network (RNN) that models sequential or time-series data with memory capacity to capture long term dependencies. LSTM was proposed to address the diminishing gradients issue suffered by RNN. A central feature of LSTM is the Constant Error Carousel (CEC), which is the control of the error flow that prevents diminishing gradients thus bridging long time lags (e.g., 1000 time steps). LSTM maintains an internal state via recurrently connected memory blocks. Each block contains one or more memory cell (s) with recurrently connected CEC. The CEC is extended with three gates called input, output and forget gates. The forget gate limits internal state growth thus discarding obsolete information. The output and input gates control access to constant error flow through the CEC thus controlling retention degree of the current state and the flow of information passed forward [176,178]. In recent years, several solutions have been proposed that extend LSTM to address missing data specifically massive missing percentage of data or large blocks of data in time-series [56,176,215]. LSTM can be used as multivariate imputation technique for non-linear data where historic information maintains its value. LSTM produces relatively better results with non-monotonic and fluctuating data, however, it is a complex algorithm that does not yield optimal results in all forecasting time-series applications as demonstrated by the authors of [62]. Authors of [56] proposed an LSTM based model to address MNAR instances of massive volumes. They incorporated forward and backward time intervals and the missing rate of the response variables in their model, thus increasing its adaptability to massive data and improving its imputation accuracy. In [130], authors proposed a hybrid LSTM that performs bi-directional imputations. The empirical evaluation of their method indicate improved imputation accuracy and better restoration of information; however, the improvement range is wide and is dependent on experimental and contextual details.

Other types of ANN were exploited to impute missing data including in [22] where authors proposed a Generative Adversarial Network based imputation method and in [95] where a Convolutional Autoencoder based model was proposed to impute missing instances. To obtain more accurate imputations, ensemble imputation techniques, which uses multiple ML techniques, were proposed such as Random Forest [78,89,179].

Multiple Imputation (MI) [168] differ from the previous techniques discussed in that each missing value is imputed with multiple values generating multiple datasets with different plausible values for the missing data. Thus, MI techniques explicitly account for uncertainty associated with missing data. MI is divided into three phases; the imputation phase, analysis phase and pooling phase. The imputation phase consists of two steps the I-Step and the P-Step. The I-Step predicts the missing data from the observed data using a model with a stochastic component such as stochastic regression. The P-Step is a Bayesian analysis that uses the imputed dataset from the previous step and a priori model as prior belief to describe the posterior distribution of the parameters. Using Monte Carlo sampling or Markov Chain Monte Carlo (MCMC) methods, new estimate parameters (mean vector and covariance matrix) are randomly drawn from the estimated posterior predictive distribution. The new parameters randomly differ from the parameters of the previous iteration and are used in the next iteration of the I-Step to obtain a model with new regression coefficients and thus estimating new imputations and creating a new imputed dataset. The imputation phase is repeated m times to create m different imputed datasets. Choosing the right m value depends on the percentage of missing instances in the dataset and the size of the dataset. In the analysis phase, m sets of parameter estimates and standard errors are obtained from the m imputed datasets using, for example, regression analysis. The pooling phase, involves combining the m parameter estimates into a single parameter estimate by taking, for example, the average. It also involves pooling the standard errors into a single standard error via, for example, Rubin's combining rules [46,49,180]. Predictive Mean Matching (PMM) is a variation of MI that imputes missing values of variables that are not normally distributed. During the imputation phase, predictions are made for both missing instances and observation and a pool of k potential replacements (donors) are defined for each missing instance. Donors are determined by selecting observations with predictions close to the prediction of the missing instance. A donor is then selected randomly from the k donor list and its observation value is used to impute the missing instance. The process is repeated for each missing instance [142]. MI avoids underestimating the standard error because the technique counts in the additional sampling error caused by having missing values. It is effective on data missing at

Table 6
Tested Outlier Detection Techniques.

Technique	Formula	Property	Results
MAD	1) Find: $MAD = b \times Median(\|x - \mu\|)$ 2) Detect outliers using the range: $\mu - (threshold \times MAD) < x < \mu + (threshold \times MAD)$	Symmetric Distribution, Statistical Based	RMSE **0.37** MAE **0.25** MAPE **28.91%**
Grubbs	For each potential outlier $i \in I$ perform the following hypothesis test: H_0: There are no outlier in the dataset H_1: There is exactly one outlier in the dataset 1) Find the Grubbs test statistic (two-tailed): $G_{test} = \frac{max_i^I \|Y_i - \widehat{Y}\|}{s}$ where $\widehat{Y}$ is the mean and s is the standard deviation. 2) Compute the critical value ($G_{critical}$) via: $\frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/(2N),N-2}}{N-2+t^2_{\alpha/(2N),N-2}}}$), where $t_{\alpha/(2N),N-2}$ is the upper critical value of t-distribution with N-2 degree of freedom. 3) Reject the null hypothesis if $G_{test} > G_{critical}$.	Gaussian Distribution, Statistical Based	RMSE **0.32** MAE **0.22** MAPE **26.70%**
GESD	1) Given $r$cycles, calculate the z-index of each element in $x$to find the Max z-index for each cycle: $T_{iMax} = \frac{\|x - \mu\|}{\sigma}$ 2) Calculate the critical value of cycle i: $\lambda_i = \frac{(n-i)t_{n-i-1,p}}{\sqrt{(n-i-1+t^2_{n-i-1,p})(n-i+1)}}$, where $i = 1, 2\ldots, r$ and $p = 1 - \frac{\alpha}{2(n-i+1)}$ (p^{th} percentile of distribution t.H_0 is rejectedwhen $T_{jMax} > \lambda_j$.	Gaussian Distribution, Statistical Based	RMSE **0.34** MAE **0.23** MAPE **27.82%**
IQR	1) Find the first quartile (Q_1), the median (Q_2) and the third quartile (Q_3) of feature j of the dataset. 2) The interquartile range is: $IQR = Q_3 - Q_1$ 3) An instance i is detected as an outlier if: $x_{ij} > UpperBound = Q_3 + (1.5 \times IQR)$ or $x_{ij} < Lowerbound = Q_1 - (1.5 \times IQR)$	Non-Gaussian Distribution, Statistical Based	RMSE **0.35** MAE **0.24** MAPE **28.41%**
DBSCAN	1) Find neighborhood for each point: $\forall\, p \in D$, $N_\epsilon(p) = \{q \in D \| dist(p, q) \leq \epsilon\}$ 2) Find core points: $CO = \{q \in D \| \|N_\epsilon(p)\| \geq MinPts\}$ Given $\{p \in D \| p \in N_\epsilon(q) \& \|N_\epsilon(q)\| \geq MinPts\}$ then $directReach(p, q)$ is satisfied and given $\exists\, p_1, \ldots, p_n$ where $p_1 = q$, $p_n = p$ such that $directReach(p_{i+1}, p_i)$ then, $reach(q, p)$ is satisfied and given $\exists\, v$ where $reach(v, p)$ and $reach(v, q)$ then, $densityConnected(p, q)$ is satisfied. 3) Find the clusters (C): $\forall\, p, q \in D$, if $q \in C$ & $reach(q, p)$, then $p \in C \,\forall\, p, q \in D$, if $densityConnected(p, q)$ then, $p, q \in C$ Outliers are defined as: $\{p \in D \| p \notin C_i \,\forall\, i = 1, \ldots, k\}$	Unsupervised, Density Based	RMSE **0.55** MAE **0.37** MAPE **37.96%**
IForest	Given the dataset $X = \{x_1, x_2, \ldots, x_n\}$ 1) Build k iTrees from a sample X' of Ψ instances by recursively splitting X' using a randomly selected feature q and a split value p. Repeat until the node has one instance or all instances have the same value. The number of external nodes will equal to Ψ. 2) The path ($h(x)$) of each instance in each iTree is computed. The path is the number of edges x traverses in an iTree starting from the root node until the traversal is terminated at an external node. 3) Find the anomaly score (s) from $h(x)$ values obtained from k iTree: $s(x, \Psi) = 2^{-\frac{E(h(x))}{c(\Psi)}}$ where $c(\Psi) =$ $\begin{cases} 2H(\Psi-1) - \frac{2(\Psi-1)}{n} & \text{for } \Psi > 2 \\ 1 & \text{for } \Psi = 2 \\ 0 & \text{otherwise} \end{cases}$ When s is close to 1 then x is likely to be an anomaly.	Unsupervised, Projection Based, Non-Linear Outliers	RMSE **0.38** MAE **0.26** MAPE **23.26%**

random if the influencer variables are included as input for the MI algorithm. MI is computationally intensive and requires large storage capacity as it generates *m* imputed datasets from originally large dataset which makes it an expensive technique [46,49,180]. As such, MI may not be deployable to edge components with limited computational and storage capacities.

In [186], authors conducted an empirical analysis to evaluate the performance of different imputation techniques on isolated and sequence missing instances. From their results, they suggest adopting a new hybrid approach that uses two imputation techniques from any type (univariate, multivariate, ML, DL, etc.). The idea is to choose an optimal technique for isolated missing instances and another technique that is optimal for sequence missing data. This provides flexibility in choosing the least expensive technique to be deployed at the edge. The approach yields better overall imputation accuracy and addresses possible perturbation in individual technique's performance.

In this survey, we have centrally evaluated a wide range of missing data solutions; from simple univariate imputation techniques to complex AI based techniques. Further empirical analysis conducted [184] also included distributing the univariate technique LI to edge components. Univariate imputation techniques can be distributed and executed at edge components without the prerequisite of data fusion. The distribution is also feasible for multivariate techniques, after data fusion, including AI based solutions. It is less complex if the trained models and networks are strictly distributed and not the whole training process. Distributing AI (training and prediction) to the edge is currently an active research direction (EdgeAI) [27], which also covers the distribution of AI based data preprocessing techniques. Few standalone solutions [57,197] have been proposed that are distributed versions or completely novel solutions of missing data imputation techniques designed for edge execution. A practical analysis of the performance and impact of the different imputation techniques distributed in IoT devices is yet to be proposed and thus we identify an empirical research gap in this area. Some missing data imputation techniques, such as MI and ensemble solutions, are intuitively expensive to distribute to the edge as they generate multiple plausible imputations using one or more technique(s) to minimize uncertainty in the final imputations. We also highlight the advantage of distributing missing data handling techniques in addressing the anomalies early in the pipeline and thus minimizing error propagation. In addition, missing data causes training and prediction to fail and thus complete input data must be made readily available for Edge AI to function correctly.

Empirical Results: Several techniques required configuration and setup. In addition to MATLAB libraries of missing data imputation techniques, we also used third-party libraries and developed our own. For the EM technique, we used the third-party library [171] implemented for MATLAB. For the SVM experiment, we trained a non-linear SVM model using the Gaussian Kernel. We created our own implementation of MI and used predictive mean matching (pmm) to replace imputations of a model with coefficients randomly drawn from a posterior predictive distribution of the P-Step. The MI imputation phase is repeated seven times. We imputed the missing data present in the real-world dataset (AirQuality) and thus there is no baseline observations for the missing instances. Fig. 8 presents the imputations of the different tested techniques. The time-series data are non-monotonic and frequently fluctuate rendering the imputation tasks difficult particularly for the tested univariate techniques as they do not capture patterns and seasonality present in the data. The plots in Fig. 9 present the predictions of the different models trained on datasets produced with different imputation techniques. The observations reflect the original data and the imputed data using the tested technique. As mentioned previously, we do not have a baseline for the missing instances as they are actual and not simulated. EM imputations included yielded the second best predictions after the hybrid approach proposed by [186] which uses optimal imputation techniques for each type of missing data (i.e., an optimal technique for isolated instances and another for blocks of missing data). We choose the combination EM to impute sequences (blocks) of missing instances and cubic spline to impute isolated missing instances. There is a slight improvement in using two techniques compared to using EM only. As expected, mean imputation produced the least accurate LSTM model in addition to adding biases to the data. Univariate techniques had comparable performances to the multivariate techniques with cubic spline outperforming most of the multivariate techniques for this dataset and experimental setup. This outcome is an advantage with regards to deployment within an IoT context as univariate techniques are more efficient at consuming resources (e.g., consume less energy, have less dependencies in that no data fusion is required, etc.). ML and DL solutions require the predictors to be free from missing instances for training to complete successfully without *Null* or *Not a Number (NaN)* propagation.

Outliers impact

are examples of noisy data that hinder knowledge extraction and models' performance [3,59]. Similarity measures can be falsified by outliers, which results in misclassifications [3]. Outliers can be classified as:

- Global Outliers: data points that significantly deviate from the entire dataset [78].
- Local Outliers: data points that are within the normal range of the entire datasets but deviate significantly from the surrounding data points (local area) [78].
- Contextual Outliers: also called conditional outliers, are data points that deviate significantly given a specific context (time, location, etc.) [78].
- Collective Outliers: data points that, together, deviate from the rest of the dataset[78].

Outliers Techniques: there are several approaches to detect outliers. Table 6 provides an overview of the techniques that were tested including their performance. While the following paragraphs detail further outlier detection techniques in addition to the ones tested.

Median Absolute Deviations (MAD) is a statistic based method. MAD is a robust scale univariate estimator that finds positive and negative deviations from the median of a sorted dataset [166]. MAD calculates the absolute difference between the median of the dataset and the data points and then multiplies the median by $\frac{1}{Q(75)}$ where $Q(75)$ is the 0.75 quantile of the distribution. The dataset's median, the MAD value and a threshold (e.g., 2, 2.5 or 3) are used to determine the range by which outliers are identified [115]. MAD assumes equal dispersion at both sides of the median (symmetry) thus it is less effective on skewed distributions [166]. It is possible to use MAD at the edge to find local outliers within a window.

Grubbs Test [71], also known as the Maximum Normalized Residual test is a univariate statistic based outlier detection technique applied on data with a gaussian distribution. With the presence of multiple outliers, the technique is iterative and produces new datasets that exclude a detected outlier in the previous iteration. Grubbs test involves evaluating the deviation of potential outliers from the mean by studentizing the instances. The studentized value is compared against a critical value to determine the significance level and whether the instance is to be rejected from the dataset or not (i.e., consider it as an outlier or not). The mean and standard deviation are calculated for each new dataset and a different critical value is used in each iteration [97].

The Generalized Extreme Studentized Deviate (GESD) test detects multiple outliers in univariate datasets with a normal distribution but only requires the upper bound for the number of outliers expected. GESD is defined for the zero hypothesis "There are no outliers in the dataset" and alternative hypothesis "There are up to r outliers in the dataset". The test uses the mean (μ) and standard deviation (σ) of the dataset to remove the data point with maximum z-score (T_{iMax}) and repeats the

step r cycles with the μ and σ of the new subset. A critical value is calculated for each cycle (λ_i). Starting with the last cycle, the maximum z-score (T_{iMax}) and λ_i are compared until the maximum z-score of a cycle r-j (T_{r-jMax}) exceeds its critical value (λ_{r-j}). T_{r-jMax} and the maximum z-scores of all prior cycles are considered outliers [134]. Having dynamic critical values or rejection threshold for each cycle offers better detection of outliers while minimizing false positives.

Interquartile Range (IQR) can be applied on multivariate and univariate data with non-Gaussian distribution to detect outliers. By finding the median (Q_2), the first (Q_1) and third (Q_3) quantiles are calculated to determine the IQR from their difference. The IQR is then used to define upper and lower bounds against which outliers are detected. IQR execution efficiency makes it deployable at the edge to detect outliers within a window but it may not necessarily detect all outliers [195].

Another class of outlier detection techniques is the Robust Estimators class. Maximum Likelihood Estimate and Log Likelihood Estimate are both sensitive to outliers. This means that the estimates are pulled towards existing outliers in the dataset. Mahalanobis distance measure is also sensitive to outliers. To calculate a multivariate mean and covariance of a gaussian distribution while being robust to the presence of outliers in the dataset, robust estimators are used instead. The general concept of robust estimators is to extract a subset that represent the underlying distribution and use it to estimate the mean and covariance. The Minimum Covariance Determinant (MCD) and the Olive-Hawkins estimate are examples of robust estimators of multivariate covariance that can be used to detect outliers by estimating the parameters of the normal data. Olive-Hawkins estimate uses the concentrations algorithm technique. The iterative process produces a sequence of estimates called attractors via k concentration steps. The final estimator is the attractor that optimizes the criterion for each parameter estimated (mean and covariance), which represent a distribution that exclude the impact of outliers [149].

Local Outlier Factor (LOF) [18] is a density based outlier detection technique with the intuition that the density around an outlier is significantly lower than the density around normal points. The technique starts by finding the k nearest neighbors of the data points using a distance metric and identifying the distance of the farthest neighbor. The neighborhood of each point is then determined using the k^{th} distance. The technique calculates the Local Reachability Density ($lrd_k(o)$) for identified neighborhoods using the number of neighbors within a neighborhood and the reachability distance of neighbors ($reachdist_k(\overline{o} \leftarrow o)$). Finally, the local outlier factor ($lof_k(o)$) is calculated for each data point using the reachability distance and local reachability density of its neighbors. The points with the highest $lof_k(o)$ are more likely to be outliers [18]. LOF can be applied on data with skewed distributions and can detect both global and local outliers. The technique, however, is computationally expensive and requires large memory space. Several modified versions have been proposed including Top-N local outlier detection (TOLF) which uses a pruning strategy to exclude normal points and only calculate ($lof_k(o)$) of potential outliers; making TOLF a more plausible choice for edge deployment than LOF [205].

Density-based spatial clustering of applications with noise (DBSCAN) [52] is another density-based technique, which is a clustering algorithm that finds arbitrary shaped and non-linearly separable clusters. DBSCAN is robust to outliers and can isolate global and local outliers. A priori requirements are limited to the maximum radium of neighborhoods (ϵ) and the minimum number of neighbors (*MinPts*). The algorithm identifies data points under three types; core, border and outlier and uses these identifications to find clusters. It starts by randomly selecting a data point and finding the number of neighbors that are within ϵ distance from it. If the number of neighbors is equal to or greater than *MinPts* then the data point is considered a core. The algorithm traverses the neighbors and performs the same check. If the former condition is not satisfied then the algorithm tests if the data point is an ϵ distance from a core point to determine whether it is a border point or not. Lastly a point is considered an outlier if it is not reachable from a core or border point. Reachable core points are clustered together with their border points. For outlier detection, DBSCAN underperforms on datasets with clusters of variable densities and on datasets with large dimensionality. The later weakness is mainly contributed to the Euclidean distance in what is known as the curse of dimensionality [173]. The term curse of dimensionality was first introduced in [9] and in our domain, refers to the need for exponentially more data with the increase in dimensionality (number of features). This because with high dimensional datasets, the available data become sparser. To address the time complexity weakness of DBSCAN ($O(n^2)$), OPTICS [5] and HDBSCAN, [23] extends DBSCAN to hierarchical clustering and AnyDBC [131] reduces the query range and label propagation times.

Isolation Forest (iForest) [124] is a projection-based learning technique that constructs fully random binary trees based on a subsampling size and two randomly selected parameters. The first random parameter is the splitting attribute (j) and the second is the splitting threshold (θ), which is uniformly selected from the [$min(x_j)$, $max(x_j)$] range of attribute j. The subsamples of a dataset are recursively split until every data point is in its own leaf node with the result of constructing an ensemble of *iTrees*. It is also possible to optimize the *iTree* construction by stopping the splitting earlier. After the construction of the *iTrees*, the average leaf depth is computed, which is used to calculate the anomaly score. The intuition of the algorithm is that outliers will be isolated faster by the splitting compared to nominal points. A score closer to 1 means the data point is likely to be an anomaly while a score less than or equal to 0.5 means the data point is likely to be nominal. The iForest algorithm consist of two phases; the training phase to construct the ensemble of *iTrees* and the prediction phase at which the anomaly score is calculated for test or new data. *iForest* is robust to swamping (non-outlier classified as an outlier) and masking (undetected outlier) and it remains effective with high-dimensional datasets with irrelevant attributes and training samples with no outliers [124]. The algorithm does not use a distance metric to calculate distances or densities thus having a linear time complexity and small memory requirement which is well suited for edge deployment. By having randomly selected j and θ, the technique avoid overfitting, however, biases are introduced in the anomaly score map as the splitting using the aforementioned parameters result in vertical and horizontal branch cuts. IForest performs well with high dimensional data [123]. The authors of Extended Isolation Forest (EIF) [80]address this limitation by performing the splitting based on branch cuts with random slopes and random intercept points. iForest was also extended to handle stream data in several work including iForestASD [43], which leverages the notion of sliding windows.

One class support vector machine (OCSVM) is a quantile-based unsupervised outlier detection algorithm. It attempts to learn the decision boundary of normal data by fitting a hyperplane that maximizes the margin of separation from the origin. Data points that fall outside the decision boundary or in other words close to the origin are considered as outliers. Learning the decision boundary and accounting outliers is achieved by the use of non-linear kernel functions that implicitly map data points into a higher dimensional feature space away from the origin. The implicit mapping avoids actual transformation of the data to the higher dimensional feature space but rather computes the inner product in the feature space by evaluating the kernel function thus improving computational efficiency and space usage. A regularization parameter controls the number of slack admitted, i.e., the number of data points allowed on the other side of the decision boundary and thus considered as outliers. The quadratic problem is solved by using Lagrange multipliers and setting the derivates of the equation's variables equal to zero [172]. Other variations of OCSVM include fitting a hypersphere with minimum radius to define the decision boundary of normal data and thus points outside the sphere are considered outliers [187]. Online and distributed variations of OCSVM were proposed by [212] in which they use linear optimization to fit a hyper-ellipsoid centered at the origin with minimum radius using Mahalanobis

Table 7
Surveyed Data Fusion Techniques.

Technique	Formula	Property
KF	**Prediction Step**: 1) Use the transition model to predict the new state from the previous state via: $x_{tp} = A_t x_{t-1} + B_t u_t + w_t$ where A_t is the state transition model, $B_t u_t$ is the effect of external factors and w_t is the process noise. 2) Find the prediction error via: $P_{tp} = A_t P_{t-1} A_t^T + Q_t$ where P_{tp} is the predicted process covariance matrix and Q_t is the process noise covariance matrix. **Correction Step**: 1) Obtain the measurement in the right format and considering the noise: $z_t = C_t x_t + v_t$ where v_t is the measurement noise and C_t is the transformation matrix (e.g., identity matrix) 2) Compute: $KG = \frac{P_{tp} H^T}{H * P_{tp} H^T + R_t}$, where R_t is the noise covariance matrix and H_t is the transformation matrix 3) Compute the current state's estimate using the prediction value, the measurement value and their weights via: $x_t = x_{tp} + KG[z_t - H_t x_{tp}]$ 4) Update the process estimate covariance matrix for the next iteration using: $P_t = (I - KG^* H_t) P_{tp}$ where I is the transformation matrix	Statistical Estimation, Assumes Linear Model and Gaussian Noise
PF	**Prediction step**: 1) The current state estimate for each particle in the previous set of particles is computed using the transition model (counting transition noise): $x_t^{[i]} = \pi(x_t \mid x_{t-1}^{[i]}, u_t)$ **Correction step**: 1) The observation model is used to compute the weight $w_t^{[i]}$ for each particle while counting observation noise via: $w_t^{[i]} = \eta[\frac{p(x_t \mid x_{t-1}^{[i]}, u_t)}{\pi(x_t \mid x_{t-1}^{[i]}, u_t)}] \propto p(z_t \mid x_t^{[i]})$ where z_t is the current observation with pre-condition that $p(x) > 0 \Rightarrow \pi(x) > 0$. 2) The current state particle set is updated with the particle and its computed weight via: $S_t = \{< x_t^{[i]}, w_t^{[i]} > \mid i = 1\ldots, J\}$ 3) The posterior distribution of the current state or belief $p(x_t)$ is: $p(x_t) = \sum_i^J w_t^i \delta_{w_t^{[i]}}(x_t)$, where δ is the Dirac delta function. **Resampling step**: 1) Select the first particle $< u_t^1, w_t^1 >= U_1$ randomly: $U_1 = S_t[0, \frac{1}{J}]$ 2) The other J −1 samples are then deterministically determined via: $\frac{1}{J}$ steps: $U_i = U_1 + \frac{i-1}{N}$ 3) A particle is selected for replication based on its weight and all selected particles are given equal weights via: $\overline{S_t} = \{< u_t^j, \frac{1}{J} >\} \mid \sum_{k=1}^{i-1} w_t^k \le u_j < \sum_{k=1}^{i} w_t^k$	Monte-Carlo localization, No Assumptions on Linearity or Distribution
D-S	Given a set of Frame Discernment $\Theta = \{\theta_1, \theta_2\}$, the Hypotheses set is: $2^\Theta = \{\varnothing, \theta_1, \theta_2, \theta_1 \cup \theta_2\}$ where $\theta_1 \cap \theta_2$ (mutually exclusive), $m(.): 2^\Theta \rightarrow [0, 1]$ (m is the mass function), $\sum_{\theta_i \in 2^\Theta} m(\theta_i) = 1$, and $m(\varnothing) = 0$ (i.e., the mass of an empty set is 0). **To determine the current state**: 1) Obtain degrees of belief for each hypothesis or question (θ_i): $Bel(\theta_i) = \sum_{\theta_j \mid \theta_j \subseteq \theta_i} m(\theta_j)$ 2) Obtain the plausibility for θ_i: $Pl(\theta_i) = \sum_{\theta_j \mid \theta_j \cap \theta_i \neq 0} m(\theta_j)$ or $Pl(\theta_i) = 1 - Bel(\overline{\theta_i})$, where $\overline{\theta_i}$ is the negation of θ_i. From (1) and (2) the confidence interval is obtained for θ_i: [$Bel(\theta_i)$, $Pl(\theta_i)$]. **To combine degrees of belief from multiple sensors**: $m_1(\theta_i) \oplus m_2(\theta_i) = \frac{\sum_{\theta_j \cap \theta_k = \theta_i \neq \varnothing} m_1(\theta_j) m_2(\theta_k)}{1 - (\sum_{\theta_j \cap \theta_k = \varnothing} m_1(\theta_j) m_2(\theta_k))}$	Bayesian Inference, No Assumptions on Probability Distribution

distance metric. Each sensor models its own hyper-ellipsoidal OCSVM within a window to detect local outliers and update the model as the data distribution changes. Model parameters are exchanged between sensors to derive global parameters from local outliers detected within a time window [212]. Authors in [207] also extended OCSVM to be more efficient for IoT applications by leveraging clustering and Gaussian mixture models to decrease prediction time and memory requirement.

Distributing Normalization and Missing Data Handling techniques to the edge requires the distribution of Outlier Handling techniques. This is due to a dependency that exist between the different preprocessing categories where outlier handling is the prerequisite. The dependency is not functional in that both normalization and missing data handling techniques can function with the presence of outliers, however, their performance is negatively impacted by outliers [186]. The main challenge with executing density-based and distance-based outlier handling techniques using edge computing is the lack of the whole dataset or sufficient data at a given window to identify global and local outliers. There are several examples in literature, however, of distributing outlier handling solutions to IoT components and applied as the data are collected including univariate techniques [184], multivariate and AI solutions [199,218]. Similar to the missing data category, AI based outlier handling techniques can be trained in the cloud and deployed to edge components to be executed on new data. Thus, the same parameters and coefficients used in training are used in deployment. There is a lack of sufficient empirical analysis that leverage edge computing to execute outliers detection techniques on sensor data streams. A comparison of the performance and impact of traditional and state-of-the-art techniques is identified as a future work.

Empirical Results: The diagrams in Fig. 10 show the outliers detected by the different tested techniques for the response variable (CO). We wanted to keep the consistency of using the same variable (CO) to present our results as it facilitates more comparisons of impact between input and output. Having said that, we also applied and tested the different techniques on the rest of the variables (predictors) to detect possible outliers. Due to space restrictions we suffice with presenting the outliers detected for one variable (CO). All instances of the dataset were included in the plot. Some techniques had different performances for each feature while other techniques detected overlapping or identical outliers. For example, GESD and Grubbs detected the same outliers for CO. Generally, the prediction accuracy from the GESD and Grubbs experiments were very similar because they identified similar outliers across the different features. Fig. 11 highlights the impact of removing the detected outliers from the predictors by the different outlier detection techniques on the prediction accuracy of the produced LSTM

models. Only the first 120 instances of the test dataset (the last 936 instances) are presented in the plots. For all the experiments, the detected outliers were replaced with plausible estimations using cubic spline interpolation (see Table 4). The tested techniques (univariate and multivariate) were applied as univariate to each feature separately. IForest required replacing *NaN* values to effectively detect outliers and thus the test for this experiment required changing the order of the DAG (see Section 1) to place outlier detection after missing data imputation. The configuration of the iForest experiments is similar to the configuration proposed by the authors of the original paper [124] and includes building 100 isolation tree with 256 instance samples with 10 rounds of repeat for each training. For the DBSCAN experiment, we set the $\epsilon = 0.01$ and number of neighbors $MinPts = 7$ as the best possible values for our dataset. From our results, removing outliers from the AirQuality dataset did not significantly improve the quality of the LSTM models with some outlier detection techniques having negative impact as the models slightly yielded higher error metrics. Grubbs performed best when considering all error metrics followed by iForest, which produced a far lower MAPE error value than the rest of the experiments. DBSCAN performed the worst introducing inaccuracies and causing deterioration in prediction accuracy. This could be because the dataset consists of clusters of varying densities, which hinders the effectiveness of DBSCAN. The results could be specific to the dataset and may indicate that the extreme values present in the dataset are not necessarily outliers and are providing valuable information to training. Despite the results, we recommend handling outliers due to their possible impact on model quality and impact on the effectiveness and performance of other pre-processing techniques that are sensitive to outliers such as missing data imputation techniques [186]. As future work, multivariate technique should be fully leveraged by using other variables to detect outliers.

Sensor fusion

Definition: Sensor or data fusion is the process of combining data from multiple sensors to produce more consistent, accurate, comprehensive, reliable and useful information than if provided by one of the data sources [?]. In this survey, we present fusion techniques that can be used to improve data quality including improving accuracy, addressing missing values and replacing noise or outliers. We consider data fusion under Dasarathy's classification, Data-In and Data-Out, and according to DARPA classification level 0 (source processing) and level 1 (object refinement). We also focus on techniques that can be implemented in decentralized architectures and are effective as non-batch processes [25]. An overview of the surveyed sensor fusion techniques can be found in Table 7.

Techniques: The Kalman Filter (KF) also known as linear quadratic estimation, is a recursive estimation technique optimal for systems with linear transition model, observation model and normalized gaussian noise. KF fuses low level data (observations) and state predictions to estimate the current state of a discrete-time process considering the prediction errors, observation errors and predicted covariance between variables. The technique consists of two main steps; the prediction step at which the state prediction (X_{tp}) is computed from the previous observation (X_{t-1}) considering, if applicable, the control factors and noise. Control factors are external factors that affect the current state such as surface inclination for moving vehicles. The prediction step is followed by the correction step at which the Kalman Gain (KG) is calculated from measurement error and the prediction error (the state covariance matrix in N-Dimensions problems). The KG indicates the kind of error to expect in the processing of the data to determine how much weight to put on the predicted and measured values when computing the current state. KG value close to one means there is more uncertainty in the predicted value and thus the measured value is given more emphasis when calculating the current state. Overtime, the KG becomes closer to zero indicating less uncertainty in the predicted value and thus having more weight and impact on the estimate of the current state. KF is an optimal solution for linear models with gaussian noises, however, it underperforms when observations are intermittent and data sources have unsynchronized clocks. KF does not handle heavy-tailed distributions and nonlinear systems [25,138,144]. Extended Kalman Filter (EKF) [99] and Unscented Kalman Filter (UKF) [194] were proposed to address non-linear systems. Compared to EKF, UKF provides more accurate approximations without the need to compute Jacobian matrices (reduced complexity). Instead it uses unscented transformation to pass a gaussian probability distribution through a non-linear function and onto a more complex distribution. The unscented transform starts with choosing deterministic sample points from the input distribution called sigma points, which have the same distribution as the input probability distribution. Weights are calculated for each sigma point based on a tuning parameter and the dimensionality of the problem. The sigma points are then passed through the non-linear transform function. The mean and covariance of the transformed weighted sigma points are used to calculate the new gaussian approximation distribution (new state estimate). EKF, however, assumes gaussian distribution [194] and remains computationally expensive for edge deployment.

Particle Filter (PF) is a sequential Monte Carlo localization technique that fuses predictions from a proposed state model with one or more sensor data. PF does not assume linear systems or gaussian distributions and it is a non-parametric approach meaning a mean and covariance matrix are not provided as inputs to represent the state distribution. Instead, PF draw random samples from a proposed distribution to represent the state space. The samples are called particles and represent hypotheses of the next sensor reading. Particles compose of a value (scalar or matrix) and a weight (scalar). Initially, PF builds the posterior density function (PDF) by selecting J number of random samples from the proposed transition model which can be assumed as a gaussian distribution. Normalized weights of each particle are then computed using the values from the proposed and actual models; a process called Importance Sampling. PF then iterates three steps: the Prediction step, Correction step and the Resampling of the particles. In the prediction step, a prediction of the next state for each particle is computed using the proposed transition model and counting in process noise. The correction step, involves computing the weights of the predicted states (hypotheses) via the observation model counting in measurement noise. The state estimate (belief) is then computed from the sum of hypotheses multiplied by their weights. Using Stochastic Universal Sampling approach, the particles are resampled by selecting J particles at once and then iterating the newly selected particles to replace weights with proportional frequencies in the sense that particles with large weights are replaced with multiple particles and particle with consistently small weights diminish after several iterations. The weights of particles are then set to be equal and are normalized. The new particles sample is dense at regions where particles had large weights and sparse in regions where particles had small weights [177]. In the adaptive Monte Carlo variation, the number of particles is reduced as the iterations advance. In [181], authors used PF to detect and isolate calibration faults (bias and scaling) by using the prediction state to evaluate whether the observation value exceeds a threshold that may render it a fault. If the threshold is surpassed then the algorithm evaluates a null hypothesis of the sensor being healthy and three alternative hypotheses to determine the type of error. Depending on the fault type the belief is compensated for bias or scaling [181]. To provide accurate estimates of complex distributions, PF require large number of particles. PF works well with low dimensional space (lower than five features). The performance quality of PF is dependent on identifying good transition (proposed) and sensor (observation) models [25,177].

Dempster-Shafer (D-F) inference theory is a generalization of Bayesian inference, which explicitly represents uncertainty and missing knowledge. It can fuse different types of sensors and does not require probability distributions as a priori information [25,175]. D-F has uncertainty management and inference mechanism analogous to the human reasoning process [200]. It obtains a degree of belief for one

Table 8
Tested Continuous Features Discretization Techniques.

Technique	Formula	Property	Results
Equal-width Binning	Given k splits where $c_i \in \{c_1, \ldots, c_k\}$, the split points are computed via: $c_i = min + i.\frac{max - min}{k+1}$ where max and min can either be the actual, semantical or contextual values of the feature.	Unsupervised, Top-down, Static	RMSE **0.47** MAE **0.33** MAPE **37.47%**
Equal-frequency Binning	Given k splits where $c_i \in \{c_1, \ldots, c_k\}$, the split points are computed via: $c_i = i.\frac{N}{k+1}$, where N is the number of instances.	Unsupervised, Top-down, Static	RMSE **0.44** MAE **0.3** MAPE **32.18%**
K-means	**Initialization**: 1) Randomly select k instances as initial centroids: $C = \{\mu_1, \mu_2, \ldots, \mu_k\}$. **Iteration**: 1) For each data point x_i, determine its cluster membership c_i via: $c_i = min_j \|\|x(i) - \mu_j\|\|^2$ where $\|\|x(i) - \mu_j\|\|$ is the Euclidean distance between the data point and the centroid. The data point is clustered with the nearest centroid. 2) For each cluster μ_j, recalculate its centroid via: $\mu_j = \frac{\sum_{i=1}^{m_j} x_i}{m_j}$, where m_j is the number of data- points in cluster j. The steps of the iteration phase are repeated until convergence.	Unsupervised, Stochastic, Static	RMSE **0.48** MAE **0.36** MAPE **32.14%**
SOM	1) Initialize the weights of $k$neurons of the one dimensional ANN. 2) Randomly select an instance $x$from the training dataset to update the network's $k$weights. 3) Compute the winning neuron via: $c^t(x) = min_{k \in 1,\ldots,K} \|\|x - m_k(t)\|\|^2$ 4) Update the weights of $k$neurons via: $m_k(t+1) = m_k(t) + \epsilon(t).h_{k.c^t(x)}(t).(x - m_k(t))$ where $\epsilon(t) = \epsilon_0 e^{\frac{-t}{\tau}}$ is the learning rate and $h_{k.c^t(x)}(t) = e^{\frac{-\|\|m_k(t) - c^t(x)\|\|^2}{2\sigma(t)^2}}$ is the topological neighborhood. Repeat steps 2–4 on all training data 5) Prune out neurons rarely updated, thus, the interval n: $\leq k$.	Unsupervised, Stochastic, Static	RMSE **0.23** MAE **0.17** MAPE **17.32%**

question from the probabilities of related questions and combine degree of beliefs of different sources [175]. The target system is represented by enumerating all possible mutually exclusive states of a target question in a frame of discernment Θ. A set Hypotheses (2^{Θ}) is then determined from a frame of discernment. Each hypothesis is assigned a probability representing its belief assignment using the mass function or the basic probability assignment m. With each belief assignment ranging between zero and one and the sum of all belief assignments is equal to one. A hypothesis may be a subset of states and has a confidence interval ranging from the sum of belief assignments of the hypothesis, to the plausibility of the hypothesis that includes evidence not ruling out the hypothesis. The confidence interval represents true belief in the hypothesis. Dempster-Shafer combining rule fuses the beliefs from multiple sensors via their mass functions (e.g., $m_i$and m_j) in addition to the sensors' plausibility of the hypothesis, which are used to compute the hypothesis with the highest probability while counting in uncertainty [25,175,200]. Authors of [64,85,86,211] use D-F inference to detect errors (noise, inaccuracies) in sensor data that would help identify faulty sensors or components. Existing work [42,104] further provide more extended surveys of sensor fusion techniques.

Distributing multivariate techniques for either preprocessing or processing to the edge requires fusing the features of data streams captured by the different sensors at the edge. Thus, there is a dependency between multivariate techniques and sensor fusion where the former is the dependent. In addition to facilitating the edge execution of multivariate techniques, sensor fusion distributed to the edge and performed prior to data transmission has several other benefits including reducing the data volume by including lineage meta-data to aggregated data once rather than duplicating shared meta-data to each sensor stream. The fusion can be as simple as a merge of multiple sensor data into one tuple that is based on the timestamp of when the data were generated. This includes merging sensors with different frequency by placing *Null* for features with no observations at a given timestamp. Many sensor fusion solutions have been proposed for IoT applications; including distributed versions of the Kalman Filter [148,182] and Particle Filter [29].

Feature engineering

The comprehensive definition of feature engineering covers feature reduction, transformation and synthesis. The ultimate goal is to formulate "the most appropriate features given the data, the model, and the task" [214]. We present the following feature engineering categories:

Discretization definition

is the process of transforming continuous features to discrete features with finite sets of adjacent intervals and associating each interval with a value (integer or nominal) thus summarizing the data while minimizing information loss [78,107,111,189].

Discretization Impact: Discretization has many benefits including adapting the data to algorithms that only process categorical data or that perform better with discretized variables [28,107,111,189]. In addition to reducing data volume, it also reduces the complexity of data and makes data patterns easier to understand [78]. Discretizing continuous feature could also improve learning or model training efficiency and accuracy [189].

Discretization Techniques: Discretization techniques can be categorized as unsupervised and supervised. In the later, the technique uses the response variable (class variable) in determining the intervals and attempt to maximize the interdependence between the response variable and the continuous predictor. Discretization techniques can also be categorized as top-down or bottom-up. The later, considers all available continuous values of the feature as potential splitting points and then merges points into bigger intervals repeating the step until the final intervals are determined via a condition or threshold. While the former starts with one interval or few splitting points and continues to segment the data towards obtaining optimal intervals [60,78,111,189]. Static discretization techniques discretize each continuous feature of the dataset in an isolated matter and are independent from the modeling technique thus they are performed as a preprocessing task. While dynamic discretization is part of the modeling and training phase and considers the interdependence between different continuous features that exist in the dataset. Lastly, global discretization techniques consume all available data instances (batch processing) while local techniques discretize on a sample of the data [60,189]. The performance of discretization techniques is measured by their ability to minimize the number of intervals created to avoid overfitting, whether the techniques consider the data distribution or not and by the performance of the modeling algorithm that consumes the discretized data [3,189]. In addition, supervised discretization techniques are evaluated based on their ability to maximize the interdependence between the class variable and the continuous predictor to be discretized [189]. In our survey we will focus on static techniques as we aim to decouple preprocessing from training, modeling and information extraction. We also focus on unsupervised discretization techniques as sensor data are typically unlabeled. Table 8 is an overview of the discretization techniques we have included in our experiments.

Binning is an unsupervised top-down discretization technique that has two variations equal-width and equal-frequency (quantile binning). Equal-width binning finds the minimum and maximum values of the feature and then divides the continuous data into k intervals. The produced bins contain variable number of instances with some bins may ending up empty. Equal width binning is sensitive to outliers and does not consider or reflect the original distribution of the feature [45,191]. Equal-frequency binning counts the number of instances of the feature and then creates intervals with approximately equal number of instances [3,45]. The technique considers the data distribution of the feature by adaptively positioning the bins based on the quantiles of the distribution to obtain for example quartile bins (if $k = 4$) or decile bins (if $k = 10$), which means each interval contain approximately 25% or 10% of the data respectively [4,28,78,107]. With equal-frequency binning, the range of some of the bins produced can be skewed by outliers or non-gaussian distributions placing unrelated instances from vastly different classes in the same bin [50,103].

Both binning variations require a predefined k value and associate each bin created with a value which could be nominal or integer (e.g., the mean or median value of the instances it contains). Equal-frequency and equal-width binning are efficient and simple techniques; however, their discretization is likely to hide patterns in the data, not produce semantically meaningful intervals and fail to learn certain concepts from the data which is the result of ignoring the class of the training instances [103,191]. Gaussian Approximation or Histogram bin count [28] is a binning technique that uses the standard deviation σ and mean μ in determining the splitting points k predefined by the user. Its space and time complexities grow linearly with the number of instances. The mathematical equations of the binning techniques can be found in Table 8.

Clustering is a popular discretization approach for continuous features that is also unsupervised and requires a predefined k value. Clustering can also be applied as a multivariate discretization approach that considers all attributes in the dataset when determining the intervals of a particular continuous attribute [60]. K-means clustering method consist of two main steps initialization and iteration. In the former, k values are randomly selected as initial centroids of k number of clusters into which the continuous feature is divided. Data points are then assigned to the nearest cluster center using a distance metric (e.g., Euclidean). The algorithm finds a compilation of clusters and member data points with minimized total sum of distances between the clusters' centers and their member data points. The iteration step involves recomputing the centroids of each cluster via the current member data points followed by reassigning the data points to new clusters if their distance with the centroid is shorter otherwise they are kept at the current cluster. K-means is a hard clustering technique, meaning each data point is assigned to one cluster only. The iteration step repeats until convergence, that is, there are no longer re-assignments of data points. The generated clusters represent the intervals of the discretized continuous feature and the value of an instance is determined by its cluster membership [4,72,191]. K-mean clustering produces intervals that reflect the distribution of the original data. A global optimum is, however, not guaranteed and the quality of the discretization depends on the k value and on the random seed of the initialization step [191]. In addition, the clustering algorithm is also sensitive to outliers and performs poorly when clusters are of different shapes, sizes and densities [72].

Shared Nearest Neighbor (SNN) can be used for discretization [72], which is more resilient to outliers and with improved performance on high dimensional data producing arbitrarily shaped clusters with different densities when applicable [51]. The distance and density measures used in SNN are based on shared neighbors and on similarities with neighbors, respectively. The introduction of representative or core points in SNN allowed for irregular shaped clusters to be formed [51]. Similar to K-means, it requires a predefined k value which controls the granularity of clusters. SNN has a quadratic time complexity.

In [191], authors proposed an unsupervised discretization technique based on the artificial neural network Self-Organizing Map (SOM). The algorithm does not require the exact number of clusters k as a-priori but only requires a fixed maximum number of clusters m. The discretization technique constructs a one dimensional SOM with m neurons. The weight of the neurons are learned during training using unlabeled training data while preserving the feature's original data distribution. The training process start with randomly initializing the network weights. A data point is then randomly selected to initiate the process of learning and updating the neurons' weights to model the input data. The Euclidean distance is used to determine the neuron closest to the current data point ($x(t)$), i.e., the neuron with the smallest Euclidean distance, which is labeled as the Best Matching Unit (BMU) or winning neuron ($c^t(x)$). The weight of the selected winning neuron is then used to update its weight and the weights of other neurons in the map space. SOM neurons are similar to the centroids of K-means; however, the neurons are connected with each other in the sense that their updates impact the weights of other neurons but in a reduced scale. This is done using a weight update formula with a learning rate that controls the significance of the weight update and a topological neighborhood term which either magnify the learning rate of the weight update for $c^t(x)$ or scale down the learning rate of the weight update for the other neurons based on their distance from $c^t(x)$. The Euclidean distance metric is used to compute the distance between the current weights of the neurons $m_k(t)$ and the current of weight of $c^t(x)$. The magnitude of the weight updates decreases over time to stabilize the learning process and the scope of the neighborhood relations. The idea is to find the neuron closest to the data point and to update the neuron's position in the map space towards the data point in addition to making smaller shifts of other neurons towards the

same direction depending on their distance from the data point and $c^t(x)$. The process is repeated for each instance in the training data until convergence (the sum Euclidean distance between data points of each cluster is minimized) or a stopping condition is fulfilled (the changes in weights are insignificant or less than a threshold) [108,191]. Neurons that are rarely updated are pruned and discretization classes produced can be less than m [191]. SOM is a stochastic algorithm that can produce different discretization intervals from the same training data. It can take many iterations before it converges. SOM is an unsupervised learning algorithm; however, it is possible to incorporate the class variable as input. Yet, the interdependence between the predictors and the class variable would still not be the focal point of the learning process. The equations of the SOM discretization technique can be found in Table 8.

In our survey we focus on unsupervised regression problems as sensor data are naturally unlabeled and are often continuous. Having said that, we also cover supervised techniques in this section as discretization is mostly intended for classification problems. Unsupervised discretization techniques do not leverage class labels, when present, and thus they are likely to result in loss of classification information and produce intervals with instances that are strongly associated with other classes (i.e., interval has no dominant class) [45]. Chi-Merge [103] is a supervised, bottom-up and univariate discretization technique that determines the intervals based on the class variable with the objective of producing a concise summarization of the continuous feature with intra-interval uniformity and inter-interval difference. The technique starts with sorting the values and placing each instance in its own interval. Chi-Merge uses Chi-Squared test χ^2 to determine whether adjacent intervals are different or similar enough to merge. The χ^2 value is computed for each pair of adjacent intervals and pairs with the lowest χ^2 value are merged. This means that interval pairs with significantly different class frequencies (the interval determines the class) are not merged while pairs where the class is independent from the intervals (similar class frequencies) are merged [103]. The step is iterated until a termination condition is fulfilled. Despite its ability to preserve the feature's distribution, Chi-Merge is sensitive to the χ^2 threshold value and overestimate the degree of difference if the expected frequency is small. The algorithm has a time complexity of $O(n^2)$ and may construct too many intervals [14].

Khiops [14] is also a supervised bottom-up method that is based on χ^2 statistics. Its objective is to minimize the confidence level of the null hypothesis that the class variable and the intervals are independent. The higher the χ^2 values is the lower the confidence level is of the null hypothesis, which means the merge of adjacent intervals achieves interdependence between the interval and the class variable. The process is repeated until a minimum frequency within each interval is respected and the confidence level can no longer be minimized. Khiops technique controls overfitting heuristically by constraining the frequency of instances within each interval to be greater than the square root of the sample size [14].

Class-Attribute Interdependence Maximization (CAIM) is a supervised univariate statistical technique that maximizes the class-features interdependence while minimizing the number of discrete intervals generated. The top-down approach selects the number of intervals without a predefined k value from the user. The technique then classifies the continuous values of the feature into the k intervals. This means that the algorithm starts with one interval $D = [d_0, d_n]$ containing all values with a global CAIM value of zero. A new splitting point is added to D and from the class variable and the feature to be discretized, a quanta matrix is defined, which includes the number of instances for each class value within each interval. The CAIM value is then computed using the sum of the maximum (max_r) count of instances that have the class value i within interval $[d_{r-1}, d_r]$ of all intervals. max_r is divided by the number of instances within the interval $[d_{r-1}, d_r]$ to account for the negative impact of the values belonging to other classes and to scale the maximum. The splitting point is accepted if the computed CAIM value is greater than the global CAIM. This makes CAIM a greedy search algorithm that approximate the optimal value by finding a local maximum which may not necessarily be the global maximum. The greater the value of CAIM is, the higher the interdependence between the class variable and the intervals. CAIM favors a discretization scheme where the majority of instances within an interval belonging to the same class value (intervals with dominant class value). CAIM is a fast algorithm that automatically finds the smallest number of intervals, however the number of intervals is typically equal to the number of class values ($k = r$).

Class-Attribute Contingency Coefficient (CACC) [189] has the same steps as CAIM, however, the quality of the intervals in each iteration is measured using the CACC value. The contingency coefficient is used to measure the interdependence between the predictor to be discretized and the class variable. Instead of only considering the class with the maximum number of instances within an interval, CACC considers the frequency (number of instances) of all classes for each interval divided by the log of the total number of interval ($log(n)$) within the iteration. The $log(n)$ division avoids overfitting and accelerates the discretization process. CACC is a greedy algorithm that keeps track of a global CACC value and terminates if the local CACC is less than the global value and the number of intervals is equal or greater than the number of classes. CAAC avoids overfitting and minimizes loss information by considering the distribution of the data.

Another technique similar to CIAM and CACC is Ameva [68] which maximizes the contingency coefficient using χ^2 statistics with the same objective of maximizing interdependence between the predictors and class variable and minimizing the intervals to avoid overfitting. For supervised discretization, we have mainly presented the statistical based techniques. The reader can refer to [60] for a comprehensive survey of discretization techniques including information based and rough-sets based techniques.

Distributing discretization techniques to the edge is correlated with EdgeAI; particularly with the distribution of classification machine learning models. Authors of [96] empirically demonstrated the feasibility of distributing and extended SOM algorithm using Apache Spark. The proposed distributed GSOM algorithm discretized a large dataset of unlabeled data into clusters based on a common feature value (human activity) while excluding temporality. Nyström-Persson et al [145] also leveraged Apache Spark to develop a distributed binning and k-mer counting tool, called *Discount*, for metagenomics data. Using *Discount*, the authors yielded evenly distributed and small sized bins, which improved the efficiency of memory and storage usage and accelerated overall processing speed. In Applications and use-cases where temporality is important to encode in the discretization of continuous data, then time-series discretization techniques are a better fit. Techniques that are applicable on time-series include Symbolic Aggregate Approximation (SAX) [119], MultIple Normal distributIONS (MINIONS) [58] and the temporal discretization technique of distributed systems proposed by D'Andrea et al [33]. In [152], authors proposed k-Shape, which is a partitional time-series clustering algorithm that iteratively produce homogeneous and well-defined clusters using the cross-correlation measure Shape-based distance (SBD). Their empirical results highlighted that in addition to preserving the shapes of time-series, k-Shape also outperformed other tested techniques in terms of yielded accuracy. Lastly, we wanted to highlight that further empirical analysis is required to evaluate the feasibility and impact of distributing the aforementioned time-series techniques to the edge (e.g., deployed in IoT devices).

Empirical Results: In our experiments, discretization was performed after the last preprocessing category, which is feature selection, of our standard preprocessing plan (see the DAG structure presented in Fig. 1). Several discretization techniques were tested on transforming continuous features to their categorical representation. K-means and SOM produced clusters while the rest of the tested techniques produced bins. Both the clustering techniques k-means and SOM were performed as multivariate discretization techniques using the feature to be discretized and the response variable (CO) as inputs. Our experiments

Table 9
Tested Feature Selection Techniques.

Technique	Formula	Property	Results
F-Test	For each feature A_i of the dataset, find: $F = \frac{MS_{between}}{MS_{within}}$ where MS is the abbreviation of Mean Squares. Expanding the numerator and denominator gives: $F = \frac{\sum_{i=1}^{K} n_i(\overline{Y_I} - \overline{Y})^2/(K-1)}{\sum_{i=1}^{K}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y_i})^2/(N-K)}$	Feature Selection, Supervised	RMSE **0.33** MAE **0.24** MAPE **27.39%**
Laplacian	1) For feature A_i, build the weight graph $G$with m nodes. 2) For each instance, find its neighbors and their distance. 3) Neighbors are connected with edges. 4) A weight matrix is constructed from $G$and assigned the following weights: for nearest neighbors: $S_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}}$, otherwise: $S_{ij} = 0$. 5) Given that $f_r = [f_{r1}, f_{r2}, \ldots, f_{rm}]^T$ contain instance values of attributeA_i, the diagonal matrix $D$is found via: $D(i,i) = \sum_{j=1}^{m} S_{ij}$, where $S$is the similarity matrix. $D$is multiplied by the identity matrix for $m \times m$ dimensions. 6) The Laplacian matrix is obtained via: $L = S - D$ 7) Instances of f_r are normalized using zero-centering: $\overline{f_r} = f_r - \frac{f_r^T D1}{1^T D1}1$ 8) The Laplacian score is computed via: $L_r = \frac{\overline{f_r}^T L \overline{f_r}}{\overline{f_r}^T D \overline{f_r}}$	Feature Selection, Unsupervised, & Supervised	RMSE **0.46** MAE **0.31** MAPE **29.67%**
RReliefF	1) Select instance R_i randomly. 2) Select the $k$nearest neighbor I_j ofR_i. 3) Find $P_{diffA} = P(differentA \mid nearest)$. $P_{diffC} = P(differentC \mid nearest)$. $P_{diffC \mid diffA} = P(differentC \mid differentA \& nearest)$. **A** is the value set ofA_i, **C** is the value set of the response variable, and **nearest** is the set of Nearest neighbors of R_i 4) Find the quality estimation: $W[A] = \frac{P_{diffC \mid diffA} P_{diffA}}{P_{diffC}} - \frac{1 - P_{diffC \mid diffA} P_{diffA}}{1 - P_{diffC}}$ The probability is modeled using relative distance: $d(i,j) = \frac{d_1(i,j)}{\sum_{l=1}^{K} d_1(i,l)}$ where $d_1(i,j) = e^{-(\frac{Rank(R_i, I_j)}{\sigma})}$. The exponential distance measure is used to significantly decrease the influence of neighbors with greater distance from R_i. Ranks are used instead of distance to diminish the influence of scales.	Feature Selection, Supervised	RMSE **0.34** MAE **0.22** MAPE **24.43%**
NCA	Given the labeled dataset: $S = \{(x_i, y_i), i = 1, 2, \ldots, n\}$ 1) Initialize the weight vector w, assigning all features p the same weight. 2) Use stochastic nearest neighbor to maximize the LOO regression accuracy: $p_{ij} = \frac{k(d_w(x_i, x_j))}{\sum_{j=1, j \neq i}^{n} k(d_w(x_i, x_j))}$, where $d_w(x_i, x_j) = \sum_{r=1}^{p} w_r^2 \lvert x_{ir} - x_{jr} \rvert$ and $k = e^{-\frac{d_w(x_i, x_j)}{\sigma}}$ 3) Evaluate the average loss value of $l(y_i, \widehat{y_i})$: $l_i = \frac{1}{n}\sum_{j=1, j \neq i}^{n} p_{ij} l(y_i, \widehat{y_i})$ 4) Subtract the regularization term from the loss function: $f(w) = \sum_{i=1}^{n} l_i - \lambda \sum_{r=1}^{p} w_r$ 5) Find the weight vector of features using: $f'(w) = 2(\frac{1}{\sigma}\sum_{i}^{n}(p_i \sum_{j \neq i} p_{ij} \lvert x_{ir} - x_{jr} \rvert - l_i \lvert x_{ir} - x_{jr} \rvert) - \lambda) w_r$	Feature Selection, Stochastic, Supervised	RMSE **0.35** MAE **0.26** MAPE **27.94%**

excluded temporality, however, it will be included as part of our future work. The histogram diagrams and plots in Fig. 12 present outcomes of each tested discretization technique. For each technique evaluation, we performed several tests or attempts (a minimum of three). The clusters produced by SOM in each test were similar while k-means reflected inconsistency with significantly varying cluster regions in each test performed. Using the discretized input data, generated using the different tested discretization techniques, we trained LSTM models for the purpose of evaluating the impact of discretization techniques on the performance and quality of AI models and networks. We used the discretized predictors (mapped into bins and clusters numerical representations) and kept the response variable continuous for LSTM to train correctly and to evaluate the performance accurately. We evaluated both univariate and multivariate K-means discretization, however, the results of the LSTM network were of low accuracy for the univariate version and thus, based on our experiments, we recommend using multivariate k-means over univariate k-means discretization. Our empirical results indicate a generally deteriorating prediction accuracy of the trained LSTM networks, which is an expected outcome due to the loss of information caused by discretization. An exception to this trend is

SOM, which produced one of the most accurate LSTM networks of all the experiments completed. The empirical results of the last test performed is presented in Fig. 13.

Feature reduction definition

As data volume increases, managing data in an effective and efficient matter becomes more challenging [117]. Reducing data volume can be achieved by either sampling the data or performing dimensionality reduction for high dimensional data. In this survey, we focus on dimensionality reduction, which involves the reduction of the number of features while capturing the essence of the data [143].

Feature Reduction Impact: High dimensional data may suffer from various problem including:

- Curse of Dimensionality: As the number of features increases, the probability of data points being within neighboring distance is low. In other words, the data become sparser and thus modeling high dimensional data accurately requires a large number of samples [117].
- Overfitting: when the number of features is similar or equal to the number of data samples, modeling algorithms struggle to produce generalized models, which mean they underperform with test data and new data [117].
- High Resource Consumption: High dimensional data results in increased memory requirements and higher computational and energy costs [117].

Feature reduction has been proven in theory and practice to improve learning efficiency of modeling algorithms and produce models with higher accuracy [13,21,74,117,125]. Dimensionality reduction also improves the efficiency of resource consumption, reduces computational and storage demands and improves sustainability with less energy costs. Effective dimensionality reduction could produce less complex and more comprehensible models [21,98,117]. There are two main categories of dimensionality reduction: Feature Selection (FS) and Feature Extraction (FE).

Feature Selection Definition: is the process of obtaining a subset of the original data based on selection criterion that retains the most relevant features and removes irrelevant variables [21,117]. FS maintains the physical meaning of the original feature space thus producing models with better readability and interoperability [117]. Due to the computational complexity of evaluating subsets of m features, most proposed solutions are heuristic methods that maximize relevance; with some techniques also minimizing redundancy. FS categories are based on:

- The training data: supervised, unsupervised and semi-supervised techniques.
- The relationship with the modeling algorithm: filters, wrappers, embedded and hybrid.
- The subset search mechanism: forward increase, backward deletion, random and hybrid.
- The subset evaluation criterion: correlation-based, statistical-based, similarity-based, distance-based and information theoretical-based.

Feature Selection Techniques: Wrappers produce a high performing subset tuned for a specific model. It is an expensive technique as each possible subset is iteratively used to train the model and the error rate of the model is then used as the score of the subset [4,59]. Greedy search strategies are feasible for wrapper methods [98]. Examples of wrapper methods include recursive feature elimination [34], Particle Swarm Optimization (PSO) [204] and Genetic Wrappers [90]. Filtering is cheaper to compute; however, it produces a general feature subset resulting in lower prediction performance. As we are decoupling preprocessing from processing (e.g., modeling algorithms), we will restrict our survey to filters and hybrid feature selection techniques. Our categorization of FS will also make a distinction between cloud execution and edge deployment. Table 9 provides an overview of the tested FS techniques and their performance based our empirical analysis.

Filters are feature selection techniques that are independent of the inference and the modeling algorithms when evaluating feature subsets and selecting features. Instead, filters use evaluation criterion to measure the correlation, relevance and redundancies of features. Unlike wrapper methods, the selected features are not optimized to particular algorithm or model [21,98,117]. Features correlation (the amount of new information introduced by a feature) and mutual information (the amount of information obtained about one feature from another feature) are some of the measures used for filtering. Features that fall below a threshold for a measure are filtered out [4,21,59]. There are several search strategies to find the subset features such as greedy forward selection, greedy backward elimination, simulated annealing, race search and exhaustive search [21,117].

Removing features with low variance is a univariate statistical filter that can be used for unsupervised problems. For numerical variables, the variance is computed using the mean to determine whether the variables' variances are below a predetermined threshold or not. The objective is to remove variables with zero or close to zero variances as they are less informative or with less predictive power [117]. Correlation based Feature Selection (CFS) [76] is a multivariate statistical-based filter that eliminates irrelevant and redundant features using a heuristic evaluation function. CFS measures the correlation between the features and the response variable (feature-to-class) and the correlation between the features (feature-to-feature). The algorithm selects a subset with features that are highly correlated with the class but uncorrelated with each other. The mathematical formula of CFS subset evaluation function can be found in Table 9. The numerator of the function indicates how predictive the subset is of the response variable. The denominator indicates the subsets' level of redundancy. The authors used forward selection, backward elimination and best first as search strategies to find candidate subsets, however, other search strategies can also be used for CFS. The three methods are prevented from searching the entire feature subset search space via stopping criteria with the objective of choosing the subset with highest evaluation merit found during the search. CFS is applicable for numerical and categorical data and for classification and regression problems, however to standardize the process, discretization [55] is applied on numerical continuous data as a preprocessing step. CFS fails to identify locally predictive features in small areas of instance space [76].

F-test is a univariate statistical hypothesis test that evaluates a predictor's ability to separate the classes (discretized values) of the response variable. It compares the mean and variance of the predictor's values grouped by the values of the response variable to test if the different groups come from normal distributions with the same mean (null hypothesis) or from populations with completely different means (alternative hypothesis). The F-test equation can be found in Table 9. The numerator calculates the variance between the classes while the denominator calculates the variance of each class distribution. The higher the F-test score is the more predictive the feature is of the response variable as it separates well the classes or discrete values of the response variable [174]. To use the F-test feature selection, it is assumed that the values of each variable are independent and that the residual errors within each class are samples from normal distributions [113].

Other statistical-based filters include Chi-Square Score [126] and T-score [35] however their use is limited to classification problems.

Relief algorithms are a series of extended similarity-based filters with the original Relief algorithm introduced in 1992 [106]. The algorithm can estimate the importance of features even with strong dependencies between them. The Relief algorithm and the extensions that followed, start by randomly selecting an instance and finding its k (k=1 for Relief) nearest neighbor(s) with the same class and k nearest neighbor(s) with the other class(es). Instances with same response variable value (class) are compared to determine whether their values for attribute A_i is

Table 10
Tested Feature Extraction Techniques.

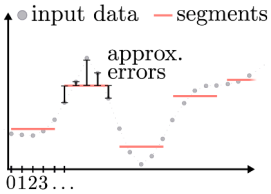
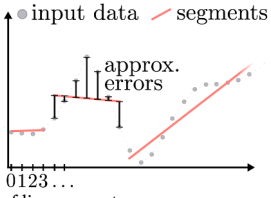
Technique	Formula	Property	Results
PCA	1) Compute the average row of the data matrix X: $\hat{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ 2) Using $\hat{x}$compute the average matrix: $\hat{X} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_n \hat{x}$ 3) Center X using the average matrix: $B = X - \hat{X}$ 4) Perform SVD on the centered data: $B = U\Sigma V^t$where $U$is an orthogonal vector with information about the column space, $V$is an orthogonal vector with information about the row spaces and $\Sigma$is a diagonal matrix with values indicating the amount of variance. 5) Given that the principle components decomposition of B is $T = BV$, then substituting B with $U\Sigma V^T$ from step 4 gives: $T = U\Sigma V^T V$. Given that V is orthogonal, $V^T V = 1$, then $T = U\Sigma$	Feature Extraction, Unsupervised	RMSE **0.5** MAE **0.36** MAPE **34.13%**
LDA	1) Compute within-class scatter matrix via: $S_w = \sum_{j=1}^{C}\sum_{i=1}^{N_j}(x_i^j - \mu_j)(x_i^j - \mu_j)^T$ where x_i^j is the i^{th}sample with class value j, μ_j is the mean of the class$j$, $C$is the number of classes, and N_j is the number of samples with class value j. 2) Compute the between-class scatter matrix via: $S_b = \sum_{j=1}^{C}(\mu_j - \mu)(\mu_j - \mu)^T$, μ is the mean of all classes. 3) Maximize the ratio: $LDA = \frac{det\|S_b\|}{det\|D_w\|}$, where det is the determinant of the matrix.	Feature Extraction, Supervised	RMSE **0.23** MAE **0.16** MAPE **16.37%**

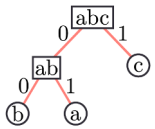
different or not. Having different values means that the attribute is separating two instances from the same class and thus the quality estimation of the attribute is decreased. On the other hand, instances with different classes and different attribute values means that A_i is successful in separating two instances with different classes and thus its quality estimation is increased. The RReliefF [164] extension was adapted for regression problems where the response variable is continuous. In the absence of discrete classes, the algorithm uses a probabilistic approach using relative distance to determine whether instances have different response values or not. Table 9 includes the equations of the RReliefF filter. The algorithm starts by randomly selecting an instance R_i and finding k nearest neighbors with I_j being a neighbor instance. The relative distance between the response values of the instances is used to model P_{diffA}, which is the probability of nearest instances having different values of attribute A_i, P_{diffC}, which is the probability of nearest instances having different response values and $P_{diffA|diffC}$, which is the probability of the nearest instances having different response values given that they have different values of A_i. The quality estimation of A_i is then determined using these probabilities and features with highest weights are selected. RReliefF computational complexity is an improvement compared to FS techniques that require subset search followed by evaluation. The iterative process of finding k nearest neighbors to estimate the quality of A attributes given n instances and using m iterations (user defined) has a computational complexity of $\mathcal{O}(m.n.A)$ [165]. RReliefF is sensitive to the curse-of-dimensionality and its performance deteriorates as the number of features increases. The algorithm may also produce a selected features subset with redundancies [190].

The Laplacian score [84] is a similarity-based univariate filter that can be performed on both supervised and unsupervised problems. It is based on the observation that instances from the same class are often found, in real-world datasets, close to each other. The importance or quality of features is determined based on their ability to preserve locality and based on their variance (higher variance being favorable). The algorithm starts by constructing the nearest neighbor graph G of feature A_i that consist of m nodes representing m instances. Two nodes (i and j) are connected with an edge if instances x_i and x_j have a nearest neighbor relationship in either direction. From G, the weight matrix S is created, which consists of the weights of edges between the different nodes. S models the local structure (relationships) of the data space. If two nodes x_i and x_j are connected, then the weight is computed via an exponential function using a distance measure. Otherwise, the weight is set to zero. The Laplacian matrix L is then obtained by subtracting the diagonal matrix D from S. The Laplacian score (L_r) of the feature is calculated using L, D and using centered (normalized) data instances $\bar{f}_r$ of A_i [84]. We present the full Laplacian algorithm in Table 9. In the L_r equation, the numerator measures the distance (similarity) between instances and the smaller the sum of distances the more preserving it is to predefined G graph structure (i.e., higher locality preserving power). The denominator measures the features variance and the higher it is the more predictive is the feature [84,117]. Maintaining undirected graph for each attribute, especially with high dimensional and large datasets renders the filter expensive especially with batch processing.

Neighborhood Component Analysis (NCA) Feature Selection [67, 209] is a filter that has two variations: NCA for classification and NCA for regression. The latter is adapted for continuous response variables. NCA is a supervised learning non-parametric algorithm that makes no assumptions about data distribution and problem linearity. It is also insensitive to large number of irrelevant features. The intuition is that important features are the ones that best cluster related instances and separate the different clusters. The objective is to find a feature subset with which the classification or regression model yields the most accurate results. This is done by computing the features weight vector using the stochastic nearest neighbor algorithm to optimize the performance of the Leave-One-Out (LOO) classification or regression on the training data. In addition, a regularization term is used to derive the weights of irrelevant features closer to zero and thus facilitate feature selection. The algorithm starts by randomly selecting a reference point x_j for all the instances x_i of the labeled dataset S^{-i} (excluding the instance x_i). The smaller the distance between x_i and x_j the greater the probability of selecting x_j nearest neighbor as the reference point. To model this relationship, an exponential kernel function is used to assign a higher probability to smaller distances. In both [67,209], the Euclidean distance is used to determine nearest neighbors. After selecting a reference point, its label y_j is assigned to instance x_i. Choosing nearest neighbors increase the probability of correctly estimating the original response value y_i and thus maximizing LOO classification or regression accuracy. A loss function $l(y_i, y_j)$ such as Mean Absolute Deviation (MAD) is used to measure the difference between the estimated and the original response values. The average LOO estimation accuracy is determined from the sum of the selected probabilities of stochastic nearest neighbors and the loss function value of all instances. Authors of [209], introduced $f(w)$, which consist of a regularization term together with the LOO estimation accuracy function. Feature weights are determined from the derivative of $f(w)$. NCA is sensitive to feature scales and thus normalization as a

Table 11
Highlighted sensor data compression techniques and their various approaches.

Technique	Property	Example & Approach	
PAA [101]	lossy, single-pass, numerical time-series	 mapping an input series to a series of constant line segments of length 5	The procedure to construct a PAA using the approach from [101] proceeds as follows, with the *segment length* m : 1) Split input data into n univariate series 2) Replace series values x_i, x_{i+1}, …, x_{i+m-1} with their mean (on the left: $m = 5$). 3) Proceed with values x_{i+m}, x_{i+m+1}, …, x_{i+2m-1}, etc. This scheme cannot give guarantees about the maximum approximation error. The compression ratio is $\approx 1/m$. The output is a series of means.
PLA [82]	lossy, single-pass, numerical data	 mapping an input series to a series of line segments	The procedure to construct a PLA as in [82] proceeds as follows: 1) Split input data into n univariate series 2) Assign an increasing counter to each series 3) Produce best-fit line segments for each series, using counter as dependent variable. The approximation error (see Figure on left) is bounded to a user-defined value per series. The lines are constructed greedily, allowing single-pass data processing. Splitting the input into univariate series allows for independent compression & decompression. The output is a series of triples: ⟨segment length, line slope, line intercept⟩.

Gorilla Compression [155]	lossless, single-pass, 64-bit doubles	value	XOR w/ prev.	The Gorilla scheme compresses timestamps and values differently. Here, we use the value compression scheme only.
		15.5	∅	1) Split input data into $n$univariate series of 64-bit doubles. 2) In each series, store first value as-is. 3) For all other values, compute the XOR with the previous value (see Table on left for a series of length 4). 4) If value is identical to previous, store '0' (1 bit). 5) Else, store '# of leading zeros' (5 bit)' + '# of meaningful bits' (6 bit) + 'meaningful bits [in table: black]' As in the PLA scheme above, splitting the input data allows independent compression and decompression of multivariate input data. The output is a bitstream.
		14.0625	0×00**32**00…	
		14.0625	0×000000…	
		8.625	0x**d6**0000…	
		XOR of doubles with prev- ous value. Significant bytes in bold.		
ZIP [39]	lossless, multi-pass, arbitrary byte strings	 Huffmann tree en- coding for the strings "a", "b", "c".		The deflate algorithm processes input data as a byte stream and consists of two main stages: 1) Detect duplicate strings and replace them with a pointer *(a variable amount ofprevious data is checked for duplicates tobound computational complexity).* 2) Construct a Huffmann tree (see left), encoding most frequent strings to shortest unique bit sequences (*see example on left:* "b" = 00, "a" = 01, "c" = 1). Stage (1) largely defines the compression ratio through the window of data that is checked for duplicates of a given string.

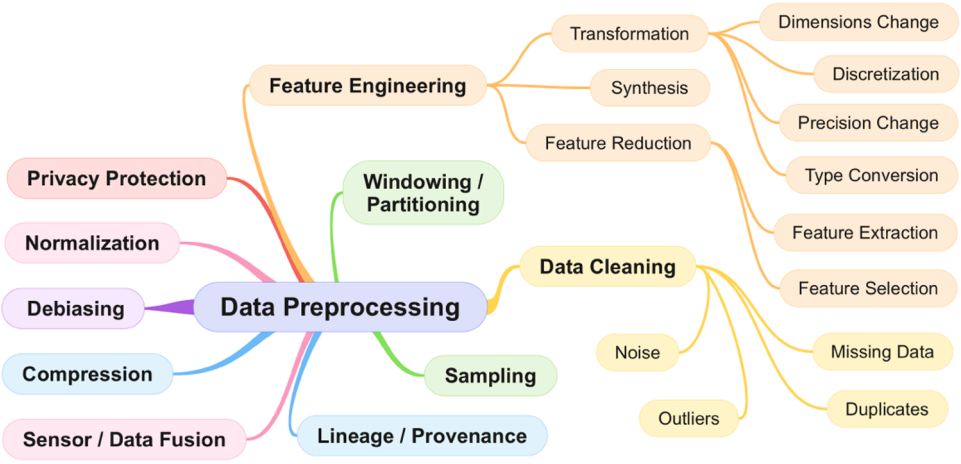


Fig. 4. Data Preprocessing Taxonomy. The Colors Distinguish the Different Categories (Sub-categories Inclusive) for Improved Readability.

preprocessing step is required when variables have significantly different scales. It is also susceptible to the local minimum problem when minimizing $f(w)$. Table 10 includes the regression equation of NCA.

Minimum Redundancy and Maximum Relevance (mRmR) [156] is an information based feature selection methods that uses mutual information to measure feature relevance and redundancy. The technique selects relevant features while controlling redundancy in the subset. A recent extension of mRmR is introduced in [213], in which Randomized Dependence Coefficient (RDC) is used as the redundancy measure and Random-Forest Correlation Quotient (RFCQ) is used as the relevance measure. RDC addresses non-linear problems while the idea behind using RFCQ is to leverage a relevance measure that is related to the processing model. Information-Theoretical-Based methods including mRmR techniques, Conditional Infomax Feature Extraction (CIFE) [118] and Joint Mutual Information (JMI) [206] are only applicable on supervised classification problems.

Genetic Algorithm (GA) [114] is also a popular heuristic approach used for feature selection which is a search function that incorporates an evaluation function (fitness function). GA belongs to a larger class of evolutionary algorithms based on the biological process natural selection. The subset search problem of FS has an exponential search space making it an NP-hard problem. This makes GA a natural optimization strategy for high dimensional datasets as it efficiently finds approximate solutions that are more compact yet more informative compared to traditional solutions. The fitness function can be a linear classifier (using least square as the error function) [83], inconsistency rate filter [112], support vector machine [132] or random forests [216]. GA can be used for regression and classification problems [114]. The general algorithm starts with randomly selecting the populations (chromosomes) of the first generation. Each population is defined as the genetic representation (binary) of the features (genes) that are included (set as 1) and excluded (set as 0) in the population. The evaluation (fitness) function determines the weight of each population based on their performance in predicting the response variable (supervised) or based on their variance (unsupervised). The next generation is constructed by inheriting elite populations, those with large weights, from the previous generation and by crossing-over randomly selected population to produce new populations (offspring). The cross-over function has several variation including choosing k number of genes randomly and swapping them between each pair. Another approach is to randomly select a splicing point for the swap. The next generation undergoes one more process called mutation, in which genes in each population are randomly selected and flipped. Mutations simulate big jumps in the search space to avoid local optimum and deadlocks [114]. GA algorithms are computationally expensive and require careful tuning of the different parameters (e.g., crossover and mutation rates) thus these limitations must be considered in the context of IoT.

In [88], authors proposed a framework that distributes five feature selection techniques to edge components using AURA neural network and Apache Hadoop. The distributed feature selection techniques calculate the weights of the different features at different edge devices. The framework collects the calculated weights from edge components and aggregate them into one result. Their empirical analysis using their framework demonstrated the feasibility of distributing feature selection techniques which can be extended to IoT. Authors of [140] recommended distributing feature selection techniques especially with large datasets. Their experimental results demonstrate improvements in runtime and in the quality and accuracy of AI solutions trained on the data. Their experimental analysis covered many techniques including CFS and ReliefF feature selection techniques. In [38], authors distributed feature selection techniques, including Chi-square, using Apache Spark with the objective of selecting a subset of sensor streams. Their empirical analysis demonstrated performance efficiency of the distributed techniques, however, they highlighted the need to distribute other preprocessing categories (e.g., normalization) to further improve the performance of feature selection techniques.

Empirical Results: Our experimental results confirm that FS techniques are more effective and yield positive results when the dataset is high dimensional. This was not the case with our dataset (derived from the AirQuality dataset) as we only had 15 features. For all of our feature selection experiments we selected the top five high scoring features. The LSTM model performed best when using all features as opposed to just using five features that had highest importance scores determined by the tested FS techniques. This is an expected outcome due to the information loss caused by FS. Having said that, FS could still prevent overfitting [117] and improve resource (computation, storage and energy) usage efficiency. Comparing the results of the different FS techniques we tested, RReliefF yielded the most accurate predictions while the Laplacian score performed the worst (highest error scores), which can be attributed to the fact that the Laplacian score is an unsupervised technique.

As for the rest of our survey's experiments, we combined the results of the four techniques tested selecting the five most commonly selected features to be used in all the survey's experiments across the different preprocessing categories. Benzene was the only feature that was selected by all four techniques tested. Hydrocarbons and NO^x where selected by at least three techniques. The results of our experiments are presented in Figs. 14, 15.

Feature Extraction Definition: Feature extraction (FE) is the projection of a high dimensional feature space to a low dimensional feature space with strong pattern recognition ability [21,117].

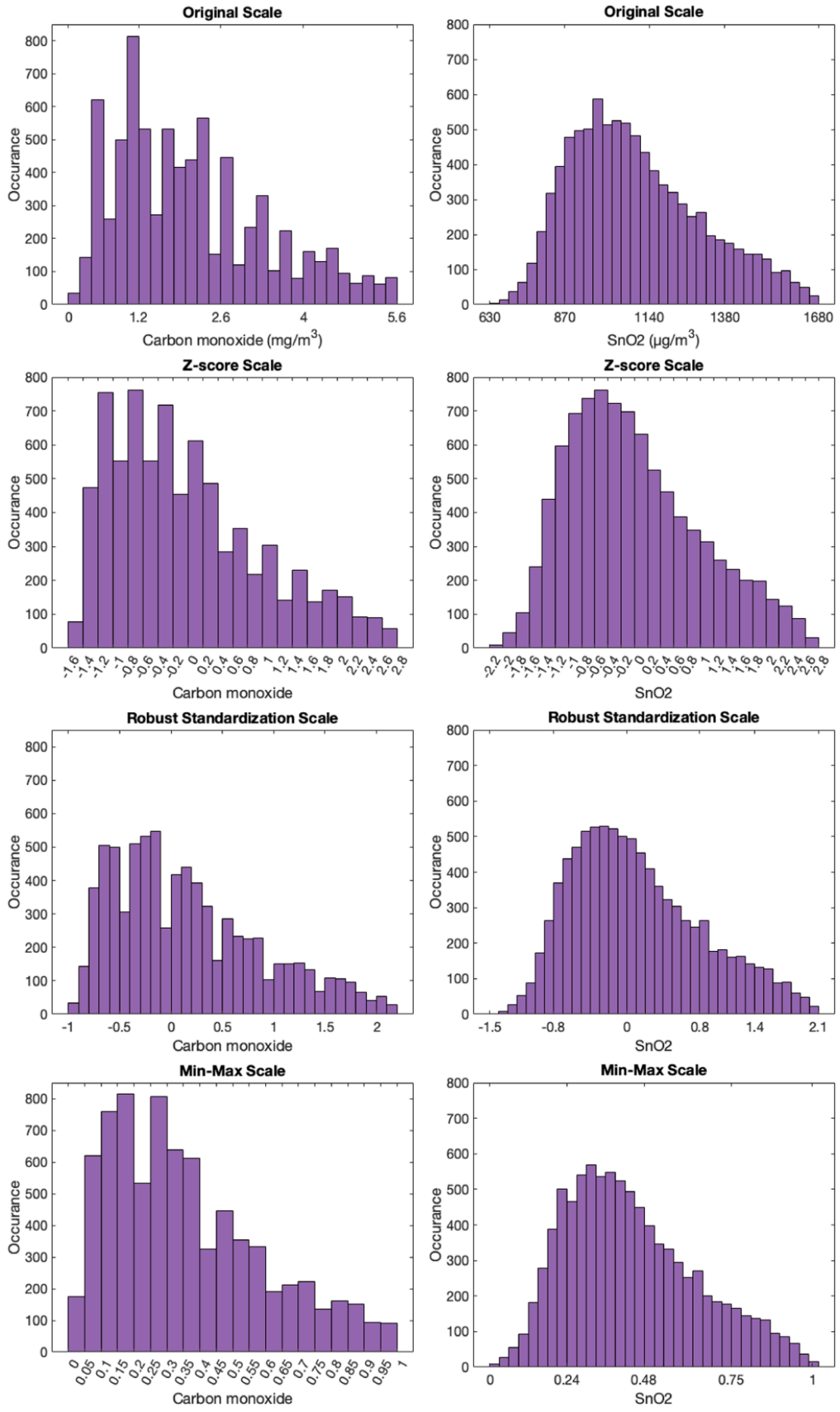


Fig. 5. Impact of Normalization Techniques on Features' Distribution and Scale.

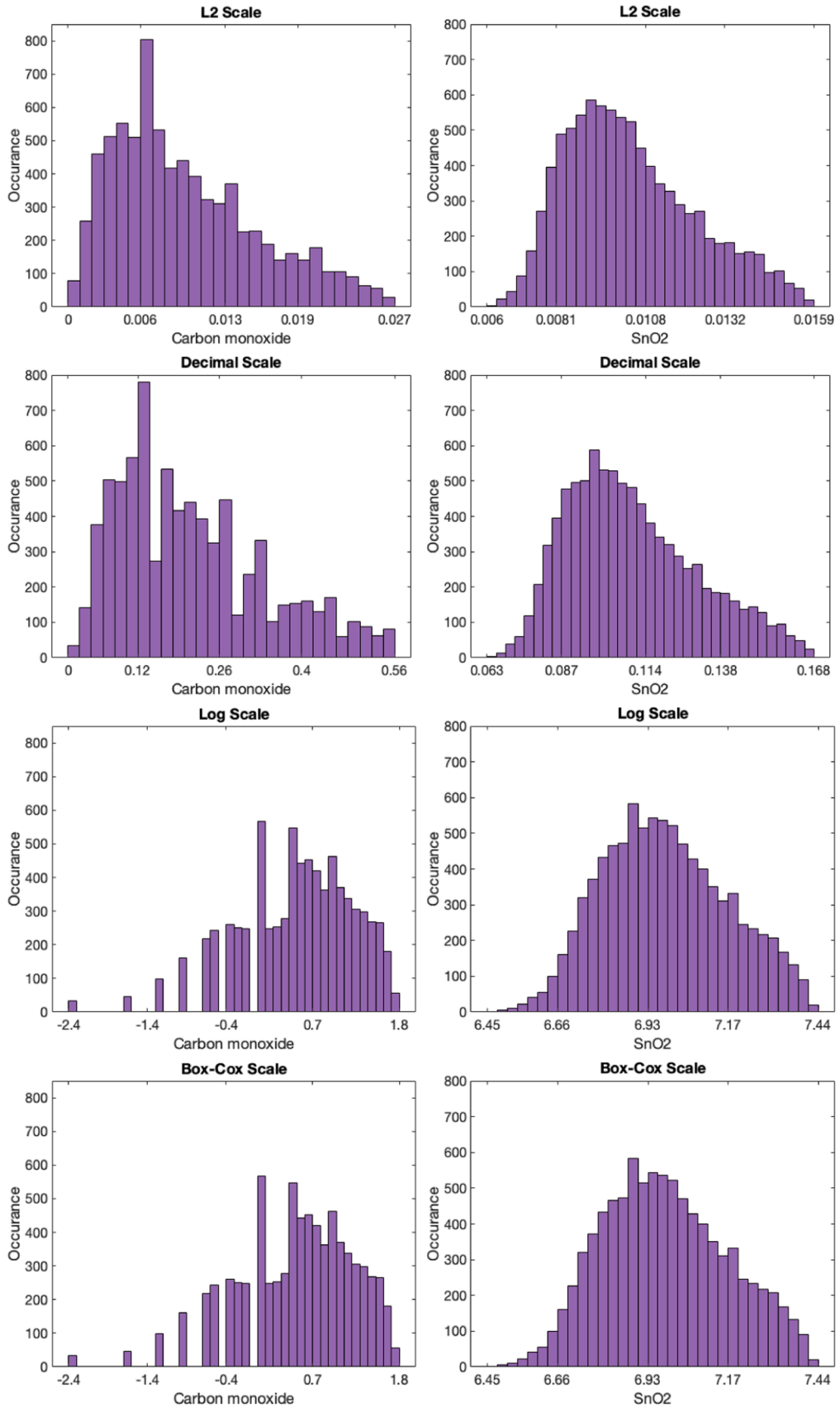


Fig. 6. Impact of Normalization Techniques on Features' Distribution and Scale.

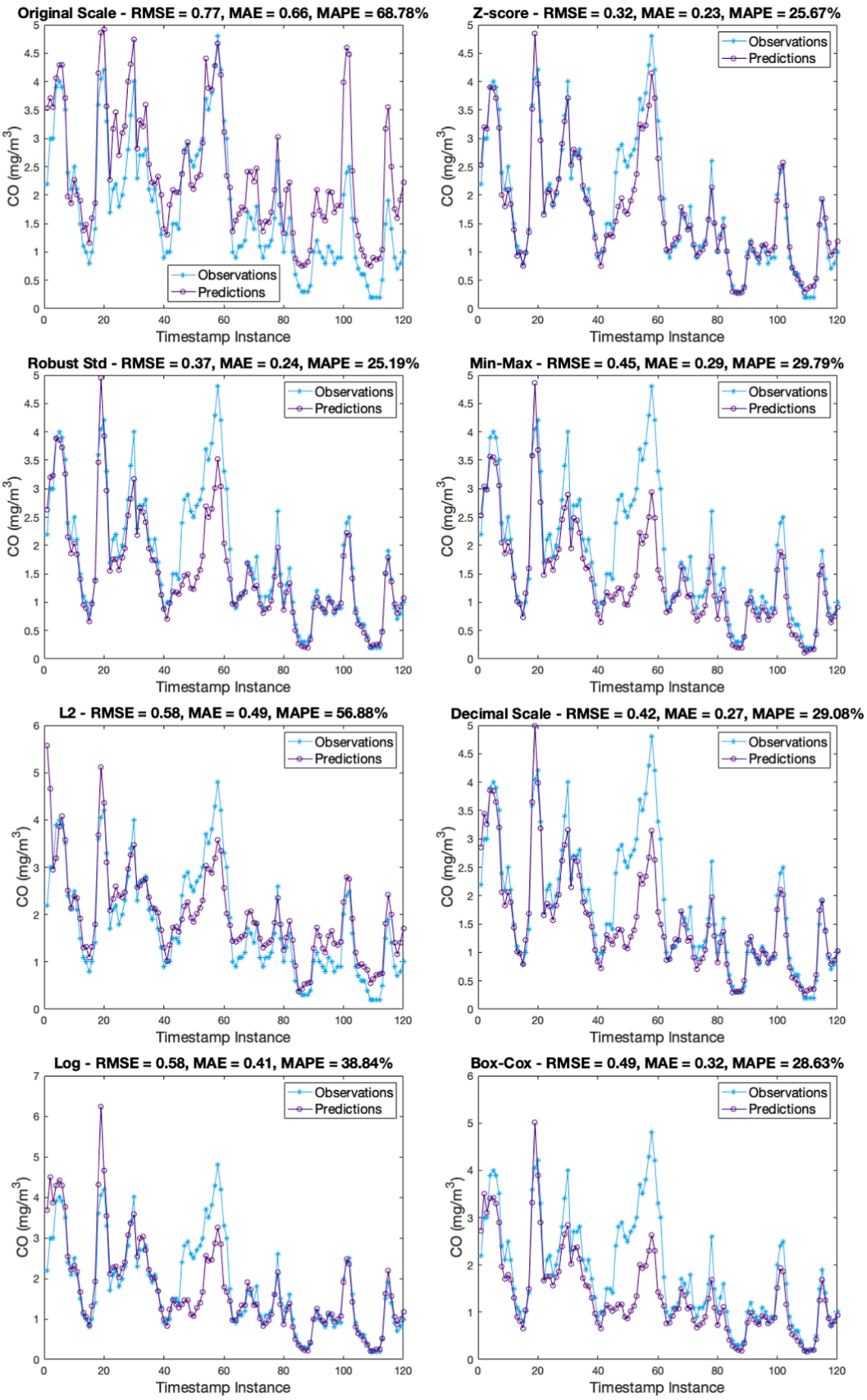


Fig. 7. Impact of normalization techniques on the prediction accuracy.

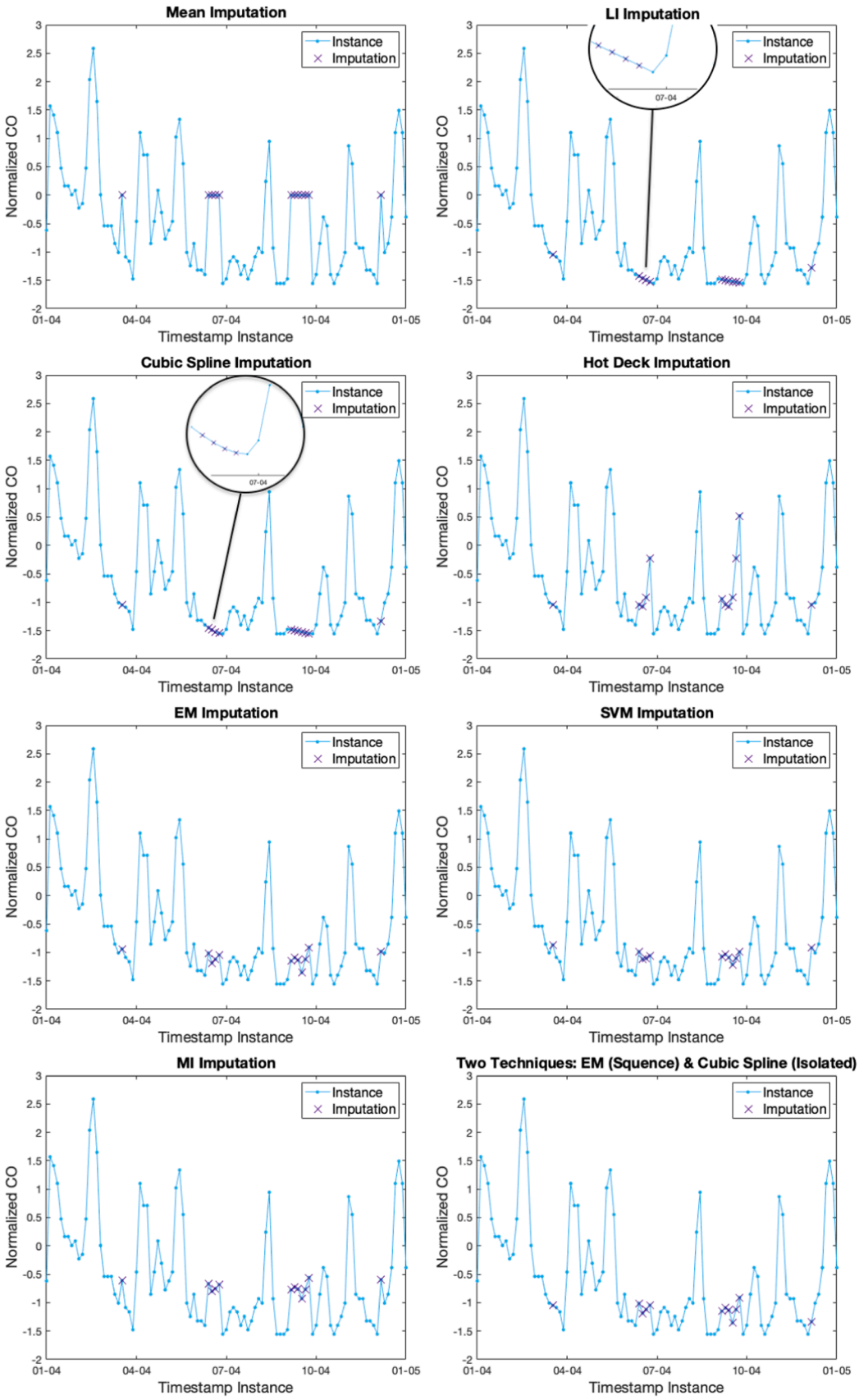


Fig. 8. Estimated Imputations of Missing Data of the Tested Techniques.

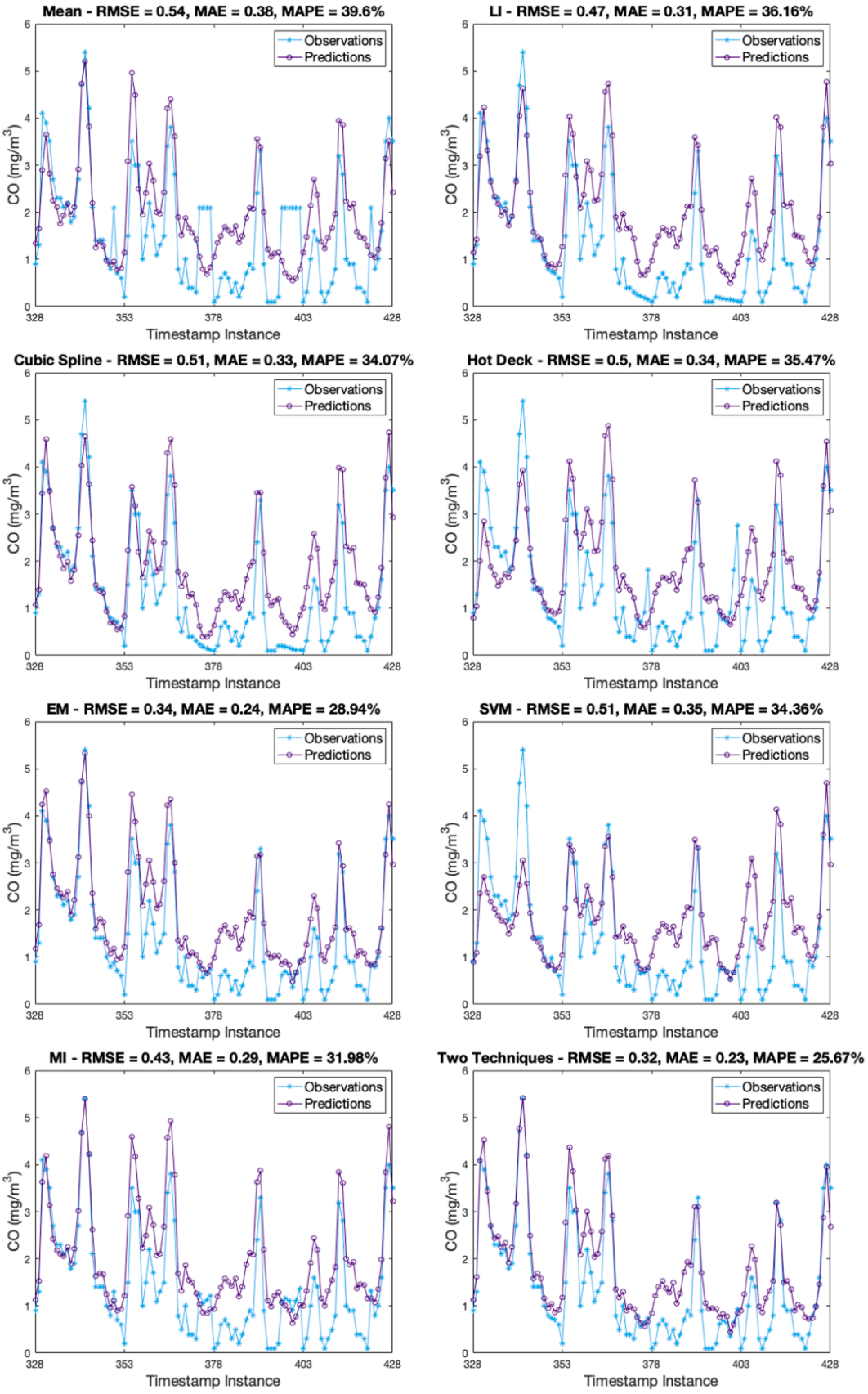


Fig. 9. Impact of Missing Data Imputation Techniques on Prediction Accuracy.

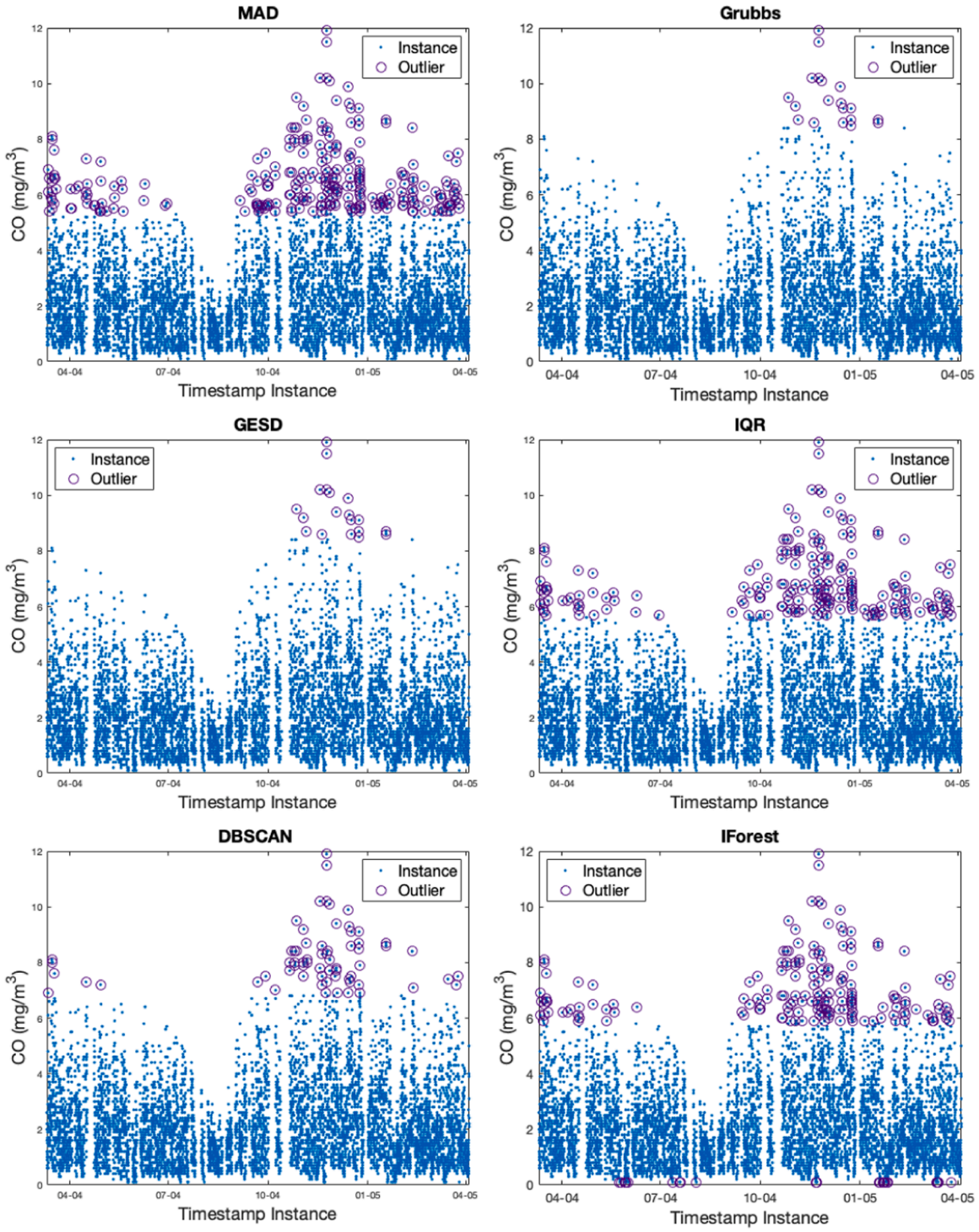


Fig. 10. Outliers Detected by the Different Techniques Tested.

Feature Extraction Techniques: We surveyed and tested two popular FE techniques that can be generalized and applied on a wide range of data including IoT data. The feature extraction techniques are summarized in Table 10.

Principle Component Analysis (PCA) is a linear feature extraction method that transforms correlated features to uncorrelated orthogonal vectors called Principle Components. It reduces the data into a lower dimensional linear subspace by replacing the original features that may consist of redundancies with new but fewer features that adequately summarize information contained in the original feature space [162]. Hence PCA uncovers low dimensional patterns that can be used to build better models. The first step of PCA is centering the data matrix X representing the multi-dimensional feature space using the mean matrix to obtain a mean centered data matrix B. This is followed by computing either the singular value decomposition (SVD) of B to obtain the loadings and scores or Eigen decomposition of the covariance matrix B^TB to obtain the eigen values and vectors. Both decomposition approaches provide the same principle components [19]. The decomposition summarizes the statistical variations of the dataset at new axes or directions. The newly derived axes are called Principle Components (PCs) and are

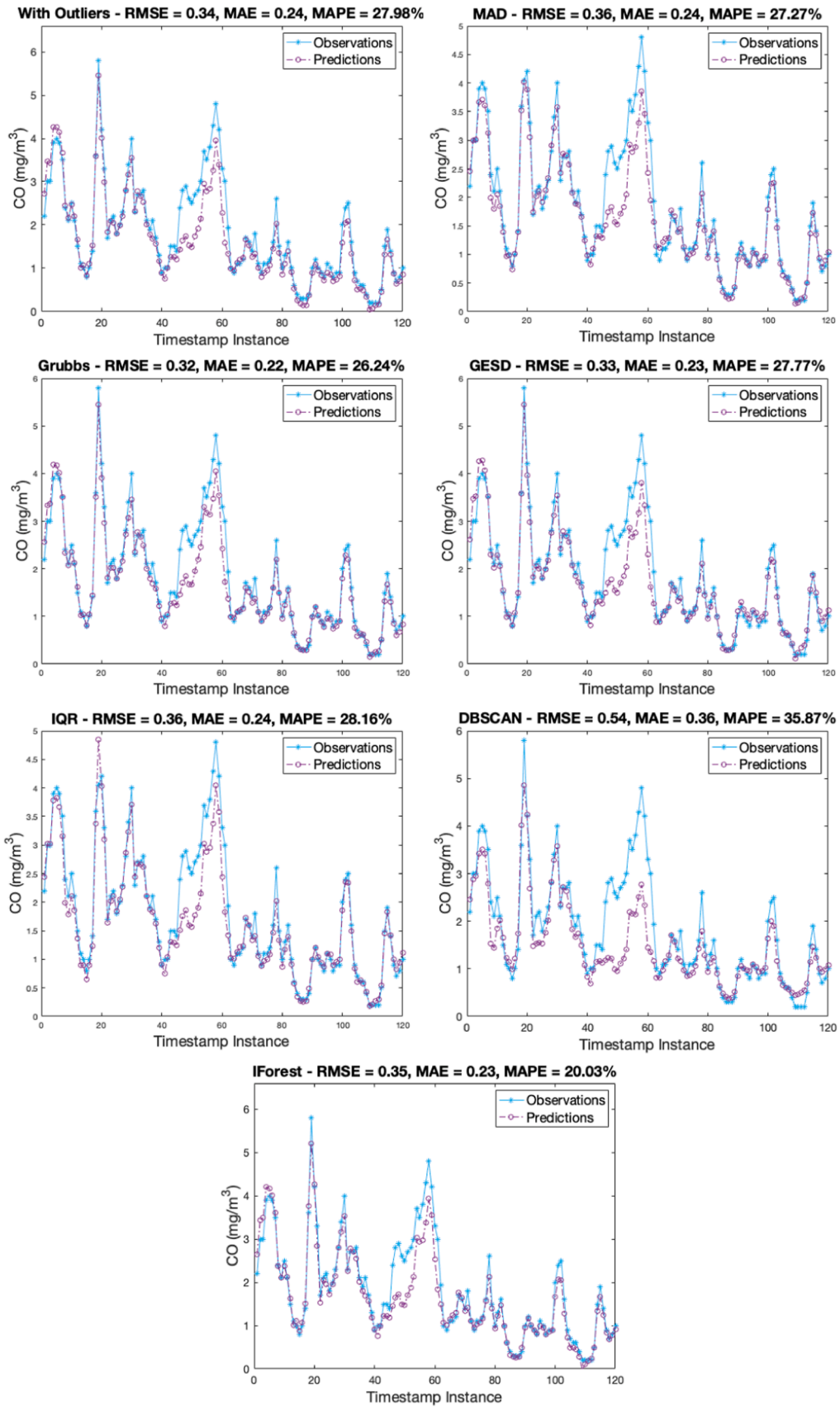


Fig. 11. Outlier Detection Techniques' Impact on Prediction Accuracy.

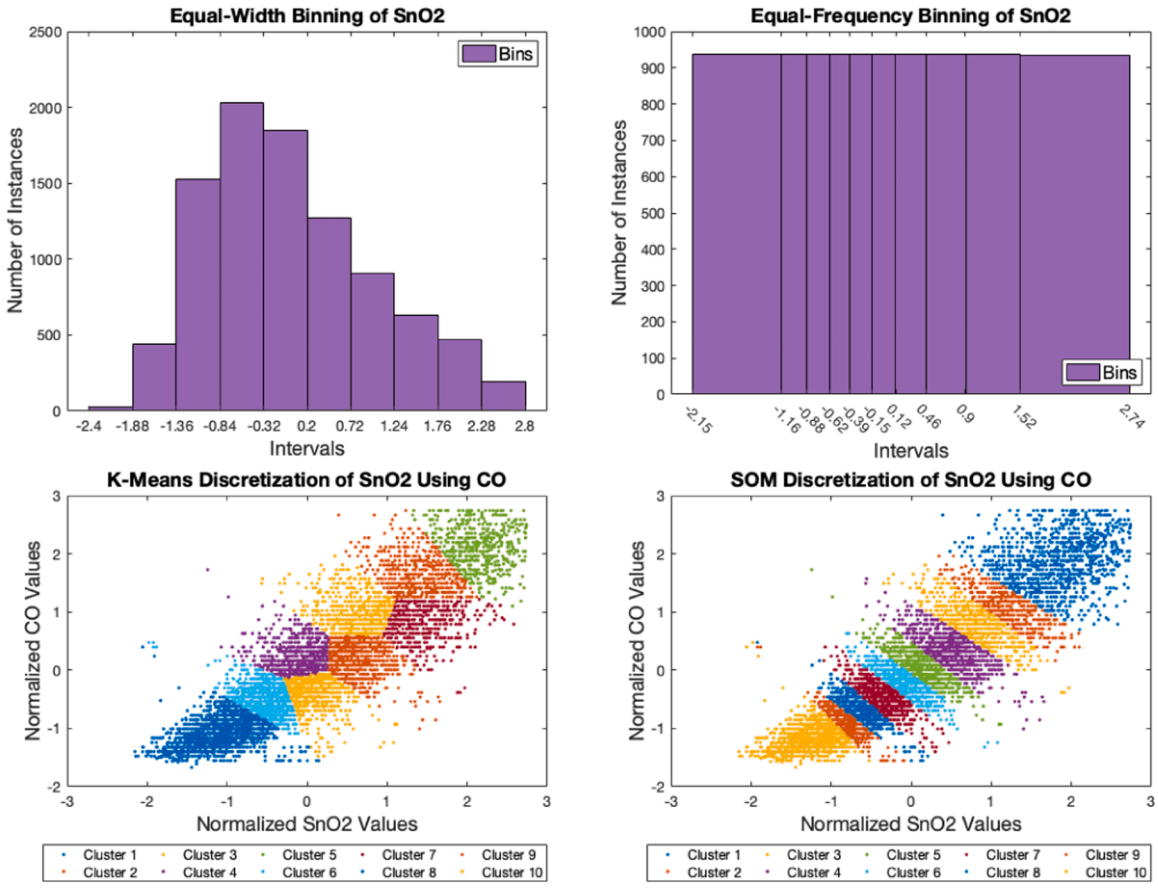


Fig. 12. Intervals and Bins Generated From the Tested Discretization Techniques.

equal to the number of features in the dataset. The PCs are ordered based on their magnitude with the first PC having the maximum variance and thus encapsulates most of the information in the data. Dimensionality reduction is possible with minimum loss of information by choosing the first n principle components as they maximize variation and better separate the data. The final step involves projecting the data into the lower dimensionality space, which can then be used as input for ML and DL models [4,19,162]. PCA, however, is computationally expensive and difficult to perform on data streams [162]. It is also sensitive to feature scales and outliers and thus normalization and outlier removal should be prerequisite steps [4,19,162]. For such reasons, applying PCA on sensor data should be periodically performed centrally to capture changes in patterns of multiple sensor data.

Linear Discriminant Analysis (LDA) is a Gaussian maximum likelihood classification used as a feature extraction technique. It creates new linear axes from exiting features that maximize the separability of the different classes. This makes LDA a supervised technique where the predictors are continuous and the response variable is categorical with two or multiple categories (i.e., classes). The algorithm searches for a vector in the underlying space that best discriminates between the different categories and creates a linear combination of features that yields the maximum mean difference between the classes and minimum variation within each class [135,163]. Details of LDA algorithm are presented in Table 10. LDA assumes that the underlying distribution of each class is Gaussian and that the classes have the same covariance. Quadratic Discriminant Analysis is a variation of LDA that allows for different feature covariance matrices for different classes thus producing quadratic separation axes [201]. In [63], authors proposed a fast incremental LDA for feature extraction that can be deployed for streaming data, which can be more ideal for IoT applications. Authors of [135] concluded from their empirical evaluation that PCA can outperform LDA in cases where the number of instances for each class value are small and when the training data do not uniformly sample the underlying distribution. LDA, however, is applicable on supervised problems where the response variable is categorical.

Other feature extraction techniques include Convolutional Auto-Encoders for non-linear dimensionality reduction [136], Gradient Direction Histogram (HOG) [32] and Speeded-Up Robust Features (SURF) [8], as image feature descriptors.

In [184], the feasibility of distributing LDA to IoT components was demonstrated. Using the training data, we computed the linear discriminants matrix of the new subspace. The matrix was then distributed to the edge and multiplied with new data to transform them onto the new subspace. Our empirical results demonstrated the feasibility and efficiency of distributing the linear discriminants after training and highlighted significant improvement in the consumption of resources including energy. Several examples [15,196,198] in literature distribute PCA using edge computing or parallel computing to reduce the dimensionality of data. These solutions either partly or completely train PCA on partitions of data streams. In [196,198], the results (local parameters) from distributed devices are aggregated into a global PCA model in a central system.

Experimental Results: We kept our experiments consistent with the feature selection experiments by using the top five features extracted by each tested technique. The first five principle components extracted by PCA explained approximately 93–94% of the total variance. Training with features extracted by PCA produced a less accurate model compared to the tested feature selection techniques and compared to using all 14 features (see Figs. 15 for comparison). LDA required using a discretization technique to transform the continuous response variable *CO* to a categorical format. We used equal-width binning to discretize the response variable *CO* into 10 categories. The predictors were kept as

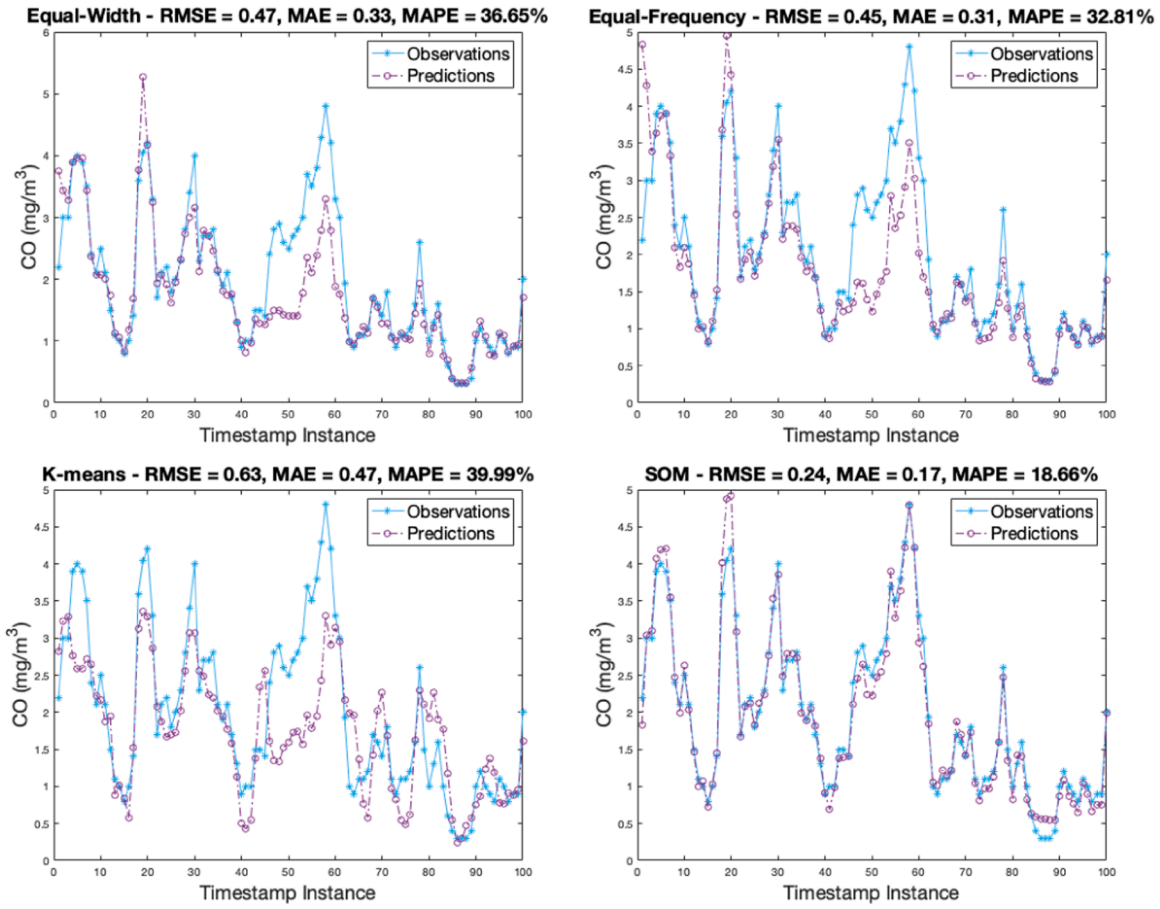


Fig. 13. Discretization Techniques Impact on Prediction Accuracy.

continuous variables as required by the LDA algorithm. We used the top five LDA extracted features to train an LSTM model. The produced model has superior accuracy compared to the rest of the feature reduction techniques tested. We attribute the significant variation in accuracy between LDA and PCA to the fact that the former is supervised and uses summarized information from the response variable in determining the new feature space. The results of our feature extraction test are presented in Figs. 16.

Data compression

The amounts of data generated by edge devices is growing rapidly. While this development is one of the major enablers of data-intensive value extraction techniques such as Machine Learning using Deep Neural Networks, which rely on vast amounts of data for creating highly complex models, it also stresses the computing and communication infrastructure. At the edge, available storage is typically smaller than at central utilities, and transmission to these may be costly and slow [159]. At central utilities, while storage capacity is larger, usually the data from many edge devices is collocated. Compressing sensor data can alleviate the storage bottlenecks occurring at the edge and at central utilities as well as reduce the cost and duration of data transmission [6,82]. While the goal of data compression is always to find a smaller representation of data, different families of techniques rely on different paradigms for data compression. In the following, we will differentiate such techniques along two major dimensions. The first of these dimensions is the processing paradigm: On the one hand, techniques that compress via multiple passes over the data, and on the other hand stream compression techniques that compress data in a single pass. The former allows for greater compression, while the latter's greedy nature enables high data throughput and the processing of unbounded data streams even on low-powered devices. A second main divider is the *faithfulness* of the compression; one may differentiate between lossless and lossy compression techniques. Lossless compression allows an identical reconstruction of the original data from the compressed representation, while lossy compression leads to strictly smaller data volumes while at most allowing guarantees on the deviations between the original and reconstructed data. We will focus on general techniques suitable for compressing sensor data in the form of numerical multivariate time-series data, to present compression tools viable in a wide range of scenarios (in contrast to more data-specific techniques such as for example image [93], video [147], or LiDAR data [94] compression).

Presentation of compression techniques

Multi-pass & lossless. In the first class of techniques, we regard multi-pass lossless techniques. An early variant is presented by the DEFLATE algorithm [39] (see also Table 11), that is implemented for example in the gzip file format. Using a general dictionary-based compression scheme (e.g., a mapping from symbols between the raw and the compressed data representation that matches the most frequent raw symbols to the smallest representation), DEFLATE can compress arbitrary types of data, including numerical time-series data (as evaluated in [82]). However, creating the necessary dictionary requires an additional pass over the data. Google's Brotli [2] is a more recent compression algorithm that expands on DEFLATE's compression paradigm. While optimized for web page compression, Brotli is capable of general-purpose compression, and its use of a predetermined static dictionary is beneficial for shorter sequences of data and lends the compression scheme more to the streaming paradigm. Facebook's Zstandard [30] dictionary

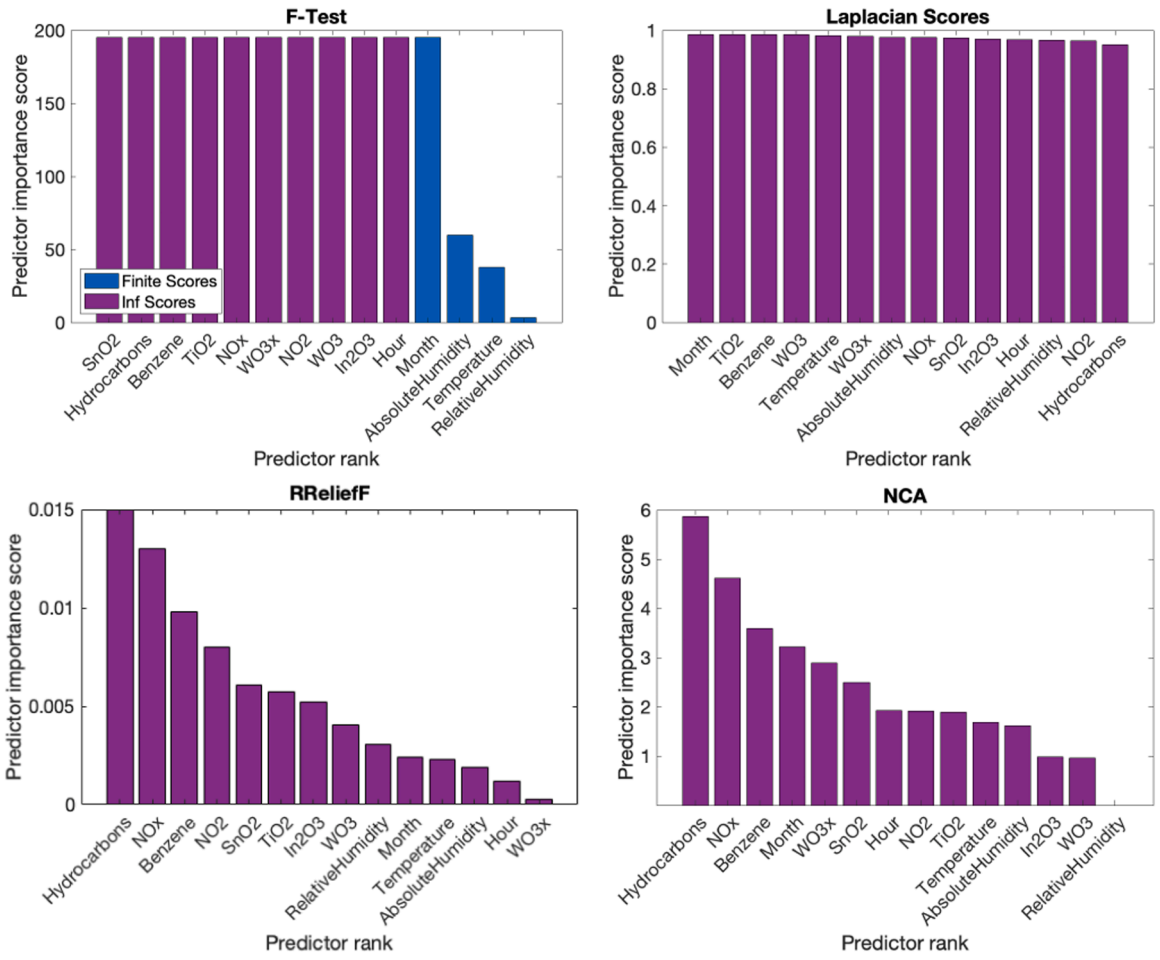


Fig. 14. Feature Selection Techniques' Feature Importance Scores.

compression algorithm further emphasizes static dictionaries by including support for training data-specific dictionaries on provided example files, thus allowing to tailor to specific data types before deployment for example at an edge device.

Single-pass & lossless. For the compression of continuously-valued time-series, dictionary-based approaches that rely on the frequency of symbols can be disadvantaged. A class of compression algorithms better adapted to this scenario is that of delta encoders, which losslessly encode the series of differences of data in a single pass. One variant is base-delta encoding, presented in [154], which encodes values as their distance from a base value. In [153], this idea was used to introduce an online compression algorithm specifically suited for increasing the throughput in data streaming systems.

Lindstrom and Isenburg present a compression scheme suitable for variable-precision floating point and integer data [121] that calculates the bitwise XOR of a value with previous series values, and provides competitive performance for 2D and 3D visualization data, while Burtscher and Ratanaworabhan's FPC algorithm, using the same idea for double-precision floats, significantly outperforms DEFLATE-family dictionary compressors in speed [20]. Facebook's Gorilla database [155] employs a similar scheme, specifically tailored to time-series data compression, by using delta-of-deltas encoding for timestamps (compressing 96% of timestamps in the authors' testing to a single bit) and a modified floating-point XOR compression scheme for the associated values (see also Table 11). A contrasting approach is present in earlier work for double-precision floats by Ratanaworabhan et. al. [161], where XORing of each value is performed with a prediction for said value based on a hash-table lookup, inspired in parts by [170].

Single-pass & lossy. The lossless compressibility of data is reversely tied to the data's entropy. Lossy compression, on the other hand, can achieve arbitrary compression rates, while sacrificing precision. Several techniques in the domain of lossless compression effectively reduce the data's dimensionality and thus enable more efficient search, while also compressing the data volume. Discrete Fourier Transform (DFT) [36,53] divides a time-series into a superposition of periodic functions, allowing a decomposition of data into its main frequencies (and a large data reduction if storing only the coefficients of leading contributions), but suffers from the assumption of periodic data. Wavelet transforms abandon this assumption and have partially superseded DFT by using a different set of local decomposition functions [158]. Using specialized compression schemes, wavelet transforms may also provide guarantees about the faithfulness of reconstructed data [61,91]. In [65], the authors present a wavelet-based approximation technique for continuous data streams of time-series data, with the promise of broad applicability to various types of signals.

A very intuitive subclass of lossy compression techniques is that of piecewise linear functions, which encode a time-series of data points by finding a set of linear approximations for subsequences. PAA (piecewise-aggregate approximation) encodes a sequence of points by their average value, thus resulting in an encoding of constant segments. In [101,210], the authors present such an encoding scheme with a constant segment length, enabling very efficient compression (by encoding each segment with only one value, its average) in a single-pass fashion. However, this approach neglects potential error bounds defined by the user of the

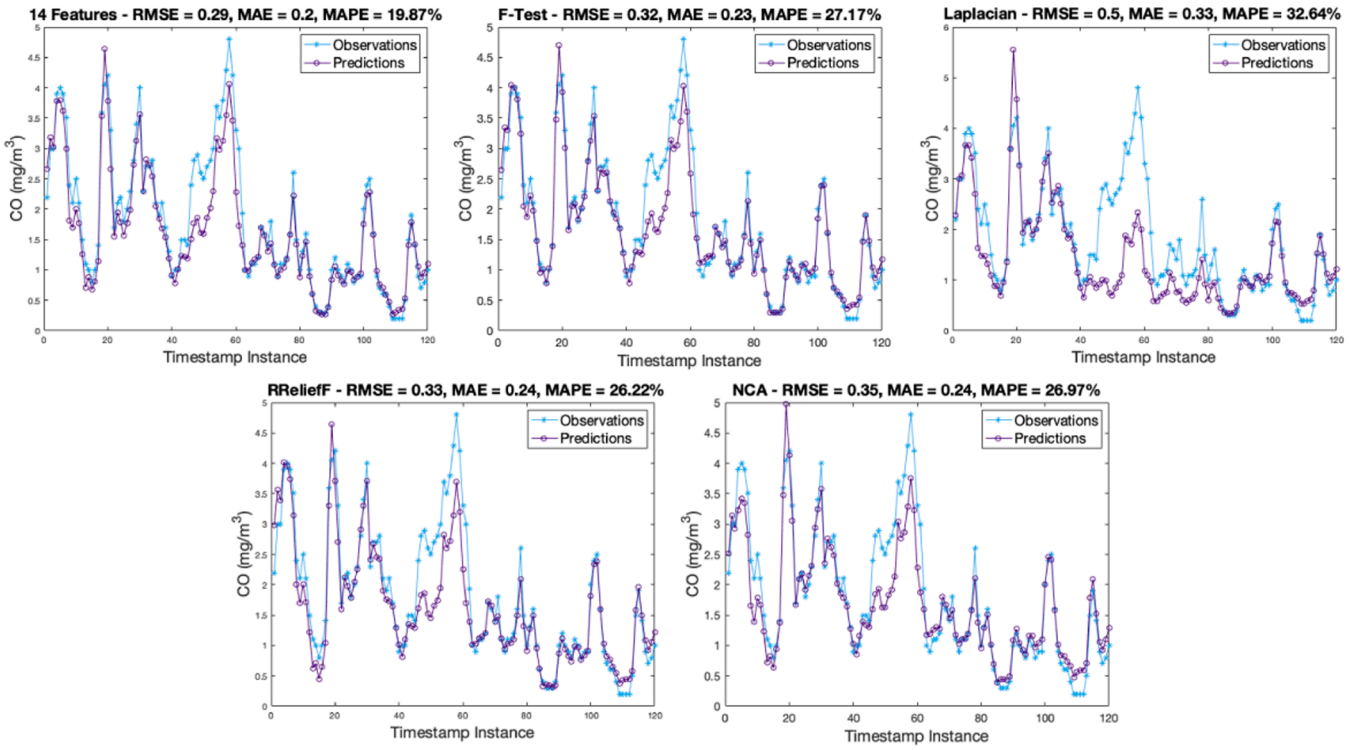


Fig. 15. Feature Selection Techniques' Impact on Prediction accuracy.

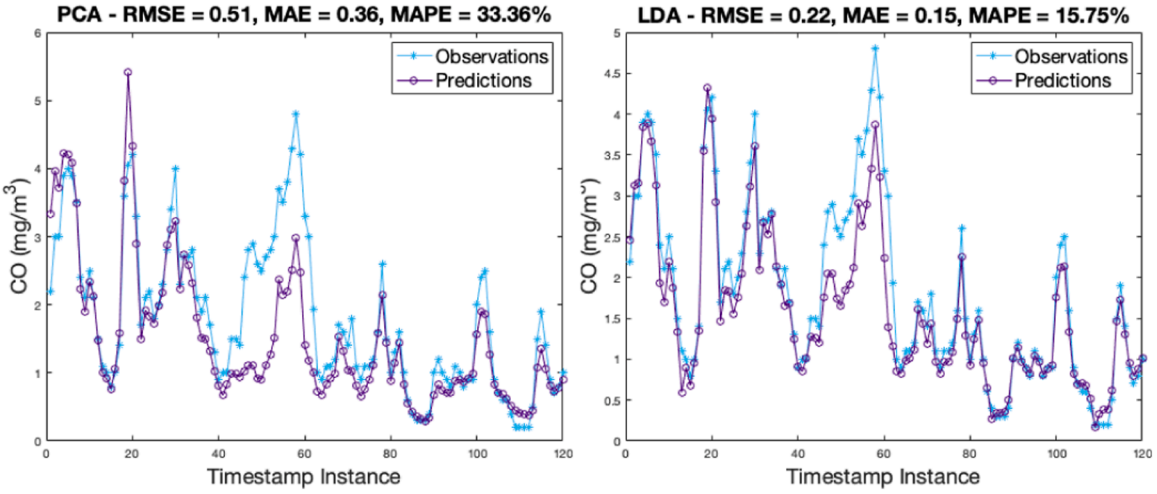


Fig. 16. Feature Extraction Techniques' Impact on LSTM accuracy.

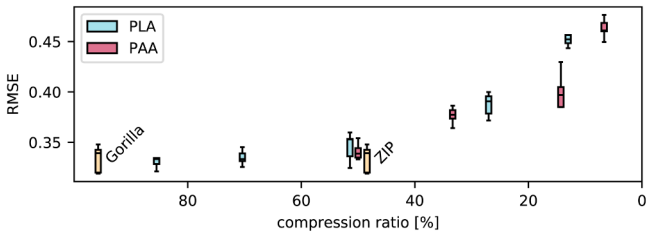


Fig. 17. RMSE achieved in LSTM experiment using compressed-decompressed data plotted against the respective compression ratio of the compressed data (boxplots of five experiment repetitions).

compression. [102] makes the segment length of PAA adaptive, at the cost of requiring multiple passes over the data, but delivering segments that minimize the reconstruction error. In [120], constant-length segments are combined with a string representation for the segment values, where the fixed-size set of strings offers a second layer of encoding (the resulting strings are prominently used for time-series similarity search). PLA (piecewise-linear approximation) is a similar technique that fits line segments of certain length and slope to subsequences of data and then encodes the line segment parameters, while commonly ensuring that the approximation error is bounded. A host of variations of PLA exist, e.g., [10,48,129], that mainly differ in how the error bound is enforced. As a common feature, these techniques all support single-pass greedy ingestion of points, as only a window of past points has to be maintained (for each subsequence). [47] furthermore uses an adaptive streaming output protocol that can also encode singleton values (values that are not

compressed) to ensure that the protocol is non-inflating, while [82] builds on this technique to deliver an implementation that compresses the time and value streams independently from each other.

Combined approaches. To forgo the disadvantages of single techniques, or combine their strengths, a body of techniques exist that present a mixtures of different compression approaches. In [12], the authors present Sprintz, a compression scheme that combines dictionary-based compression with delta encoding to further reduce data sizes. Their technique is optimized for deployment on highly resource-constrained edge devices. Similarly, in [203] the authors suggest combining PLA with delta encoding for on-board compression of vehicular data. [6,40] present Squeeze, which locally optimizes the prediction of the next data point in a series by choosing among constant, linear, or quadratic fitting, or using a binary compression scheme, and demonstrate its advantages in edge deployment scenarios. Finally, the authors of [128] present a middleware that chooses the optimal compression method on the edge among a set of dictionary-based (e.g., DEFLATE) or combined (Squeeze) techniques, relative to available CPU and network capacity as well as the data type.

Selected evaluations

Table 11 displays a selection of compression techniques, representative of different approaches to time-series compression, in more detail: PAA, PLA, Gorilla, and ZIP. The selection of highlighted techniques was partly guided by the availability of source code (or ease of reproduction). For these techniques, we evaluate in the following the achievable compression ratio (in % of original binary data volume) as well as the achievable accuracy as measured by the RMSE when using compression for the running use-case of LSTM-based time-series prediction, to give pointers concerning the trade-offs these techniques introduce. After the preprocessing of the whole dataset, the training set **AirQuality**[193] is compressed and decompressed. In the case of PAA and PLA, the resulting dataset is an approximation of the original data (due to the lossy character of the compression). Then, the LSTM is trained using the decompressed data, and evaluated against the testing set (which has not undergone compression).

Figure 17 shows the results of the experiment, with the RMSE achieved in the LSTM experiment plotted against the compression ratio achieved by the respective technique (boxplots of five experiment repetitions). Gorilla and ZIP are shown as single boxplots, as the compression achieved with these techniques is not variable. As both are lossless, the achieved RMSE equals the one achieved on data that has not undergone compression and decompression. It appears that Gorilla has difficulties compressing the data used in this experiment, achieving a compression ratio below 90%, while ZIP compresses the dataset to around 50% of its original size, demonstrating the general-purpose nature of the deflate algorithm used in ZIP. The authors of Gorilla [155] demonstrate much more beneficial compression ratios on time-series data than the one observed here, which may be an indication of the correspondence between signal shape and achievable compression for Gorilla. The variable compression ratios of PLA are achieved by varying the maximum error bound, and those of PAA by varying the segment length (see Table 11). Both compression techniques demonstrate that any reconstruction losses from PAA and PLA do not hurt the achievable RMSE (which is close to that of using no compression), with clearer trade-offs for higher compression ratios. Clear comparisons between PAA and PLA are difficult, but the graph hints that for reducing the compression ratio from 50% to below 20% leads to a reduction in RMSE of 70%, and scales roughly linearly in the region in-between. From this experiment, one may choose PLA or PAA compression over ZIP for the potential speed benefit of a single-pass technique, and choose a compression ratio that is aligned with a target RMSE. It should be noted here that while using lossy compression leads to factually lower RMSE, it potentially allows the transmission and storage of larger amounts of data, thereby potentially increasing the accuracy of a trained model and thereby the RMSE.

Preprocessing steps governed by dependencies

It is important to highlight that dependencies between preprocessing categories and techniques exist. As an example, outlier detection and removal can precede data normalization such that the data mean and variance for normalization are not distorted by artifacts. Similarly, missing data imputation may require the data to be normalized first, in case the imputation method is sensitive to the scale of the data. A more detailed analysis of dependencies between different preprocessing categories and their impact can be found in [183].

In general, preprocessing can encompass any combination, including repeated application, of individual preprocessing techniques. This combination of preprocessing techniques can usually be expressed as a directed acyclic graph (DAG), where nodes are individual preprocessing steps (preprocessing tasks) and edges are the dependencies between them. Fig. 1 shows how a series of preprocessing steps can be modeled as a DAG, with data flowing along the arrows and nodes being preprocessing steps or operators: First, outliers in the data are detected using the data's interquartile range (IQR) `step 1`. In `step 2`, the data is normalized to the 0, 1 interval. Now, missing data is to be imputed. To improve the performance of the imputation, in `step 3` isolated missing instances are passed to a Linear Interpolation (LI) preprocessing step, while sequences of missing data are imputed using Expectation Maximization (EM); afterwards, the data is passed forward as a single dataset again. In `step 4`, important features of the data are selected using multiple FS techniques, including RReliefF, F-test, and NCA.

Modeling applications in this fashion as graphs of operators is a common technique used, for example, by parallel and distributed Big Data processing tools such as Apache Spark and Apache Flink. Viewing preprocessing in this light allows us to leverage results, especially from the latter field, to optimize the processing of more complex preprocessing DAGs. As an example, `step 4` in Fig. 1 may be situated in the cloud due to the increased computational complexity, while `steps 1–3` could be deployed on said edge device close to the data source. This notion of targeted deployment of operators, e.g., to reduce processing latencies or optimize resource usage, is widely researched in the context of stream processing (see, e.g., [41]).

While it is not in the scope of this survey to review results from the field of data stream processing, we want to highlight one more interesting connection. As the nodes in a preprocessing graph can have non-trivial interdependencies, it can be difficult to reason about, replicate, or improve upon the preprocessing graph, which introduces novel challenges that may be overlooked when considering only individual preprocessing steps. Here, the concept of data provenance may be helpful, as it allows us to connect results from various processing steps with each other to answer questions about why and how a result came to be. [66] (for the stream processing engine Borealis) offers fine-grained data-level provenance by instrumenting the individual processing steps of the processing graph. Genealog [150] is a similar framework that delivers fine-grained provenance in, among others, the Apache Flink stream processing engine while incurring only a constant overhead per data point for tracking data provenance. In addition to providing insight into individual data transformations or the data workflow at large, data provenance may also be employed for sophisticated selection of input data, as demonstrated in [151]: Rewriting the preprocessing step of data selection as a query over the input data that may involve multiple transformations and aggregations with secondary data types and a final filter for checking conformity to a desired attribute, provenance allows the retrieval of exactly that input data that contributed to any result passing the final filter. Tools like these can aid in disentangling, understanding, and improving complex preprocessing workflows, and they underline how the adoption of a graph view of data preprocessing can aid in improving the eventual quality of the data, but also the process of

preprocessing itself.

Conclusion and future work

In this survey, an extensive number of preprocessing techniques were surveyed under each category. In addition, we empirically evaluated the impact of selected techniques on the data and on the prediction accuracy of an LSTM network centrally trained on the respective preprocessed data. This makes the survey a distinctive account that includes both theoretical and empirical analysis. We expanded the scope of data preprocessing to include traditional categories such as data cleaning and non-traditional categories such as data compression. We also highlighted the importance of considering dependencies between different preprocessing techniques and categories as they impact the performance and functioning of the dependent techniques. The survey focuses on preprocessing numerical time-series data, which is significantly collected from the edge given the popularity of Internet of Things applications. Thus, a third dimension of the survey includes discussions and examples that suggest the applicability and feasibility of deploying preprocessing techniques to the edge and the promising impact of such distribution. The survey is a starting point and a guide for researchers and practitioners to address data preprocessing comprehensively, effectively, and via a standardized and normalized approach.

Due to the depth and scope of this survey and other factors and limitations, our experiments were restricted to one dataset and to selected preprocessing techniques from some categories. To remove biases and extract generalized conclusions, the empirical analysis should be extended to other real-world datasets in future work. We recommend selecting datasets from different domains that demonstrate different characteristics (e.g., different distributions). Having said that, our results provide valuable insights and highlight disparities in the impact of the different preprocessing techniques under the same category on both the dataset and the AI algorithm consuming the data. The impact of some categories or techniques on the data can also be generalized due to their fixed objectives. For example, regardless of the dataset and original feature scale, min-max scaling will always transform features to a scale between zero and one. We also plan to extend our empirical analysis to cover categories that were not evaluated in this survey. For example, handling non-stationarity (changing data characteristics) in the data as a new data preprocessing category [169]. Lastly, we identify the need to systematically evaluate the feasibility and impact of distributing data preprocessing techniques to the edge to be applied on IoT data early in the dataflow.

Declaration of Competing Interests

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

Data Preprocessing: Definition and Scope

We introduce an extended definition of data preprocessing based on a holistic and modern prospective. We also present the first taxonomy of data preprocessing based on the proposed definition:

Definition. Data preprocessing includes any operation performed on the data, prior to information and value extraction and after data ingestion, to improve the quality of the data and prepare it for the consuming algorithms. The operations fall under several categories as highlighted by the taxonomy in Fig. 4 to achieve many data quality characteristics including accuracy, unbiased, completeness, and traceability

Figures of the Empirical Results

We used different types of figures (e.g., plots, bar-charts, etc.), included in the appendix, to present the impact of the preprocessing techniques on the data (Input Quality). The figures highlight the transformation of the data or outline the outcomes and diagnosis of the techniques. We align the figures presenting the impact of the different techniques for effective comparison and provide a baseline only when relevant.

We also evaluated the impact of the preprocessing techniques on AI algorithms (Output Quality), specifically LSTM networks. The performances of the trained LSTM networks under the different preprocessing categories are presented in two dimensional line plots of the predicted values against the observations after preprocessing, if any was applied. The plots represent the results of the last test performed of that particular experiment including the metric values at the top of the plot.

We present a graphical representation of our empirical results based on the two main evaluation approaches clarified above and in the order the category was discussed in the survey Fig. 5, 6,7,8, 9, 10, 11, 12, 13, 14, 15, 16, 17.

References

[1] D. Adhikari, W. Jiang, J. Zhan, Z. He, D.B. Rawat, U. Aickelin, H.A. Khorshidi, A comprehensive survey on imputation of missing data in internet of things, ACM Comput. Surv. (2022) (Just Accepted).
[2] J. Alakuijala, A. Farruggia, P. Ferragina, E. Kliuchnikov, R. Obryk, Z. Szabadka, L. Vandevenne, Brotli: A general-purpose data compressor, ACM Trans. Inf. Syst. (TOIS) 37 (1) (2018) 1–30.
[3] S.-A. Alexandropoulos, S. Kotsiantis, M. Vrahatis, Data preprocessing in predictive data mining, Knowl. Eng. Rev. 34 (2019).
[4] A.C. Alice Zheng, Feature Engineering for Machine Learning Models, O'Reilly UK Ltd., 2018.
[5] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, OPTICS. Proceedings of the 1999 ACM SIGMOD international conference on Management of data - SIGMOD' 99, ACM Press, 1999.
[6] J. Azar, A. Makhoul, M. Barhamgi, R. Couturier, An energy efficient IoT data compression approach for edge machine learning, Future Gener. Comput. Syst. 96 (2019) 168–175.
[7] C. Batini, C. Cappiello, C. Francalanci, A. Maurino, Methodologies for data quality assessment and improvement, ACM Comput. Surv. 41 (3) (2009).
[8] H. Bay, A. Ess, T. Tuytelaars, L.V. Gool, Speeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (3) (2008) 346–359.
[9] R.E. Bellman, Adaptive Control Processes: A Guided Tour, Princet. Leg. Libr. 04 (2016).
[10] E. Berlin, K. Van Laerhoven, An on-line piecewise linear approximation technique for wireless sensor networks. IEEE Local Computer Network Conference, IEEE, 2010, pp. 905–912.
[11] C.M. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag New York Inc., 2006.
[12] D. Blalock, S. Madden, J. Guttag, Sprintz: Time series compression for the internet of things, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2 (3) (2018).
[13] A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artif. Intell. 97 (1-2) (1997) 245–271.
[14] M.Khiops Boullé, A discretization method of continuous attributes with guaranteed resistance to noise, in: P. Perner, A. Rosenfeld (Eds.), Machine Learning and Data Mining in Pattern Recognition (Berlin, Heidelberg, Springer, Berlin Heidelberg, 2003, pp. 50–64.
[15] Boutsidis, C., Woodruff, D.P., and Zhong, P. Optimal principal component analysis in distributed and streaming models, 2015.
[16] E.P. Box George, G.C.R.G.M.J.G.M.L. Box, Time Series Analysis 5e, John Wiley & Sons, 2015.
[17] G.E.P. Box, D.R. Cox, An analysis of transformations, J. R. Stat. Soc. Ser. B (Methodol.) 26 (2) (1964) 211–252.
[18] Breunig, M.M., Kriegel, H.-P., Ng, R.T., and Sander, J. LOF: Identifying density-based local outliers.In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (New York, NY, USA, 2000), SIGMOD '00, Association for Computing Machinery, 93-104.
[19] S.L. Brunton, J.N. Kutz, Data-Driven Science and Engineering, Cambridge University Press, 2019.
[20] M. Burtscher, P. Ratanaworabhan, High throughput compression of double-precision floating-point data. 2007 Data Compression Conference (DCC'07), IEEE, 2007, pp. 293–302.
[21] J. Cai, J. Luo, S. Wang, S. Yang, Feature selection in machine learning: A new perspective, Neurocomputing 300 (2018) 70–79.
[22] Cai, L., Wang, Z., Gao, H., Shen, D., and Ji, S. Deep adversarial learning for multi-modality missing data completion.In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2018).

[23] R.J.G.B. Campello, D. Moulavi, A. Zimek, J. Sander, Hierarchical density estimates for data clustering, visualization, and outlier detection, ACM Trans. Knowl. Discov. Data 10 (1) (2015) 1–51.
[24] Capes, T., Coles, P., Conkie, A., Golipour, L., Hadjitarkhani, A., Hu, Q., Huddleston, N., Hunt, M., Li, J., Neeracher, M., Prahallad, K., Raitio, T., Rasipuram, R., Townsend, G., Williamson, B., Winarsky, D., Wu, Z., and Zhang, H. Siri on-device deep learning-guided unit selection text-to-speech system.In: Proc. Interspeech 2017 (2017), 4011-4015.
[25] F. Castanedo, A review of data fusion techniques, Sci. World J. (2013) 1–19.
[26] S. Chakrabarti, E. Cox, E. Frank, Data Mining: Know It All, Morgan Kaufmann Publ Inc, 2008.
[27] Z. Chang, S. Liu, X. Xiong, Z. Cai, G. Tu, A survey of recent advances in edge-computing-powered artificial intelligence of things, IEEE Internet Things J. 8 (18) (2021).
[28] D. Chickering, C. Meek, R. Rounthwaite, Efficient determination of dynamic split points in a decision tree. Proceedings 2001 IEEE International Conference on Data Mining, IEEE Comput. Soc, 2001.
[29] Coates, M. Distributed particle filters for sensor networks.2004 IPSN '04, Association for Computing Machinery.
[30] Collett, Y. Zstandard compression and the application/zstd media type ⟨https://tools.ietf.org/html/rfc8478⟩.Accessed: 2021-03-10.
[31] S. Dai, L. Li, Z. Li, Modeling vehicle interactions via modified lstm models for trajectory prediction, IEEE Access 7 (2019) 38287–38296.
[32] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE, 2005.
[33] D'Andrea, R., Beck, C., and Dullerud, G. Temporal discretization of spatially distributed systems.In: Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304) (1999), vol. 1.
[34] B.F. Darst, K.C. Malecki, C.D. Engelman, Using recursive feature elimination in random forest to account for correlated variables in high dimensional data, BMC Genet. 19 (S1) (2018).
[35] J.C. Davis. Statistics and Data Analysis in Geology, 3 ed., John Wiley & Sons, 2002.
[36] Davood Rafiei, A.M. Efficient retrieval of similar time sequences using dft.In: 5th Intl. Conf. on Foundations of Data Organizations and Algorithms(1998).
[37] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. 39 (1) (1977) 1–38.
[38] D'Este, C., Sharman, C., and Rahman, A. Distributed feature selection with big sensor data.2014 MLSDA'14, Association for Computing Machinery.
[39] Deutsch, L.P., DEFLATE compressed data format specification version 1.3 2021 ⟨https://tools.ietf.org/html/rfc1951⟩.Accessed: 2021-03-10.
[40] Di, S., and Cappello, F. Fast error-bounded lossy hpc data compression with sz.In: IEEE International Parallel and Distributed Processing Symposium (IPDPS) (2016), 730-739.
[41] M. Dias de Assunção, A. da Silva Veith, R. Buyya, Distributed data stream processing and edge computing: A survey on resource elasticity and future directions, J. Netw. Comput. Appl. 103 (2018) 1–17.
[42] W. Ding, X. Jing, Z. Yan, L.T. Yang, A survey on data fusion in internet of things: Towards secure and privacy-preserving fusion, Inf. Fusion 51 (C) (2019).
[43] Z. Ding, M. Fei, An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window, IFAC Proc. Vol. 46 (20) (2013) 12–17.
[44] Y. Dong, C.-Y.J. Peng, Principled missing data methods for researchers, SpringerPlus 2 (1) (2013).
[45] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features. Machine Learning Proceedings 1995, Elsevier, 1995, pp. 194–202.
[46] J. Du, M. Hu, W. Zhang, Missing data problem in the monitoring system: A review, IEEE Sens. J. (2015).
[47] R. Duvignau, V. Gulisano, M. Papatriantafilou, V. Savic, Streaming piecewise linear approximation for efficient data management in edge computing, Proc. 34th ACM/SIGAPP Symp. . Appl. Comput. (2019) 593–596.
[48] H. Elmeleegy, A.K. Elmagarmid, E. Cecchet, W.G. Aref, W. Zwaenepoel, Online piece-wise linear approximation of numerical streams with precision guarantees, Proc. VLDB Endow. 2 (1) (2009).
[49] C.K. Enders, Applied Missing Data Analysis, Guilford Publications, 2010.
[50] K.M. Engle, A. Gangopadhyay, An efficient method for discretizing continuous attributes, Int. J. Data Warehous. Min. 6 (2) (2010) 1–21.
[51] Ertöz, L., Steinbach, M., and Kumar, V. A new shared nearest neighbor clustering algorithm and its applications.2002.
[52] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, AAAI Press, 1996.
[53] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. Fast subsequence matching in time-series databases.In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data (New York, NY, USA, 1994), SIGMOD '94, Association for Computing Machinery, 419-429.
[54] C.M.D. Farias, W. Li, F.C. Delicato, L. Pirmez, A.Y. Zomaya, P.F. Pires, J.N. D. Souza, A systematic review of shared sensor networks, ACM Comput. Surv. 48 (4) (2016).
[55] Fayyad, U.M., and Irani, K.B. Multi-interval discretization of continuous-valued attributes for classication learning.In: International Joint Conference on Artificial Intelligence (1993), 1022-1029.
[56] N. Fouladgar, K. Främling, A novel LSTM for multivariate time series with massive missingness, Sensors 20 (10) (2020) 2832.
[57] Fountas, P., and Kolomvatsos, K. Ensemble based data imputation at the edge.In: 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI) (2020).
[58] Gandhi, S., Oates, T., Boedihardjo, A., Chen, C., Lin, J., Senin, P., Frankenstein, S., and Wang, X. A generative model for time series discretization based on multiple normal distributions.PIKM '15, Association for Computing Machinery.2015.
[59] S. García, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, Springer-Verlag GmbH, 2014.
[60] S. García, J. Luengo, J.A. Sáez, V. López, F. Herrera, A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning, IEEE Trans. Knowl. Data Eng. 25 (4) (2013).
[61] Garofalakis, M., and Gibbons, P.B. Wavelet synopses with error guarantees.In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (New York, NY, USA, 2002), SIGMOD '02, Association for Computing Machinery, 476-487.
[62] F.A. Gers, D. Eck, J. Schmidhuber, Applying LSTM to time series predictable through time-window approaches. Artificial Neural Networks — ICANN 2001, Springer, Berlin Heidelberg, 2001, pp. 669–676.
[63] Y.A. Ghassabeh, F. Rudzicz, H.A. Moghaddam, Fast incremental LDA feature extraction, Pattern Recognit. 48 (6) (2015) 1999–2012.
[64] N. Ghosh, R. Paul, S. Maity, K. Maity, S. Saha, Fault matters: Sensor data fusion for detection of faults using Dempster–Shafer theory of evidence in IoT-based applications, Expert Syst. Appl. 162 (2020).
[65] A.C. Gilbert, Y. Kotidis, S. Muthukrishnan, M.J. Strauss, One-pass wavelet decompositions of data streams, IEEE Trans. Knowl. data Eng. 15 (3) (2003) 541–554.
[66] Glavic, B., Sheykh Esmaili, K., Fischer, P.M., and Tatbul, N. Ariadne: Managing fine-grained provenance on data streams.In: Proceedings of the 7th ACM international conference on Distributed event-based systems (2013), 39-50.
[67] J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov, Neighbourhood components analysis. Proceedings of the 17th International Conference on Neural Information Processing Systems (Cambridge, MA, USA, MIT Press, 2004.
[68] L. Gonzalez-Abril, F. Cuberos, F. Velasco, J. Ortega, Ameva: An autonomous discretization algorithm, Expert Syst. Appl. 36 (3) (2009) 5327–5332.
[69] Google.Introduction to machine learning, 2020.
[70] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 855–868.
[71] F.E. Grubbs, Procedures for detecting outlying observations in samples, Technometrics 11 (1) (1969).
[72] A. Gupta, K.G. Mehrotra, C. Mohan, A clustering-based discretization for supervised learning, Stat. Probab. Lett. 80 (9-10) (2010) 816–824.
[73] V. Gupta, R. Hewett, Adaptive normalization in streaming data. Proceedings of the 2019 3rd International Conference on Big Data Research, ACM, 2019.
[74] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003).
[75] S.J. Hadeed, M.K. O'Rourke, J.L. Burgess, R.B. Harris, R.A. Canales, Imputation methods for addressing missing data in short-term monitoring of air pollutants, Sci. Total Environ. 730 (2020) 139140.
[76] Hall, M.A. Correlation-based feature selection for machine learning, 1999.
[77] J. Hamilton-Paterson, Time Series Analysis, Princeton Univers. Press, 1994.
[78] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 2011, Morgan Kaufmann, 2012.
[79] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, Elsevier LTD, Oxford, 2017.
[80] S. Hariri, M.C. Kind, R.J. Brunner, Extended isolation forest, IEEE Trans. Knowl. Data Eng. (2019) 1.
[81] B. Havers, R. Duvignau, H. Najdataei, V. Gulisano, A.C. Koppisetty, M. Papatriantafilou, DRIVEN: a framework for efficient data retrieval and clustering in vehicular networks. 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019.
[82] B. Havers, R. Duvignau, H. Najdataei, V. Gulisano, M. Papatriantafilou, A. C. Koppisetty, DRIVEN: A framework for efficient data retrieval and clustering in vehicular networks, Future Gener. Comput. Syst. 107 (2020) 1–17.
[83] F. He, H. Yang, Y. Miao, R. Louis, A hybrid feature selection method based on genetic algorithm and information gain. 2016 5th International Conference on Computer Science and Network Technology (ICCSNT), IEEE, 2016.
[84] X. He, D. Cai, P. Niyogi, Laplacian score for feature selection, NIPS'05. Proceedings of the 18th International Conference on Neural Information Processing Systems (Cambridge, MA, USA, MIT Press, 2005, pp. 507–514. NIPS'05.
[85] F. Hermans, N. Dziengel, J. Schiller, Quality estimation based data fusion in wireless sensor networks. 2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, IEEE, 2009.
[86] D.T. Hoang, H.J. Kang, A bearing fault diagnosis method using transfer learning and Dempster–Shafer evidence theory. Proceedings of the 2019 International Conference on Artificial Intelligence, Robotics and Control, ACM, 2019.
[87] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[88] V. Hodge, S. O'Keefe, J. Austin, Hadoop neural network for parallel and distributed feature selection, Neural Netw. 78 (2016).
[89] S. Hong, H.S. Lynn, Accuracy of random-forest-based imputation of missing data in the presence of non-normality, non-linearity, and interaction, BMC Med. Res. Methodol. 20 (1) (2020).

[90] W.H. Hsu, Genetic wrappers for feature selection in decision tree induction and variable ordering in bayesian network structure learning, Inf. Sci. 163 (1-3) (2004) 103–122.
[91] Chen, H., Li, J., and Mohapatra, P. Race: time series compression with rate adaptivity and error bound for sensor networks.In: 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE Cat. No.04EX975) (2004), 124-133.
[92] Y. Huang, M. Milani, F. Chiang, PACAS: Privacy-aware, data cleaning-as-a-service. 2018 IEEE International Conference on Big Data (Big Data), IEEE, 2018.
[93] A.J. Hussain, A. Al-Fayadh, N. Radi, Image compression techniques: A survey in lossless and lossy algorithms, Neurocomputing 300 (2018) 44–69.
[94] M. Isenburg, Laszip: lossless compression of lidar data, Photogramm. Eng. Remote Sens. 79 (2013) 2.
[95] J.-H. Jang, J. Choi, H.W. Roh, S.J. Son, C.H. Hong, E.Y. Kim, T.Y. Kim, D. Yoon, Deep learning approach for imputation of missing values in actigraphy data: Algorithm development study, JMIR mHealth uHealth 8 (7) (2020).
[96] Jayaratne, M., Alahakoon, D., De Silva, D., and Yu, X. Apache spark based distributed self-organizing map algorithm for sensor data analysis.In: IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society (2017), 8343-8349.
[97] A.R. Jones, Tails of the unexpected (2): Outing the outliers. Probability, Statistics and Other Frightening Stuff, Routledge, 2018, pp. 392–441.
[98] A. Jovic, K. Brkic, N. Bogunovic, A review of feature selection methods with applications. 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2015.
[99] S. Julier, J. Uhlmann, A new extension of the kalman filter to nonlinear systems, Proc. AeroSense Symp. (1997) 54–65.
[100] A. Karkouch, H. Mousannif, H.A. Moatassime, T. Noel, Data quality in internet of things: A state-of-the-art survey, J. Netw. Comput. Appl. 73 (2016) 57–81.
[101] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases, Knowl. Inf. Syst. 3 (3) (2001) 263–286.
[102] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. Locally adaptive dimensionality reduction for indexing large time series databases.In: Proceedings of the 2001 ACM SIGMOD international conference on Management of data (2001), 151-162.
[103] R. Kerber, Chimerge: Discretization of numeric attributes, AAAI'92. Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press, 1992, pp. 123–128. AAAI'92.
[104] B. Khaleghi, A. Khamis, F.O. Karray, S.N. Razavi, Multisensor data fusion: A review of the state-of-the-art, Inf. Fusion 14 (1) (2013).
[105] M. Khayati, A. Lerner, Z. Tymchenko, P. Cudré-Mauroux, Mind the gap, Proc. VLDB Endow. 13 (5) (2020) 768–782.
[106] K. Kira, L.A. Rendell, The feature selection problem: Traditional methods and a new algorithm, AAAI'92. Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press, 1992, pp. 129–134. AAAI'92.
[107] K. Kirchner, J. Zec, B. Delibašić, Facilitating data preprocessing by a generic framework: a proposal for clustering, Artif. Intell. Rev. 45 (3) (2015) 271–297.
[108] T. Kohonen, The self-organizing map, Proc. IEEE 78 (9) (1990) 1464–1480.
[109] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, Data preprocessing for supervised learning, Int. J. Comput. Sci. 1 (2006) 111–117.
[110] Krishnan, S., Franklin, M.J., Goldberg, K., Wang, J., and Wu, E. Activeclean: An interactive data cleaning framework for modern machine learning.In: Proceedings of the 2016 International Conference on Management of Data (New York, NY, USA, 2016), SIGMOD '16, Association for Computing Machinery, 2117-2120.
[111] L. Kurgan, K. Cios, CAIM discretization algorithm, IEEE Trans. Knowl. Data Eng. 16 (2) (2004) 145–153.
[112] P. Lanzi, Fast feature selection with genetic algorithms: a filter approach. Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97), IEEE, 1997.
[113] M.G. Larson, Analysis of variance, Circulation 117 (1) (2008) 115–121.
[114] R. Leardi, R. Boggia, M. Terrile, Genetic algorithms as a strategy for feature selection, J. Chemom. 6 (5) (1992) 267–281.
[115] C. Leys, C. Ley, O. Klein, P. Bernard, L. Licata, Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median, J. Exp. Soc. Psychol. 49 (4) (2013) 764–766.
[116] H. Li, K. Ota, M. Dong, Learning IoT in edge: Deep learning for the internet of things with edge computing, IEEE Netw. 32 (1) (2018) 96–101.
[117] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, ACM Comput. Surv. 50 (6) (2018) 1–45.
[118] D. Lin, X. Tang, Conditional infomax learning: An integrated framework for feature extraction and fusion. Computer Vision – ECCV 2006 (Berlin, Heidelberg, Springer, Berlin Heidelberg, 2006, pp. 68–82.
[119] J. Lin, E. Keogh, S. Lonardi, B. Chiu, A symbolic representation of time series, with implications for streaming algorithms, Data Min. Knowl. Discov. (2003) 2.
[120] Lin, J., Keogh, E., Lonardi, S., and Chiu, B. A symbolic representation of time series, with implications for streaming algorithms.In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (2003), 2-11.
[121] P. Lindstrom, M. Isenburg, Fast and efficient compression of floating-point data, IEEE Trans. Vis. Comput. Graph. 12 (5) (2006) 1245–1250.
[122] R.B. Litterman, Forecasting with bayesian vector autoregressions: Five years of experience, J. Bus. Econ. Stat. 4 (1) (1986) 25.
[123] Liu, F.T., Ting, K.M., and Zhou, Z.-H. Isolation forest.In: 2008 Eighth IEEE International Conference on Data Mining(2008), 413-422.
[124] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, CM Trans. Knowl. Discov. Data 6 (1) (2012) 1–39.
[125] H. Liu, H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, Springer, US, 1998.
[126] H. Liu, R. Setiono, Chi2: feature selection and discretization of numeric attributes, in: Anon (Ed.), Proceedings of the International Conference on Tools with Artificial Intelligence, IEEE, 1995, pp. 388–391.
[127] Y. Liu, T. Dillon, W. Yu, W. Rahayu, F. Mostafa, Missing value imputation for industrial IoT sensor data with large gaps, IEEE Internet Things J. 7 (8) (2020) 6855–6867.
[128] Lu, T., Xia, W., Zou, X., and Xia, Q. Adaptively compressing IoT data on the resource-constrained edge.In: 3rd {*USENIX*} Workshop on Hot Topics in Edge Computing (HotEdge 20) (2020).
[129] G. Luo, K. Yi, S.-W. Cheng, Z. Li, W. Fan, C. He, Y. Mu, Piecewise linear approximation of streaming time series data with max-error guarantees. 2015 IEEE 31st international conference on data engineering, IEEE,, 2015.
[130] J. Ma, J.C. Cheng, F. Jiang, W. Chen, M. Wang, C. Zhai, A bi-directional missing data imputation scheme based on lstm and transfer learning for building energy data, Energy Build. 216 (2020) 109941.
[131] S.T. Mai, I. Assent, M. Storgaard, AnyDBC. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016.
[132] Mao, Y., Zhang, Z., and Fan, D. Hybrid feature selection based on improved genetic algorithm for stock prediction.In: 2016 6th International Conference on Digital Home (ICDH) (2016), IEEE.
[133] Märgner, V., and Abed, H.E.ICDAR 2009 arabic handwriting recognition competition.In: 2009 10th International Conference on Document Analysis and Recognition(2009), IEEE.
[134] A.R. Martel, The detection of outliers in nondestructive integrations with the generalized extreme studentized deviate test, Publ. Astron. Soc. Pac. (2015).
[135] A. Martinez, A. Kak, PCA versus LDA, IEEE Trans. Pattern Anal. Mach. Intell. 23 (2) (2001) 228–233.
[136] J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction. Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 2011, pp. 52–59.
[137] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A survey on bias and fairness in machine learning, ACM Comput. Surv. 54 (6) (2021) 1–35.
[138] H.B. Mitchell, Data Fusion: Concepts and Ideas, Springer, Berlin Heidelberg, 2012.
[139] B. Mohebali, A. Tahmassebi, A.H. Gandomi, A. Meyer-Bäse, A big data inspired preprocessing scheme for bandwidth use optimization in smart cities applications using raspberry pi, in: F. Ahmad (Ed.), Big Data: Learning, Analytics, and Applications, SPIE, 2019.
[140] L. Morán-Fernández, V. Bolón-Canedo, A. Alonso-Betanzos, Centralized vs. distributed feature selection methods based on data complexity measures, Know. -Based Syst. 117 (C) (2017).
[141] Moreno, H. The importance of data quality - good, bad or ugly, 2017.
[142] T.P. Morris, I.R. White, P. Royston, Tuning multiple imputation by predictive mean matching and local residual draws, BMC Med. Res. Methodol. 14 (1) (2014).
[143] K.P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.
[144] E.F. Nakamura, A.A.F. Loureiro, A.C. Frery, Information fusion for wireless sensor networks, ACM Comput. Surv. 39 (3) (2007) 9.
[145] J. Nystrom-Persson, G. Keeble-Gagnére, N. Zawad, Compact and evenly distributed k-mer binning for genomic sequences, Bioinformatics 37 (17) (2021).
[146] Ogasawara, E., Martinez, L.C., de Oliveira, D., Zimbrao, G., Pap, G.L., and Mattoso, M. Adaptive normalization: A novel data normalization approach for non-stationary time series.In: The 2010 International Joint Conference on Neural Networks (IJCNN) (2010), IEEE.
[147] J.-R. Ohm, G.J. Sullivan, H. Schwarz, T.K. Tan, T. Wiegand, Comparison of the coding efficiency of video coding standards–including high efficiency video coding (hevc), IEEE Trans. Circuits Syst. Video Technol. 22 (12) (2012) 1669–1684.
[148] Olfati-Saber, R. Distributed kalman filtering for sensor networks.In: 2007 46th IEEE Conference on Decision and Control(2007).
[149] D.J. Olive, A resistant estimator of multivariate location and dispersion, Comput. Stat. Data Anal. 46 (1) (2004) 93–102.
[150] D. Palyvos-Giannas, V. Gulisano, M. Papatriantafilou, Genealog: Fine-grained data streaming provenance in cyber-physical systems, Parallel Comput. 89 (2019) 102552.
[151] D. Palyvos-Giannas, B. Havers, M. Papatriantafilou, V. Gulisano, Ananke: a streaming framework for live forward provenance, Proc. VLDB Endow. 14 (3) (2020) 391–403.
[152] J. Paparrizos, L. Gravano, K-shape: Efficient and accurate clustering of time series, SIGMOD Rec. 45 (1) (2016).
[153] Pekhimenko, G., Guo, C., Jeon, M., Huang, P., and Zhou, L. Tersecades: Efficient data compression in stream processing.In: 2018 USENIX Annual Technical Conference(2018), 307-320.
[154] G. Pekhimenko, V. Seshadri, O. Mutlu, M.A. Kozuch, P.B. Gibbons, T.C. Mowry, Base-delta-immediate compression: Practical data compression for on-chip caches. 2012 21st international conference on parallel architectures and compilation techniques (PACT), IEEE, 2012, pp. 377–388.

[155] T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, K. Veeraraghavan, Gorilla: A fast, scalable, in-memory time series database, Proc. VLDB Endow. 8 (12) (2015) 1816–1827.
[156] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005).
[157] T.D. Pigott, A review of methods for missing data, Educ. Res. Eval. 7 (4) (2001) 353–383.
[158] Popivanov, I., and Miller, R.J. Similarity search over time-series data using wavelets.In: Proceedings 18th International Conference on Data Engineering (2002), 212-221.
[159] Psaras, I., Ascigil, O., Rene, S., Pavlou, G., Afanasyev, A., and Zhang, L. Mobile data repositories at the edge.In: Workshop on Hot Topics in Edge Computing (HotEdge 18) (2018).
[160] C. Rabbath, D. Corriveau, A comparison of piecewise cubic hermite interpolating polynomials, cubic splines and piecewise linear functions for the approximation of projectile aerodynamics, Def. Technol. 15 (5) (2019).
[161] P. Ratanaworabhan, J. Ke, M. Burtscher, Fast lossless compression of scientific floating-point data. Data Compression Conference (DCC'06), IEEE, 2006, pp. 133–142.
[162] Richardson, M. Principal component analysis, 2009.
[163] Riffenburgh, R.H. Linear discriminant analysis, 1957.
[164] M. Robnik-Sikonja, I. Kononenko, An adaptation of relief for attribute estimation in regression, ICML '97. Proceedings of the Fourteenth International Conference on Machine Learning (San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 1997, pp. 296–304. ICML '97.
[165] M. Robnik-Šikonja, I. Kononenko, Theoretical and empirical analysis of relieff and rrelieff, Mach. Learn. 53 (2003).
[166] P.J. Rousseeuw, C. Croux, Alternatives to the median absolute deviation, J. Am. Stat. Assoc. 88 (424) (1993).
[167] S. Ruan, R. Li, J. Bao, T. He, Y. Zheng, CloudTP: A cloud-based flexible trajectory preprocessing framework. 2018 IEEE 34th International Conference on Data Engineering (ICDE), IEEE, 2018.
[168] Rubin, D.B. Multiple imputations in sample surveys - a phenomenological bayesian approach to nonresponse.In: Proceedings of the Survey Research Methods Section of the American Statistical Association, 20-34.
[169] R. Salles, K. Belloze, F. Porto, P.H. Gonzalez, E. Ogasawara, Nonstationary time series transformation methods: An experimental review, Knowl. -Based Syst. 164 (2019).
[170] Y. Sazeides, J.E. Smith, The predictability of data values. Proceedings of 30th Annual International Symposium on Microarchitecture, IEEE, 1997, pp. 248–258.
[171] T. Schneider, Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values, J. Clim. (2001).
[172] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.
[173] E. Schubert, J. Sander, M. Ester, H.P. Kriegel, X. Xu, DBSCAN revisited, revisited, ACM Trans. Database Syst. 42 (3) (2017) 1–21.
[174] Seltman, H.J. One-way ANOVA 2018, ch. Chapter 7.
[175] Shafer, G. Dempster-shafer theory.(http://www.glennshafer.com/assets/downloads/articles/article48.pdf) (2002).
[176] W. Song, C. Gao, Y. Zhao, Y. Zhao, A time series data filling method based on LSTM–taking the stem moisture as an example, Sensors 20 (18) (2020) 5045.
[177] C. Stachniss, Particle filters for robot navigation, Found. Trends Robot. 3 (4) (2012) 211–282.
[178] Staudemeyer, R.C., and Morris, E.R. Understanding lstm – a tutorial into long short-term memory recurrent neural networks.2019.
[179] D.J. Stekhoven, P. Buhlmann, MissForest–non-parametric missing value imputation for mixed-type data, Bioinformatics 28 (1) (2011) 112–118.
[180] Sterne, J.A.C., White, I.R., Carlin, J.B., Spratt, M., Royston, P., Kenward, M.G., Wood, A.M., and Carpenter, J.R. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls.2009.
[181] P. Tadić, Ž. Durović, Particle filtering for sensor fault diagnosis and identification in nonlinear plants, J. Process Control 24 (4) (2014) 401–409.
[182] Talebi, S.P., and Werner, S. Distributed kalman filtering: Consensus, diffusion, and mixed.In: 2018 IEEE Conference on Control Technology and Applications (CCTA) (2018).
[183] Tawakuli, A. Transforming Data Preprocessing: A Holistic, Normalized and Distributed Approach.PhD thesis, 2022.
[184] Tawakuli, A., Kaiser, D., and Engel, T. Modern data preprocessing is holistic, normalized and distributed.2022.
[185] A. Tawakuli, D. Kaiser, T. Engel, Synchronized preprocessing of sensor data. 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 3522–3531.
[186] A. Tawakuli, D. Kaiser, T. Engel, Experience: Differentiating between isolated and sequence missing data, ACM J. Data Inf. Qual. 14 (3) (2022).
[187] D.M. Tax, R.P. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[188] H.Y. Teh, A.W. Kempa-Liehr, K.I.-K. Wang, Sensor data quality: a systematic review, J. Big Data 7 (1) (2020).
[189] C.-J. Tsai, C.-I. Lee, W.-P. Yang, A discretization algorithm based on class-attribute contingency coefficient, Inf. Sci. 178 (3) (2008) 714–731.
[190] R.J. Urbanowicz, M. Meeker, W.L. Cava, R.S. Olson, J.H. Moore, Relief-based feature selection: Introduction and review, J. Biomed. Inform. 85 (2018) 189–203.
[191] Vannucci, M., and Colla, V. Meaningful discretization of continuous features for association rules mining by means of a som.In: Proceedings12th European Symposium on Artificial Neural Networks ESANN2004 (2004).
[192] C. Velasco-Gallego, I. Lazakis, Real-time data-driven missing data imputation for short-term sensor data of marine systems. a comparative study, Ocean Eng. 218 (2020) 108261.
[193] S.D. Vito, E. Massera, M. Piga, L. Martinotto, G.D. Francia, On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario, Sens. Actuators B: Chem. 129 (2) (2008).
[194] Wan, E., and Merwe, R.V.D. The unscented Kalman filter for nonlinear estimation. In: Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373) (2000), IEEE.
[195] C. Wang, J. Caja, E. Gómez, Comparison of methods for outlier identification in surface characterization, Measurement 117 (2018) 312–325.
[196] Wang, L. Research on distributed parallel dimensionality reduction algorithm based on pca algorithm.In: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)(2019).
[197] T. Wang, H. Ke, A. Jolfaei, S. Wen, M.S. Haghighi, S. Huang, Missing value filling based on the collaboration of cloud and edge in artificial intelligence of things, IEEE Trans. Ind. Inform. 18 (8) (2022).
[198] Wang, X., and Chen, J. Distributed principal component analysis based on randomized low-rank approximation.In: 2020 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)(2020).
[199] Z.-M. Wang, G.-H. Song, C. Gao, An isolation-based distributed outlier detection framework using nearest neighbor ensembles for wireless sensor networks, IEEE Access 7 (2019).
[200] Wu, H., Siegel, M., Stiefelhagen, R., and Yang, J. Sensor fusion using dempster-shafer theory [for context-aware HCI].In: Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference (2002), IEEE.
[201] W. Wu, Y. Mallet, B. Walczak, W. Penninckx, D. Massart, S. Heuerding, F. Erni, Comparison of regularized discriminant analysis linear discriminant analysis and quadratic discriminant analysis applied to NIR data, Anal. Chim. Acta 329 (3) (1996) 257–265.
[202] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, K. Łukasz, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean, Google's neural machine translation system: Bridging the gap between human and machine translation, CoRR. abs/1609 08144 (2016).
[203] Xing, R. The Compression of IoT operational data time series in vehicle embedded systems. PhD thesis, 2018.
[204] B. Xue, M. Zhang, W.N. Browne, Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms, Appl. Soft Comput. 18 (2014) 261–276.
[205] Yan, Y., Cao, L., and Rundensteiner, E.A. Scalable top-n local outlier detection.In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2017), ACM.
[206] Yang, H.H., and Moody, J. Feature selection based on joint mutual information.In: In Proceedings of International ICSC Symposium on Advances in Intelligent Data Analysis (1999), 22-25.
[207] Yang, K., Kpotufe, S., and Feamster, N. An efficient one-class svm for anomaly detection in the internet of things.
[208] K. Yang, T. Xing, Y. Liu, Z. Li, X. Gong, X. Chen, D. Fang, cDeepArch: A compact deep neural network architecture for mobile sensing, IEEE/ACM Trans. Netw. 27 (5) (2019) 2043–2055.
[209] W. Yang, K. Wang, W. Zuo, Neighborhood component feature selection for high-dimensional data, J. Comput. 7 (1) (2012).
[210] Yi, B.-K., and Faloutsos, C. Fast time sequence indexing for arbitrary lp norms.In: Proceedings of the 26th International Conference on Very Large Data Bases (San Francisco, CA, USA, 2000), VLDB '00, Morgan Kaufmann Publishers Inc.
[211] Zahedi, S., Szczodrak, M., Ji, P., Mylaraswamy, D., Srivastava, M., and Young, R. Tiered architecture for on-line detection, isolation and repair of faults in wireless sensor networks.In: MILCOM 2008 - 2008 IEEE Military Communications Conference (2008), IEEE.
[212] Y. Zhang, N. Meratnia, P.J. Havinga, Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine, Ad Hoc Netw. 11 (3) (2013) 1062–1074.
[213] Zhao, Z., Anand, R., and Wang, M. Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform. In: 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA) (2019), IEEE.
[214] X. Zheng, M. Wang, J. Ordieres-Meré, Comparison of data preprocessing approaches for applying deep learning to human activity recognition in the context of industry 4.0, Sensors 18 (7) (2018) 2146.
[215] Zhou, J., and Huang, Z. Recover missing sensor data with iterative imputing network.
[216] Zhou, Z., Wang, Y., and Li, M. Feature selection method based on hybrid SA-GA and random forests.In: 2020 International Conference on Computing and Data Science (CDS)(2020), IEEE.
[217] Zhu, M., and Shi, H. A novel support vector machine algorithm for missing data. In: Proceedings of the 2nd International Conference on Innovation in Artificial Intelligence - ICIAI '18 (2018), ACM Press.
[218] R. Zhu, T. Yu, Z. Tan, W. Du, L. Zhao, J. Li, X. Xia, Ptaod: A novel framework for supporting approximate outlier detection over streaming data for edge computing, IEEE Access 8 (2020).