Edge-based Data Profiling and Repair as a Service for IoT

Simeon Tverdal, Arda Goknil, Phu Nguyen, Erik Johannes Husom, Sagar Sen
SINTEF
Oslo, Norway
firstname.lastname@sintef.no

Jan Ruh, Francesca Flamigni
TTTech Computertechnik AG
Vienna, Austria
firstname.lastname@tttech.com

ABSTRACT

With the proliferation of IoT devices and the consequent exponential growth in data generation, ensuring data quality has become a critical challenge in IoT applications. Erroneous data can significantly impact the reliability and effectiveness of decision-making processes and downstream analytics. Leveraging the computational abilities of edge devices enables data profiling and repair tasks at the edge, allowing for immediate remediation of erroneous data within the data stream and improved scalability through distributed repair across multiple edge devices. Cloud-based data profiling and repair methods have been extensively researched, but limited computational resources constrain their applicability at edge/fog devices. To overcome this limitation and enhance generalizability, Machine Learning (ML) offers a promising solution, allowing sensor substitution, missing value prediction, and corrupt data replacement. ML-based data repair techniques can be flexibly deployed at the edge using containerized repair services for real-time data repair. In this paper, we propose and assess EDPRaaS (Edge-based Data Profiling and Repair as a Service), a novel approach designed for efficient data quality profiling and repair in IoT environments. EDPRaaS incorporates an ML-based data repair component, enabling real-time data repair at the edge. It leverages pandas profiling and Great Expectations tools for data profiling, providing comprehensive insights into the dataset and detecting data quality issues.

KEYWORDS

Edge computing, data quality, data profiling, data repair

ACM Reference Format:
Simeon Tverdal, Arda Goknil, Phu Nguyen, Erik Johannes Husom, Sagar Sen and Jan Ruh, Francesca Flamigni. 2023. Edge-based Data Profiling and Repair as a Service for IoT. In *The International Conference on the Internet of Things (IoT 2023), November 07–10, 2023, Nagoya, Japan.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3627050.3627065

1 INTRODUCTION

The Internet of Things (IoT) has experienced a remarkable expansion in recent years, resulting in a proliferation of interconnected devices that generate vast volumes of data. However, the quality of data collected from IoT devices is often compromised due to various factors such as sensor inaccuracies, network latency, and environmental conditions [10, 13, 30]. Data quality issues, including incomplete or inconsistent data, can significantly impact the reliability and effectiveness of IoT applications and services [7, 22]. Therefore, guaranteeing the reliability and accuracy of IoT data becomes imperative for facilitating trustworthy decision-making processes and extracting valuable insights within the IoT domain.

Two critical and complementing data quality management techniques for IoT systems are data profiling and repairing, which involve checking data for data quality requirements and repairing data for detected data quality problems. Traditional approaches to data profiling and repair [15, 17, 33, 34] often rely on centralized architectures, where all data processing tasks are performed in the cloud. However, this centralized approach poses challenges concerning network bandwidth, latency, scalability, privacy, and security. For instance, cloud-based data profiling and repair solutions rely on the availability and reliability of cloud services. Any disruptions or downtime in the cloud infrastructure can impact the ability to perform real-time or near-real-time data profiling and repair tasks. On the other hand, as the number of IoT devices and the volume of data grow, cloud-based data profiling and repair systems need to scale accordingly to handle the increasing workload. Scaling cloud resources can be costly as it requires provisioning and maintaining additional infrastructure to handle the processing demands.

Edge computing emerged as a promising paradigm for processing data closer to the data source, reducing latency, bandwidth requirements, and reliance on centralized cloud resources. It opens up new possibilities for performing data profiling and repair directly at the edge. For instance, data repair at the edge involves immediate remediation of erroneous data within the data stream. The distributed data repair across multiple edge devices or gateways also enables the workload to be distributed, improving scalability.

Although considerable research has been devoted to devising and employing data profiling and repair techniques on the cloud [6, 15, 17, 29, 34, 36], only a few approaches [19, 23, 26, 27] detect and repair "corrupt" data at edge/fog devices near the data source, where computational resources are scarce. However, these approaches have specific constraints that limit their applicability and generalizability in IoT applications [7]. For instance, Lin et al. [19] require all dependent data computations in the application state history, which are not always available. To overcome such limitations, Machine Learning (ML) offers a promising solution that can be combined with existing data profiling techniques. ML models can learn correlations among data sources (sensors), enabling the substitution of sensors, prediction of missing values, and generation of new data to replace corrupt data. ML-based data repair techniques can be deployed at the edge or in the cloud, leveraging available training data to create containerized repair services for real-time data repair.

IoT 2023, November 07–10, 2023, Nagoya, Japan
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0854-1/23/11.
https://doi.org/10.1145/3627050.3627065

In this paper, we propose, apply, and assess our approach (EDPRaaS) for Edge-based Data Profiling and Repair as a Service for IoT. EDPRaaS provides scalable and efficient data profiling and repair capabilities at the edge of IoT networks. It leverages pandas profiling [25] and Great Expectations [8] tools for data profiling tasks. It utilizes the Nerve Edge platform [21] as the runtime environment for executing the data profiling and repair service. The data repair component of EDPRaaS involves training an ML model on the cloud, which is then deployed to the edge for real-time data repair. The ML-based data repair component of EDPRaaS can effectively handle complex patterns and relationships within the data. ML models can learn from existing data patterns and make predictions or corrections for erroneous or missing data. This ML-driven approach allows for automated and adaptive data repair, reducing manual intervention and enabling scalability.

EDPRaaS utilizes pandas profiling [25] and Great Expectations [8] for data profiling, leveraging their complementary capabilities in assessing data quality. Pandas profiling provides comprehensive insights with descriptive statistics and visualizations, generating user-friendly reports summarizing data quality problems in the data stream. In contrast, Great Expectations enables finer control with custom assertions, detecting data inconsistencies, duplicate records, data range violations, missing values, and schema discrepancies. The data repair component of EDPRaaS employs the output of Great Expectations as input for data repair operations; pandas profiling generates reports on data quality problems for end-users.

Edge servers such as IBM Edge computing [11] and Nerve Edge servers [21] are getting more powerful and can be offered as a service. We worked with the Nerve Edge platform, which has industrial environment compatibility, essential infrastructure, support for diverse communication protocols, and Docker containerization. The platform's seamless integration with custom software components and efficiency make the Nerve Edge platform a suitable choice for deploying EDPRaaS at the edge.

We evaluated EDPRaaS using various ML models/architectures on three industrial IoT datasets. These ML models include lightweight models, i.e., XGBoost (XBG) and Stochastic Gradient Descent (SGD), and deep learning models, i.e., Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). The evaluation results demonstrate the potential benefits of leveraging simpler ML models like XGBoost for improved reliability in some scenarios. Furthermore, the findings indicate that edge-based inference with moderately sophisticated models could be feasible, considering the model training is performed in the cloud.

The contributions of this work can be summarized as follows: (i) the development of an edge-based framework that provides data profiling and repair as a service for IoT, (ii) the design and implementation of an ML-based data repair technique for the edge, (iii) the evaluation and analysis of the proposed framework's performance and efficiency in improving data quality for IoT applications.

2 BACKGROUND

2.1 Data Quality for IoT

Data quality is defined in ISO/IEC 25012:2008 [12] as *a degree to which the characteristics of data satisfy stated and implied needs when used under specified conditions.* **Data quality metrics** are the measurements by which you assess your data. They benchmark how complete, valid, accurate, timely, and consistent the data is and help differentiate between high-quality and low-quality data. They can be obtained from **data quality dimensions**, i.e., the measurement attributes of data, which we can assess and improve. Data accuracy and completeness are two data quality dimensions addressed by quality metrics. **Data completeness** refers to the degree to which all parts of the data are given with no missing information [35]; **data accuracy** is the degree of similarity of a measured quantity to its actual value.

Data quality requirements describe the needs or conditions that high-quality data should meet. They are checked on the input data to compute the corresponding metrics. The data quality requirement violation indicates a data quality problem/issue. Figure 1 shows some data quality problems on time-series data. **Missing data** refers to cases when a variable or attribute has no value. **Outliers** are extreme values that deviate from other observations of data. **Duplicated records** are two or more adjacent data points in the same timestamp. A **sudden rate of change** refers to cases where a variable changes suddenly (unrealistically) over a specific period of time.

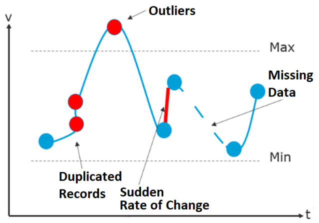

Figure 1: Example quality problems on time-series data.

Data quality management techniques (in short, data quality techniques) improve and maintain data quality across system components. There are three types of data quality techniques:

- ***Data Profiling*:** Data is monitored to check **data quality requirements** for detecting quality issues such as outliers.
- ***Data Cleaning*:** It entails the removal of corrupt and unusable data, e.g., those affected by environmental noise or extreme operating conditions such as high temperature.
- ***Data Repair*:** It restores data that has been lost, accidentally deleted, corrupted, or made inaccessible, e.g., by using simulation data or data from redundant sources (other sensors).

Data quality techniques can be **online** (real-time at or close to the data source) and **offline** (for large historical datasets on the cloud) [14, 28]. Our approach provides an online data profiling and repair service at the edge. More specifically, EDPRaaS is primarily categorized as a data repair solution rather than a data cleaning solution because of its focus on correcting erroneous or corrupt data within a constant data stream. In contrast, data cleaning involves only removing corrupt data.

As part of data profiling, we utilize the Great Expectations (GE) framework [8], an open-source solution designed to support engineers in defining expectations (assertions) regarding data quality. GE facilitates the generation of code that enables data analysis and the creation of interactive hypertext data documents. These documents serve the purpose of profiling the data and evaluating its quality by assessing the degree to which expectations are met. Specifically, they prove invaluable for auditing the acquisition and persistence of data from industrial processes. Additionally, they play a crucial role in fostering a culture of data quality, instilling confidence among engineers in their data-driven decision-making processes, and providing estimates of associated uncertainties.

Another data profiling solution is pandas-profiling [25], an open-source Python library that automates the process of creating exploratory data analysis reports. It is built on top of the pandas library [24], widely used for data manipulation and analysis in Python. Pandas-profiling generates reports with descriptive statistics, interactive visualizations, and insights on variables, correlations, missing values, data types, interactions, and more.

2.2 Edge Computing and Nerve Platform

Traditionally, cloud computing has been the dominant model to process and store data. However, cloud-centric architectures face challenges concerning latency, bandwidth limitations, privacy concerns, and reliance on centralized resources. These challenges are particularly relevant in the IoT realm, where data is generated and processed in real time and often requires rapid response and local decision-making. Edge computing addresses these challenges by placing computing resources (e.g., servers, storage, and analytic capabilities) closer to the network edge. Since edge servers such as IBM Edge computing [11] and Nerve Edge servers [21] are getting more powerful, they can be offered as a service and act as local data processing units performing computations and analysis.

Among various Edge platforms, the Nerve Edge platform [21], developed by TTTech Industrial [32], offers a software infrastructure for industrial automation. It seamlessly integrates with various commercial off-the-shelf hardware platforms, such as TTTech Industrial's Nerve-certified Intel Atom-based MFN 100 edge computing device [20]. The Nerve software infrastructure encompasses the local software stack of a Nerve node, incorporating a virtualization solution, as well as the cloud-hosted Nerve Management System.

The local software stack of a Nerve node employs hypervisor technology to virtualize the underlying hardware platform. Supported hypervisors are Linux KVM [16], ACRN [18], and Xen [1]. A node's virtualized architecture allows for the most flexibility in workload deployment supporting (i) virtual machines (VMs) for workloads requiring strict spatial isolation for security reasons, (ii) dedicated statically partitioned real-time VMs for workloads also requiring temporal isolation, such as (soft) real-time workloads, and (iii) Docker containers for workloads requiring less strict spatial and temporal isolation but more flexibility, faster startup times, and tighter development cycles. An administrator can register nodes in the web-based Nerve Management System, which offers users role-based access control to multiple web-based tools. These tools ease the administration of edge devices and enable the seamless deployment and management of workloads on several nodes regardless of their physical locations.

3 RELATED WORK

The literature on data profiling and repair techniques has seen significant contributions, with a focus on cloud-based implementations [6, 15, 17, 29, 34, 36]. However, only a limited number of approaches [19, 23, 26, 27] have specifically targeted detecting and repairing "corrupt" data at edge and fog devices near the data source, where computational resources are constrained. These edge-based solutions present promising opportunities but encounter distinct constraints that limit their applicability and generalizability in the context of IoT applications [7]. For instance, an edge-based approach proposed by Lin et al. [19] requires access to all dependent data computations in the application state history, which may not always be practically feasible in real-world scenarios. Russel et al. [26] propose a data repair solution enhancing the initial sensed data from cameras by incorporating raw data from ambient sensors at the edge. This approach employs sensory substitution to bolster the data's robustness, resilience, and dependability. However, it is tailored to data sourced from cameras and ambient sensors for motion detection, which may restrict its applicability to these particular sensor types.

ML can address these limitations of edge-based data repair solutions while enhancing data repair in IoT environments. ML-based data repair techniques can offer the flexibility of deployment options, allowing them to operate at the edge or in the cloud. However, only a few ML-based techniques support data repair for IoT [7]. Flick et al. [6] utilize ML algorithms, specifically K-means for clustering and regression modeling, to identify outliers within clusters. However, their approach does not involve predicting new values to replace the outliers. Instead, they employ overflow, overweight, substitution value, and algebraic sign calculations to compute the new values. Sen et al. [29] present an ML pipeline to train ML models that can step in for faulty sensors to maintain reasonable quality and continuity in sensor data streams. These two approaches [6, 29] are offline data repair solutions and do not consider ML-based data repair at the edge. Okafor et al. [23] propose an online ML-based solution for data repair that involves calibrating low-cost sensors to align their measurements with concentrations obtained from reference sensors. However, it does not explicitly address deployment and versioning scenarios of ML models at the edge. Sen et al. [31] propose an edge-cloud AI pipeline architecture where online data repair services can be offered to tame data quality. They do not devise or implement scenarios for edge-based data repair. EDPRaaS supports online data repair scenarios where ML models are trained and deployed based on the availability of training data. The ML models in EDPRaaS are containerized as online repair services and deployed on edge for real-time data repair.

4 EDPRAAS APPROACH: DATA PROFILING AND REPAIR AT THE EDGE

EDPRaaS is a real-time data profiling and repair solution designed for edge environments. Figure 2 illustrates the comprehensive EDPRaaS architecture, encompassing the IoT-Edge-Cloud computing continuum. EDPRaaS operates through a series of events and services, starting with acquiring (industrial) IoT sensor data via IoT gateways (Section 4.1), followed by profiling the data (Section 4.2), and then repairing data on the fly at the Edge (Section 4.3).

4.1 Data Acquisition at the Edge

Data acquisition is achieved through IoT gateways, such as the Danobat Smart Box [4], which act as universal interfaces to machine tools, sensors, and Programmable Logic Controllers (PLCs), ensuring secure data gathering and transmission. Standardized protocols such as MQTT or OPC UA [9] are employed to stream data to the Edge, with the Nerve platform offering Edge-based services

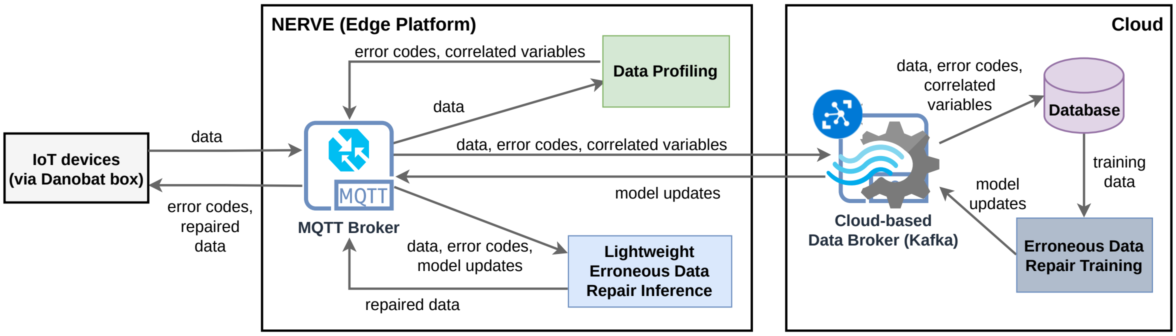
NERVE (Edge Platform)
error codes, correlated variables
Data Profiling
data
IoT devices (via Danobat box)
data
data, error codes, correlated variables
error codes, repaired data
MQTT
MQTT Broker
model updates
data, error codes, model updates
Lightweight Erroneous Data Repair Inference
repaired data
Cloud
data, error codes, correlated variables
Database
training data
model updates
Cloud-based Data Broker (Kafka)
Erroneous Data Repair Training

Figure 2: EDPRaaS architecture showcasing EDPRaaS steps from IoT devices to both edge and cloud platforms. An industrial IoT device, e.g., the Danobat Smart Box, acquires sensor data from machine tools, sensors, and PLCs. The data is then transmitted via an MQTT broker to the Nerve Edge Platform for data profiling and repair operations.

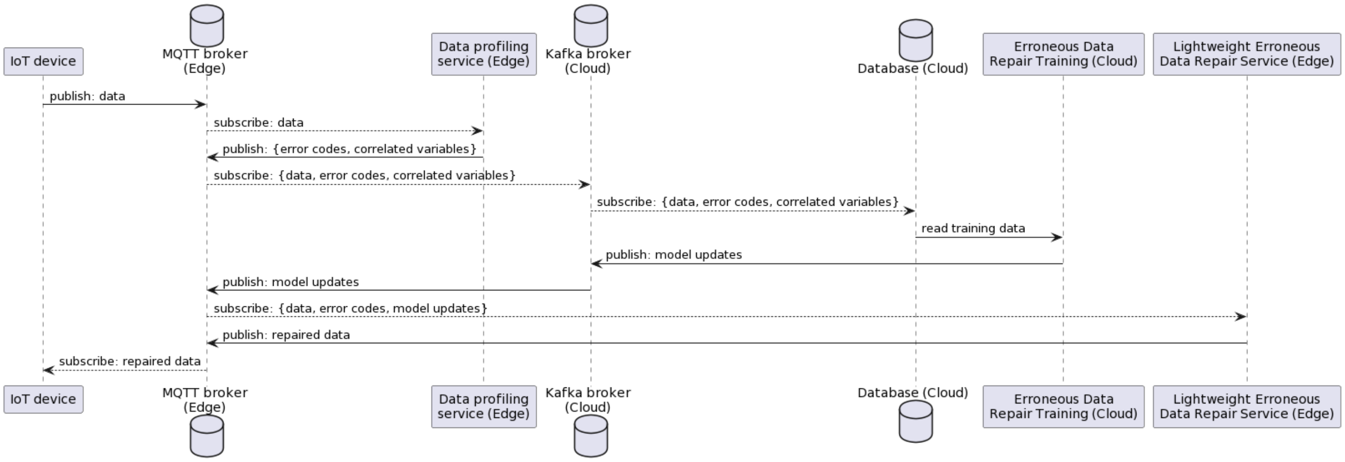
IoT device
MQTT broker (Edge)
Data profiling service (Edge)
Kafka broker (Cloud)
Database (Cloud)
Erroneous Data Repair Training (Cloud)
Lightweight Erroneous Data Repair Service (Edge)
publish: data
subscribe: data
publish: {error codes, correlated variables}
subscribe: {data, error codes, correlated variables}
subscribe: {data, error codes, correlated variables}
read training data
publish: model updates
publish: model updates
subscribe: {data, error codes, model updates}
publish: repaired data
subscribe: repaired data

Figure 3: The sequence diagram of data flow between Edge-Cloud services.

supporting these protocols. Figure 2 illustrates a representative MQTT broker facilitating efficient and reliable data communication between IoT devices and subsequent services. OPC UA is utilized when high-frequency data streaming is necessary. EDPRaaS implements a data profiling service and a lightweight erroneous data repair service leveraging ML to tackle data quality issues. Figure 3 presents a sequence diagram of EDPRaaS depicting the coordinated process of data profiling, repair, and cloud-based ML model training.

4.2 Data Profiling at the Edge

The data profiling service subscribes to live data streams and utilizes pandas profiling and Great Expectations to identify data quality issues in the streaming data. Great Expectations are used to assert the correctness of the data and validate the data quality (see an example expectation in Figure 5). Employing Great Expectations, EDPRaaS uncovers data quality issues, including missing values, outliers, and duplications, and provides an overall dataset assessment. Pandas Profiling enables exploratory data analysis, general warnings related to possible errors, and finding correlations between the variables in the dataset. This information is forwarded to the MQTT broker (see Figure 3).

4.3 Data Repair at the Edge

Model Training on the Cloud: To ensure high-quality data repair models, ML model training is conducted on a cloud platform like Amazon AWS or Microsoft Azure, offering ample scalability and computational resources. The cloud platform utilizes a scalable data broker (e.g., Apache Kafka) to efficiently handle data streams, supported by a robust database and an ML pipeline for training data repair models. These models are tailored to rectify erroneous or corrupt data within the streamed data.

The Kafka broker, deployed on a cloud platform, subscribes to data, error codes, and correlated variables published on the MQTT broker (see Figure 3). This information is stored in the cloud database. The Erroneous Data Repair Training pipeline utilizes this data, including error-free data, to train and validate one or more ML models (see Figure 4 for the details of the pipeline). These models are trained to replace erroneous values in variables based on the data profile generated during data acquisition. Variables correlated to the target sensor serve as input data, while error-free sensor values (before failure) are used as output for training and validation. A configuration file defines the window size of input and output variables, the train/validation/test split, and the criteria

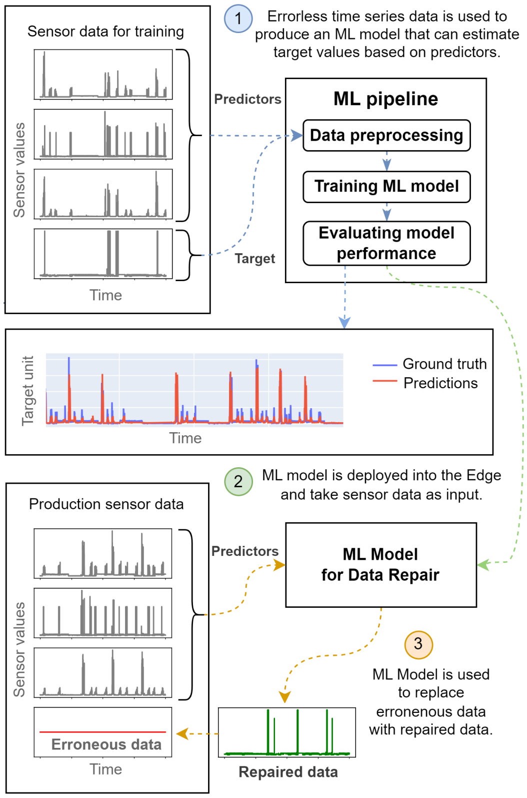
Sensor data for training
Sensor values
Time
1
Errorless time series data is used to produce an ML model that can estimate target values based on predictors.
Predictors
Target
ML pipeline
Data preprocessing
Training ML model
Evaluating model performance
Target unit
Time
Ground truth
Predictions
2
ML model is deployed into the Edge and take sensor data as input.
Production sensor data
Sensor values
Predictors
ML Model for Data Repair
3
ML Model is used to replace erronenous data with repaired data.
Erroneous data
Time
Repaired data

Figure 4: ML Pipeline and its ML Models for Data Repair.

for selecting representative data for training. Periodic updates to these ML models are published to the Kafka broker on the cloud and subsequently to the MQTT broker on the Nerve Edge.

The Lightweight Erroneous Data Repair Service, subscribed to new data, error codes, and model updates, receives the updated model from the MQTT broker on the Nerve Edge. When the validation module detects erroneous data through the error codes, it sends instructions to the repair module on how to batch the data for repair. The repair module utilizes the appropriate ML model from the updated model and applies it to the batched data, inferring new values for the affected variables. The repaired data is then published to the MQTT broker, replacing the original erroneous values. Cloud services or edge devices can now use the repaired data on the MQTT broker for further analysis or decision-making involving tasks such as predictive maintenance or tool wear prediction. IoT devices subscribing to the MQTT broker can also receive the repaired data to use in their operations.

Figure 5 illustrates the erroneous data repair process using sample data from a temperature sensor in a manufacturing setup. We employ Great Expectations to define the expected temperature range of 20 to 40 degrees Celsius. The first diagram represents the original sensor data, including a segment of erroneous values caused by environmental interference. Upon detecting anomalies, an error code is transmitted via the MQTT broker, triggering the data correction process using the data repair model. The second diagram displays the data after the repair process, demonstrating the successful restoration of accurate values.

```
{
        "expectation_type": "expect_column_values_to_be_between",
        "kwargs": {
                "column": "Temperature_X",
                "max_value": 40,
                "min_value": 20
        },
        "meta": {}
}
```

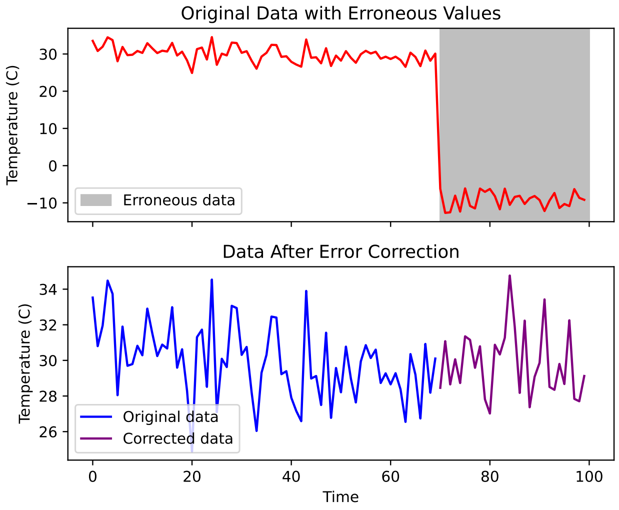

Figure 5: Temperature sensor data profiling and correction in a manufacturing environment. The JSON body in Great Expectations defines the expected data range for the sensor. The middle graph shows the original time series data, with erroneous measurements highlighted in grey due to environmental interference. The bottom graph displays the time series after erroneous data repair.

EDPRaaS offers several advantages. Firstly, it facilitates real-time detection and repair of erroneous sensor data by leveraging data from other sensors, ensuring uninterrupted functionality for analytics solutions reliant on the affected data. Secondly, since the ML model training occurs in the cloud, there is no need for resource-intensive training on the edge, reducing computational overhead at the edge devices. Lastly, the edge ML module can be enhanced to estimate uncertainties in the ML model or input data, providing valuable feedback to the ML pipeline in the cloud for continuous improvements. This iterative feedback loop can improve the overall performance of the data repair process.

5 EVALUATION

We evaluate our approach on three datasets from three distinct domains to address the following Research Questions (RQ)s:

- ***RQ1.*** *To what extent can EDPRaaS repair erroneous data in a constant data stream?* The goal of this research question is to assess the reliability and accuracy of EDPRaaS in maintaining data quality and ensuring the integrity of the data stream.
- ***RQ2.*** *Is EDPRaaS capable of timely response to erroneous data?* The goal of this research question is to evaluate how quickly and effectively EDPRaaS can perform repair actions for data quality issues, such as inconsistencies or missing values.

- ***RQ3.*** *How can EDPRaaS be deployed and run on the edge for industrial environments?* The goal of this research question is to provide insights into the technical considerations, challenges, and potential solutions for seamless integration and utilization of EDPRaaS in industrial environments.

5.1 Experiment Design

Datasets. Our evaluation used three Industrial IoT datasets (sensor time series data from three industrial manufacturing setups) and two different model configurations (light and heavy configurations). The first dataset (*FERSA dataset*) contains accelerometer data collected at a rate of 1 HZ. We used the raw data of one variable to predict the amplitude in the light model configuration. In the heavy one, we employed a lower window size and all variables as input with feature engineering. We trained the models on 100 MB of data.

The second dataset (*Automotive dataset*) is from the automotive domain and contains 14 variables collected at 10 HZ from the milling process of a cylinder head's combustion chamber. It includes spindle torque and position and power for each ax. In the light configuration, we used one variable to predict the target variables without feature engineering. We utilized all input variables and feature engineering in the heavy configuration.

The third dataset (*Aeromec dataset*) is from the aerospace domain and contains standard CNC machine data, including temperature measurements and high-frequency accelerometer data. It has over 200 variables with some data quality metrics such as SNR, jitter, and ski slope. In the light configuration, we predicted the amplitude of an accelerometer using its frequency, while we utilized all input variables and feature engineering in the heavy configuration.

Setup. EDPRaaS is assessed using the Nerve platform and runs on an MFN 100, i.e., a Nerve device optimized for use with the Nerve software [20]. The device is designed for harsh industrial environments (-40°C to +70°C). It is based on an Intel Atom x5-E3940/50 CPU and offers 4 GB/8 GB RAM and up to 512 GB SSD storage. The MFN 100 offers one I/O port for Ethernet-based fieldbus connectivity, four GbE switch ports, and one SFP port.

5.2 Results

This section discusses the results of our case studies, addressing, in turn, each of the RQs.

RQ1: To what extent can EDPRaaS repair erroneous data in a constant data stream? In response to RQ1, we investigated the performance of diverse ML architectures/models in EDPRaaS through case studies, comprising both lightweight, XGBoost (XBG) and Stochastic Gradient Descent (SGD), and deep learning models, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). We aimed to assess the trade-off between computational resources and performance, particularly in relatively simple cases where lightweight algorithms might suffice. The evaluation utilized the coefficient of determination (R^2 score) as a metric to gauge predictive accuracy, with an R^2 score of 1 representing perfect predictions and scores closer to 0 indicating lower accuracy. When comparing the performance of different models on the same dataset, we used mean squared error (MSE) to assess how well a model's predictions match the true observed values.

Table 1 displays the models' performance (R^2 and MSE scores) on our three distinct datasets. We selected two model configurations (light and heavy in Table 1) to investigate the impact of model complexity on performance and latency (see RQ2 for latency). The light configuration employs limited input variables, avoids feature engineering, and adopts a larger window size. Conversely, the heavy configuration incorporates most or all variables as input, applies relevant feature engineering, and employs a smaller window size to increase resource utilization.

Observing the results, XGBoost consistently achieves relatively high R^2 and low MSE scores across different datasets and configurations, indicating its effectiveness in making accurate predictions for data repair. However, SGD exhibited significant variations, e.g., showcasing promising performance (R^2 = 0.69) in the light configuration in the Fersa dataset but encountering challenges (R^2 = -8.478e+20) in the heavy configuration in the Aeromec dataset, possibly indicating model complexity trade-offs. Conversely, the RNN and CNN models showed mixed results, with relatively lower R^2 scores and higher MSE values across different configurations, suggesting varying degrees of fit to the data.

Based on the performance results presented in Table 1, we can conclude that there are trade-offs between model complexity and predictive accuracy in the context of data repair in EDPRaaS. While more complex ML architectures (such as RNN and CNN) under the heavy configuration can offer higher predictive accuracy in some cases (e.g., in the Automotive dataset), they may also exhibit significantly poorer performance in other scenarios. On the other hand, the light configuration, which uses fewer variables and features in the models, may not always achieve the highest predictive accuracy but can provide consistent and acceptable performance across different datasets. Overall, the choice of model architecture and configuration seems pivotal in balancing predictive accuracy (R^2) and error magnitudes (MSE) in EDPRaaS. As such, decision-makers must weigh the advantages of more complex models against the potential pitfalls they may encounter, ensuring that the selected models align effectively with the specific characteristics of the datasets and operational requirements.

Summary of the RQ1 Results: *Selecting the ML architecture and configuration for EDPRaaS should consider dataset characteristics and the edge use case. Balancing model complexity with performance risk is crucial. For scenarios where reliability and consistency are crucial, adopting simpler ML architectures like XGBoost, which demonstrate relatively high R^2 scores across datasets, may be preferred. Understanding trade-offs allows informed decision-making to choose the most suitable ML solutions for efficient and accurate data repair in EDPRaaS.*

RQ2: Is EDPRaaS capable of timely response to erroneous data? To address RQ2, we evaluated the efficiency of the ML pipeline output (ML models) considering different parameters, such as model architecture, batch size, and dataset size. The goal was to determine the effectiveness of ML models as data repair tools within the edge environment and identify the most suitable configurations for optimal performance.

The resource consumption of the ML models is relatively modest, but optimization efforts are still warranted. The combined size of the modules exceeds 5 GB, and during execution, they utilize

Table 1: ML Model performance (R^2 and MSE) and efficiency (latency in seconds) in EDPRaaS.

	Fersa Dataset						Automotive Dataset						Aeromec Dataset					
Model	Light Configuration			Heavy Configuration			Light Configuration			Heavy Configuration			Light Configuration			Heavy Configuration		
	Late.	R^2	MSE	Late.	R^2	MSE	Late.	R^2	MSE	Late.	R^2	MSE	Late.	R^2	MSE	Late.	R^2	MSE
XGB	4.38	0.791	4.04e-09	5.28	0.8741	2.34e-09	4.81	0.75	22.81	5.03	0.90	9.13	5.73	0.82	0.003	6.28	0.99	0.0002
SGD	4.35	0.693	5.94e-9	4.49	0.7515	4.63e-9	4.81	0.08	84.32	4.90	0.34	59.81	4.96	0.25	0.011	6.72	-8.48e+20	1.26e+19
RNN	5.46	-0.004	1.94e-8	5.71	-2.51e-5	1.86e-08	6.00	0.26	67.66	6.17	0.76	21.79	6.13	0.03	0.015	7.94	0.80	0.0029
CNN	4.70	0.725	5.30e-9	4.90	-0.0004	1.86e-8	5.35	0.22	71.18	5.53	0.67	29.52	5.47	0.82	0.003	7.05	0.70	0.0044

approximately one core's worth of CPU power and up to 700 MB of RAM. These resource requirements might slightly increase if both modules experience peak usage simultaneously, which could occur in datasets with frequent errors. Further optimization is recommended to enhance efficiency.

We conducted latency assessments of EDPRaaS using lightweight and deep learning architectures, evaluating the time from validation to repair. By comparing the added latency with performance metrics, we determined the benefits of the edge-based solution. Furthermore, we investigated the impact of different parameter sets on latency, focusing on the influence of reducing the number of input variables and their complexity on prediction time. To do so, we selected two parameter sets for each dataset: one with few input variables and another incorporating feature engineering.

Table 1 displays the latency results for a batch of 1000 data points. As expected, deep learning architectures exhibit higher latency, with the recurrent neural network showing a maximum increase of up to 1.5 seconds. In the Fersa dataset, the simple models assert dominance by showcasing the low latencies in light and heavy configurations, indicating their ability to process and repair data swiftly. The CNN and RNN models exhibit relatively higher latencies, implying a trade-off between model complexity and processing speed. The Automotive dataset presents a similar trend, with the XGBoost model continuing to demonstrate commendable latency efficiency. The SGD model, which exhibited the lowest latency in the Fersa dataset, displays stable and competitive latencies in the Automotive dataset, showcasing its adaptability to different contexts. The RNN and CNN models again exhibit relatively higher latencies in the Aeromec dataset, highly likely because this dataset contains more input variables (larger input size).

As expected, the heavy configuration and complex architectures require more inference time, although the time difference is less pronounced (compared to model training). Thus, deploying edge-based inference for cases requiring moderately sophisticated models becomes a feasible option if we perform training on the cloud.

Summary of the RQ2 Results: *EDPRaaS efficiently repairs erroneous data using both simple and complex ML models at the edge. Our findings indicate that deploying resource-intensive deep learning models is feasible when their training occurs in the cloud.*

RQ3: How can EDPRaaS be deployed and run on the edge for industrial environments? To address RQ3, we analyzed the deployment of EDPRaaS in three industrial settings, summarizing the experience based on AI engineering dimensions. AI engineering, a discipline focusing on tools, systems, and processes for applying AI in real-world contexts, has been studied extensively [2, 3].

Data versioning and dependency management: We employed Data Version Control (DVC) [5] to manage EDPRaaS on edge devices by versioning only crucial data from the MQTT broker, eliminating redundancy and saving resources. In the IoT realm, where data needs constantly change, storing all data is unnecessary. The DVC 'dvc gc' command for garbage collection helps establish purging policies, automatically removing unused data and preserving vital information for current and future tasks. Therefore, EDPRaaS runs efficiently under resource constraints. By leveraging DVC for data versioning and dependency management, ML processes in EDPRaaS are streamlined and optimized without cloud offloading.

Deployment infrastructure: EDPRaaS, deployed on the Nerve Edge platform via Docker containers, comprises separate components. One container hosts the ML model and data processing logic, while another serves as an MQTT broker for data communication. Data from the Nerve platform's data gateway, supporting various protocols, is received by EDPRaaS. The MQTT broker facilitates real-time communication between components. The ML model within the Docker container analyzes incoming data, detects errors, and conducts data repair. Repaired data is published back to the MQTT broker for further processing or storage. The streamlined deployment using Docker containers ensures portability and seamless integration of EDPRaaS into the Nerve platform.

Monitoring and Logging: DVC and the Nerve software infrastructure facilitate monitoring and logging for EDPRaaS. DVC provides the "dvc add" command to track data files and "dvc run" to execute EDPRaaS. It enables versioning, replication, and reproducibility of data and models. Docker, integrated into Nerve, offers monitoring and logging features such as "docker stats" to obtain real-time resource usage statistics, "docker logs" to access container logs, and "docker events" to monitor container lifecycle events.

Integration of models and components. EDPRaaS utilizes ML frameworks (e.g., TensorFlow) for creating ML models and employs DVC for versioning, facilitating model management and reproducibility. Docker containerizes the ML models with necessary dependencies, ensuring easy portability and execution on the Nerve Edge platform. Docker Compose is used to integrate the containerized models with custom software components, forming a comprehensive data repair service. EDPRaaS provides a REST API endpoint for data repair requests, utilizing the trained ML model to identify and correct erroneous data. The Nerve platform, equipped with diverse communication protocols (e.g., MQTT), interacts seamlessly with the REST API. For instance, an MQTT broker collects IoT device data, forwarding it to the REST API endpoint for data repair. This integrated approach of containerized ML models, software components, and protocols on the Nerve platform enables efficient online data repair at the edge.

Summary of the RQ3 Results: *EDPRaaS is successfully deployed on the edge for industrial environments. It leverages DVC for data versioning and dependency management and employs Docker containers for streamlined deployment. Monitoring and logging capabilities are enabled through DVC and the Nerve software infrastructure. Integration of ML models and components, facilitated by ML frameworks and*

Docker, ensures seamless operation and efficient online data repair at the edge.

5.3 Threats to Validity

Internal validity. To limit threats to internal validity, we use EDPRaaS to exploit the correlation between sensor variables and not a cause-effect relationship between input and output sensors. We did not design EDPRaaS to predict future values.

External validity. To ensure the generalizability of our approach to different types of IoT data, we address the external threat to validity by utilizing three representative production lines. These lines manufacture different components for different companies, generating diverse data in terms of size and characteristics.

Construct validity. Validity threats related to measurement accuracy are a critical consideration. In the case of datasets comprising sensor data, it is possible for the measurements to lack correlation, leading to potential noise when assessing the performance of ML models. To address this concern, we took steps to mitigate the threat by consulting domain experts who possess expert knowledge in identifying the relevant sensor data to be utilized for making accurate predictions.

6 CONCLUSIONS

In this paper, we presented EDPRaaS (Edge-based Data Profiling and Repair as a Service), a novel and efficient approach for data quality enhancement in IoT environments. By deploying data repair operations at the edge, EDPRaaS addresses the challenges of real-time decision-making, bandwidth efficiency, resource constraints, and online operation. Additionally, the integration of Machine Learning (ML) in the data repair component enhances the accuracy and adaptability of the repair process, ensuring continuous data quality improvement. EDPRaaS leverages the power of pandas profiling and Great Expectations for comprehensive data profiling, enabling the identification of data quality issues such as missing values, outliers, and duplications. Our evaluations demonstrated the effectiveness and versatility of EDPRaaS, showcasing its potential for real-time data quality management in various IoT applications.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Commission's H2020 Programme under the grant agreement number 958363 (DAT4.Zero).

REFERENCES

[1] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the art of virtualization. *SOSP'03* 37, 5, 164–177.

[2] Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. 2021. Engineering AI systems: A research agenda. In *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*. IGI Global, 1–19.

[3] Anita D Carleton, Erin Harper, Tim Menzies, Tao Xie, Sigrid Eldh, and Michael R Lyu. 2020. The AI Effect: Working at the Intersection of AI and SE. *IEEE Software* 37, 4 (2020), 26–35.

[4] Danobat Smart Box. [n. d.]. https://digitalfactoryalliance.eu/components/danobat-smart-box/. Visited in 2023.

[5] Data Version Control. Visited in 2023. https://dvc.org/.

[6] Dominik Flick, Sebastian Gellrich, Marc-André Filz, Li Ji, Sebastian Thiede, and Christoph Herrmann. 2019. Conceptual Framework for manufacturing data preprocessing of diverse input sources. In *INDIN'19*. 1041–1046.

[7] Arda Goknil, Phu Nguyen, Sagar Sen, Dimitra Politaki, Harris Niavis, Karl John Pedersen, Abdillah Suyuthi, Abhilash Anand, and Amina Ziegenbein. 2023. A Systematic Review of Data Quality in CPS and IoT for Industry 4.0. *ACM Comput. Surv.* 55, 14s, Article 327 (jul 2023), 38 pages. https://doi.org/10.1145/3593043

[8] Great Expectations. [n. d.]. https://greatexpectations.io/. Visited in 2023.

[9] Sten Grüner, Julius Pfrommer, and Florian Palm. 2016. RESTful Industrial Communication With OPC UA. *IEEE Transactions on Industrial Informatics* 12, 5 (2016), 1832–1841. https://doi.org/10.1109/TII.2016.2530404

[10] Erik Johannes Husom, Simeon Tverdal, Arda Goknil, and Sagar Sen. 2022. UDAVA: An unsupervised learning pipeline for sensor data validation in manufacturing. In *CAIN'22*. 159–169.

[11] IBM Edge Computing. [n. d.]. https://www.ibm.com/docs/en/cloud-private/3.2.0?topic=edge-computing-servers. Visited in 2023.

[12] ISO/IEC International. 2008. *ISO/IEC 25012:2008 Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model.*

[13] Mauro Isaja, Phu Nguyen, Arda Goknil, Sagar Sen, Erik Johannes Husom, Simeon Tverdal, Abhilash Anand, Yunman Jiang, Karl John Pedersen, Per Myrseth, et al. 2023. A blockchain-based framework for trusted quality data sharing towards zero-defect manufacturing. *Computers in Industry* 146 (2023), 103853.

[14] Aimad Karkouch, Hajar Mousannif, Hassan Al Moatassime, and Thomas Noel. 2016. Data quality in internet of things: A state-of-the-art survey. *Journal of Network and Computer Applications* 73 (2016), 57–81.

[15] Mohammad Ayoub Khan and Fahad Algarni. 2020. A Healthcare Monitoring System for the Diagnosis of Heart Disease in the IoMT Cloud Environment Using MSSO-ANFIS. *IEEE Access* 8 (2020), 122259–122269.

[16] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. 2007. kvm: the Linux virtual machine monitor. In *the Linux symposium*, Vol. 1. 225–230.

[17] Tianxiang Kong, Tianliang Hu, Tingting Zhou, and Yingxin Ye. 2021. Data Construction Method for the Applications of Workshop Digital Twin System. *Journal of Manufacturing Systems* 58 (2021), 323–328.

[18] Hao Li, Xuefei Xu, Jinkui Ren, and Yaozu Dong. 2019. ACRN: a big little hypervisor for IoT development. In *VEE'19*. 31–44.

[19] Wei-Tsung Lin, Fatih Bakir, Chandra Krintz, Rich Wolski, and Markus Mock. 2019. Data Repair for Distributed, Event-Based IoT Applications. In *DEBS'19*. 139–150.

[20] MFN 100. [n. d.]. https://www.tttech-industrial.com/sps_flyer-download-mfn-100. Visited in 2023.

[21] Nerve Platform. [n. d.]. https://docs.nerve.cloud/. Visited in 2023.

[22] Phu H. Nguyen, Sagar Sen, Beatriz Bretones-Cassoli, Nicolas Jourdan, and Maria Chiara Magnanini. 2023. Software Engineering and AI for Data Quality in Cyber-Physical Systems/Internet of Things - SEA4DQ'22 Report. *SIGSOFT Softw. Eng. Notes* 48, 1 (jan 2023), 108–111. https://doi.org/10.1145/3573074.3573103

[23] Nwamaka U. Okafor, Yahia Alghorani, and Declan T. Delaney. 2020. Improving Data Quality of Low-cost IoT Sensors in Environmental Monitoring Networks Using Data Fusion and Machine Learning Approach. *ICT Express* 6, 3 (2020), 220–228.

[24] pandas. [n. d.]. https://pandas.pydata.org/. Visited in 2023.

[25] pandas-profiling. [n. d.]. https://pypi.org/project/pandas-profiling/.

[26] Luke Russell, Felix Kwamena, and Rafik Goubran. 2019. Towards Reliable IoT: Fog-Based AI Sensor Validation. In *IEEE Cloud Summit*. 37–44.

[27] Sunny Sanyal and Puning Zhang. 2018. Improving Quality of Data: IoT Data Aggregation Using Device to Device Communications. *IEEE Access* 6 (2018), 67830–67840.

[28] Saket Sathe, Thanasis G Papaioannou, Hoyoung Jeung, and Karl Aberer. 2013. A survey of model-based sensor data acquisition and management. In *Managing and mining sensor data*. 9–50.

[29] Sagar Sen, Erik Johannes Husom, Arda Goknil, Dimitra Politaki, Simeon Tverdal, Phu Nguyen, and Nicolas Jourdan. 2023. Virtual sensors for erroneous data repair in manufacturing a machine learning pipeline. *Computers in Industry* 149 (2023), 103917.

[30] Sagar Sen, Erik Johannes Husom, Arda Goknil, Simeon Tverdal, and Phu Nguyen. 2023. Uncertainty-aware Virtual Sensors for Cyber-Physical Systems. *IEEE Software* (2023).

[31] Sagar Sen, Erik Johannes Husom, Arda Goknil, Simeon Tverdal, Phu Nguyen, and Iker Mancisidor. 2022. Taming Data Quality in AI-Enabled Industrial Internet of Things. *IEEE Software* 39, 6 (2022), 35–42.

[32] TTTech Industrial. [n. d.]. https://www.tttech-industrial.com/. Visited in 2023.

[33] K. Villalobos, V.J. Ramírez-Durán, B. Diez, J.M. Blanco, A. Goñi, and A. Illarramendi. 2020. A three level hierarchical architecture for an efficient storage of industry 4.0 data. *Computers in Industry* 121 (2020), 103257.

[34] Chang Wang, Yongxin Zhu, Weiwei Shi, Victor Chang, P. Vijayakumar, Bin Liu, Yishu Mao, Jiabao Wang, and Yiping Fan. 2018. A Dependable Time Series Analytic Framework for Cyber-Physical Systems of IoT-Based Smart Grid. *ACM Transactions on Cyber-Physical Systems* 3, 1 (2018), 18.

[35] Y Richard Wang, Lisa M Guarascio, and Richard Wang. 1991. Dimensions of data quality: Toward quality data by design. (1991).

[36] Sholom M. Weiss, Amit Dhurandhar, and Robert J. Baseman. 2013. Improving Quality Control by Early Prediction of Manufacturing Outcomes. In *KDD'13*. 1258–1266.