Supplementary Materials

Input data requirements and preprocessing assumptions for the WATERVERSE Data Quality Framework

The WATERVERSE Data Quality Framework (WDQF) is designed to operate on harmonised tabular datasets that have already been ingested into the broader data management environment. For time-series assessment, the minimum required inputs are: (i) one or more identifier columns, used to distinguish assets, devices, locations, or entities; (ii) a timestamp column, used to define temporal ordering and expected sampling continuity; and (iii) one or more measured-variable columns to be assessed.

In addition, the DQA-tool requires dataset metadata describing the temporal scope of the assessment and the expected sampling frequency. This metadata is used in particular for the evaluation of completeness and timeliness. Optional validation rules may also be supplied to define admissible formats and variable-specific value constraints, such as lower and upper thresholds, positivity or non-zero requirements, decimal formatting, or flatline criteria.

The WDQF is schema-configurable and is not tied to a specific source format. In practice, heterogeneous operational exports can be mapped to the required structure through upstream ingestion and harmonisation processes within the broader data management environment. These upstream processes may include, where relevant, timestamp normalisation, unit standardisation, vendor-specific schema mapping, and conversion of cumulative meter-register readings into interval values.

Accordingly, the DQA/DVR workflow does not require raw AMI data in a particular native format. If cumulative readings are used by a source system, the conversion to interval consumption is assumed to be completed before the data enter the WDQF workflow. Any artefacts remaining after this preprocessing step, such as negative interval values caused by rollovers, resets, or timestamp misalignment, can then be detected and assessed by the framework through its configured quality rules.

For reconciliation, the DVR-tool requires historical observations sufficient for the selected anomaly-detection and reconciliation method. The minimum data requirement therefore depends on the method used. Deterministic approaches such as interpolation can be applied with very limited history, in the simplest case requiring only neighbouring observations around a gap. By contrast, data-driven reconciliation methods generally require a larger volume of historical data to learn representative temporal patterns and relationships among variables.

Limited historical data may introduce uncertainty in reconciled values, especially for variables with strong seasonality, irregular event-driven behaviour, or long-term temporal dependencies. In such cases, model performance may be constrained when only short periods of historical data are available. Periodic retraining as more data become available, the use of auxiliary predictors, or the application of mechanistic or rule-based approaches may provide more robust alternatives under sparse-data conditions.

The framework is schema-configurable and not tied to a single source format. In practice, heterogeneous operational exports can be mapped to the required structure through upstream ingestion and harmonisation processes. Source-specific preprocessing steps, such as timestamp normalisation, unit standardisation, treatment of cumulative versus interval values, or vendor-specific schema mapping, are assumed to be completed before the DQA/DVR workflow begins unless explicitly incorporated into an upstream processing pipeline.

Overview of DVR-tool methods

The DVR-tool supports a set of configurable methods for anomaly detection and data reconciliation. These methods can be selected according to the type of data-quality issue, the structure of the dataset, and the intended corrective action.

For anomaly detection, the available methods include missing data detection, threshold detection, flatline detection, zero-value detection, spike/drop detection, and a method combining Minimum Covariance Determinant (MCD) with Mahalanobis-distance calculation. Missing data detection identifies absent timestamps by comparing observed

and expected temporal intervals, using as main inputs the timestamp column, identifier column(s), start and end of the time window, and expected sampling frequency. Threshold detection flags values outside predefined lower and/or upper admissible bounds. Flatline detection identifies suspicious runs of nearly constant values that may indicate sensor malfunction or reporting errors. Zero-value detection flags zero values when these are implausible or disallowed for a given variable. Spike/drop detection identifies sudden local deviations relative to neighbouring observations or a local baseline. The MCD + Mahalanobis-distance anomaly detection extends the approach to multivariate settings by identifying outliers covariance .

For reconciliation, the DVR-tool supports forward fill, interpolation-based reconciliation and Random Forest regression reconciliation. The forward fill reconciliation method provides the previous timestamp value, the preceding known value, as alternative to anomalous or missing values. Interpolation-based reconciliation replaces anomalous or missing values using local interpolation, with configurable interpolation rules and gap-length constraints. Random Forest regression reconciliation predicts anomalous or missing values using supervised learning from historical or auxiliary data, with configurable training data, predictor sets, lag structure, and model hyperparameters.

In terms of practical use within this paper, missing data detection was applied in the Spanish pilot, while threshold detection, flatline detection, and spike/drop detection were applied in the Dutch pilot. Random Forest regression reconciliation was used in both pilots. Zero-value detection, Mahalanobis-distance anomaly detection, and interpolation-based reconciliation were available in the DVR-tool but were not used in the reported pilot implementations.

Overall, these methods provide the DVR-tool with a high degree of flexibility, ranging from simple rule-based checks to multivariate anomaly detection and machine-learning-based reconciliation.

Random Forest Regression

Random Forest regression (RFR) uses the Random Forest (RF) algorithm developed by Breiman (Breiman 2001). By Bootstrap resampling the original dataset is converted to multiple samples. For each Bootstrap sample a decision tree is set up that makes a prediction. The predictions by all the individual trees are then averaged to give the final prediction. The RFR algorithm is essentially a multi-decision tree model: multiple decision trees are combined to generate a single prediction (Cheng et al 2023).

The main characteristics of a RFR model are (Cheng et al 2023):

- **The number of estimators:** the number of trees in the random forest.
- **The maximum number of features:** the number of features that are considered upon splitting.
- **The maximum depth:** how deep each tree can grow at maximum.
- **The minimum number of samples for splitting:** the number of samples that are minimally required for splitting an internal node.
- **The minimum number of samples at a leaf node:** the number of samples that are minimally required at a leaf node.
- **Criterion:** the evaluation criteria that is used for node-splitting.

As criterion the mean squared error was used in this research, given by (Cheng et al 2023):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2,$$

where y_i represents the actual value and $\hat{y}_i$ the value predicted by the model.

Anomaly Detection and Reconciliation for Spanish Pilot

Anomaly Detection (AD) for the Spanish pilot was performed utilising the missing data AD method of the DVR-tool. This AD method checks whether the time difference between two sequential data points is equal to the expected data frequency. If not, the missing timestamps are determined and added to the dataset, flagged as anomalous.

To train the RFR model for the reconciliation of the missing data points for meter 202 different type of features were used:

- **Temperature:** temperature data coming from meteorological records for Malaga (Agencia Estatal de Meteorología (AEMET) 2025). Both the temperature of the current timestamp and lagged temperature values were used (temperature lagged by one, two and three timestamps).
- **Historic DELTA data:** the difference between sequential lagged DELTA values for lag 1 and 2, lag 2 and 3, lag 3 and 4, lag 4 and 5, lag 8 and 9, and lag 12 and 13 were used to provide the model information on the peak height. In addition, the maximum value of a rolling window on the historic consumption data was used as feature to emphasise the peak height. The rolling window had a window size of 5, containing the five previous DELTA values.
- **Date and time:** based on the timestamp the month (1 – 12), the week of the month (1–5), and the hour (0 – 23) of each data point was derived and used as feature for training. Especially the month and hour features are presumed to give the model insight in the seasonability and daily patterns in household water consumption, respectively.

To help the model capture the high DELTA values a weight of ten was given to the values above 200. Using the RandomizedSearchCV function from the scikit-learn python package with five-fold cross validation and 5000 iterations the hyperparameters for the RF model were tuned. The resulting hyperparameters for the RF model were as follows:

- **Bootstrap:** True
- **Max. features:** sqrt
- **Max. depth:** None
- **Min. sample split:** 5
- **Min. sample leaf:** 2
- **Max. leaf nodes:** None
- **n estimators:** 120
- **Criterion:** squared error
- **Random state:** 42

The resulting model performance for the RF model is given in Table 2. The R2 score for the test set is rather low. The limited amount of available data (only one year) appears to limit the degree of seasonality that can be learned by the model. To enhance the model performance multiple years of data must be available. This way the model can learn the seasonality and use the DELTA value for the same timestamps in previous years to predict this specific timestamp in the current year. Figure 1 shows the actual and predicted values for both the train and test set. As described in the main text, the train set contained the first three weeks of each month present in the dataset, while the test set contained the last week(s) of each month (week 4 and 5). To avoid data leakage hour 0 – 13 of day 1 and 22 were removed from the dataset before the dataset was split into a train and test set.

Figure 2 presents the feature importance. The difference between the 12th and 13th lag of the DELTA parameter contributes most to the predictions of the model, followed by the difference between the 8th and 9th lag and the hour. The daily patterns are thus of main importance for the model.

Figure 3 shows the detected gaps of missing data, the actual values measured by meter 202 and the by the RF model reconciled values for the complete dataset. Notably, the model predicts higher DELTA values in the summer months (July – September) than in the winter months (December – February). By taking the first three weeks of each month for training the model was thus able to learn seasonality to a certain extent.

Anomaly detection and reconciliation for Dutch pilot

AD for the Dutch pilot was performed utilising the flatline and spikedrop method of the DVR-tool. The flatline method checks for stretches of constant values over a minimum duration, while the spikedrop method detects abrupt rises or drops relative to a local baseline. Detected anomalous segments were flagged for reconciliation.

To train the RFR model for the reconciliation of the anomalous data for conductivity sensor 1, lagged features of sensor 2 were used:

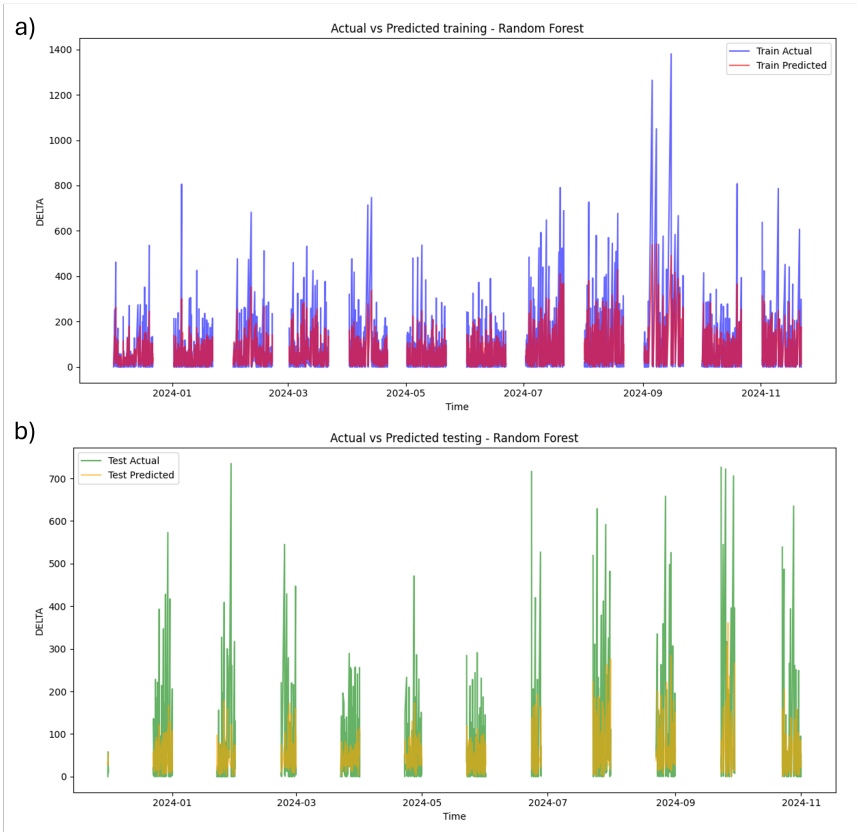

Figure 1. Predicted and actual values for a) the train and b) the test set of the RF model for the Spanish model.

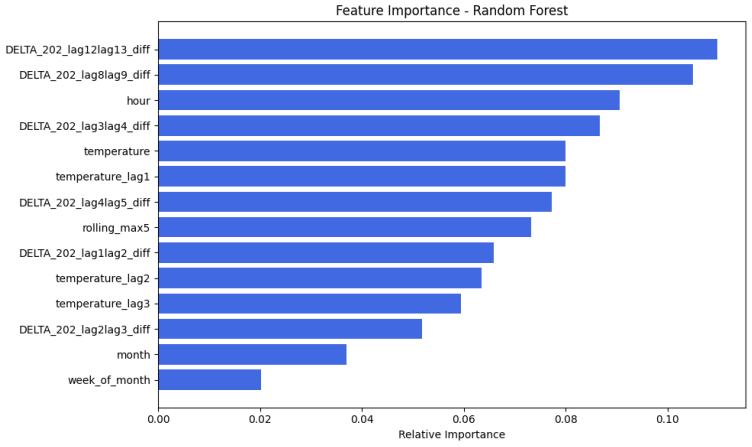

Figure 2. Feature importance for the RF model for the Spanish model.

- **Cross-sensor lagged conductivity:** lagged values (1–24 timestamps) of conductivity sensor 2 were used as features to capture short-term dynamics and the daily cycle.

Hyperparameters for the RF model were tuned using Bayesian optimisation with five-fold cross validation. The resulting hyperparameters for the RF model were as follows:

- **Bootstrap:** True
- **Max. features:** 1.0
- **Max. depth:** 18
- **Min. sample split:** 2
- **Min. sample leaf:** 23
- **Max. leaf nodes:** None
- **n estimators:** 45

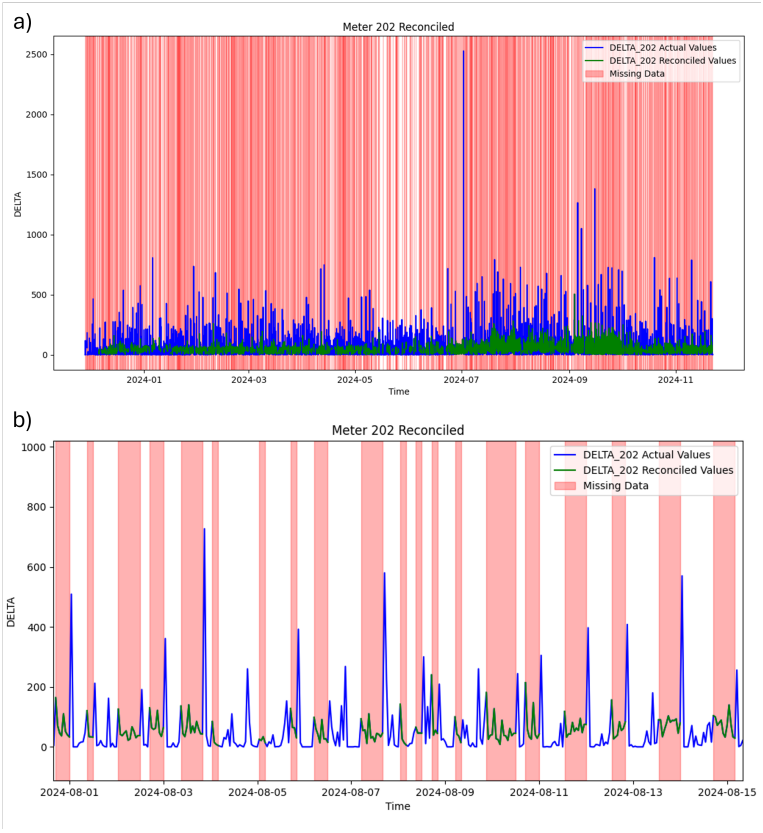

Figure 3. a) Detected missing data anomalies for meter 202 together with the reconciled values for the complete dataset, and b) zoom on two weeks period.

- **Criterion:** squared error
- **Random state:** 42

The resulting model performance for the RF model is given in Table 2. Both the train and test R^2 scores are high, indicating that with only cross-sensor lagged inputs the model adequately captures the relevant dynamics. Figure 4 shows the actual and predicted values for both the train and test set. As described in the main text, the first 80% of the time series was used for training and the remaining 20% for testing in order to respect temporal ordering.

Figure 5 presents the feature importance. Cross-sensor lags within the last few hours and the diurnal component (lag 24) contribute most to the predictions of the model.

Figure 6 shows the detected anomalies, the actual values measured by conductivity sensor 1 and the by the RF model reconciled values for the complete dataset. The model follows the diurnal signal and short-term variability, while smoothing anomalous spikes and drops.

To train the RFR model for the reconciliation of the anomalous data for conductivity sensor 2, lagged features of sensor 1 were used:

- **Cross-sensor lagged conductivity:** lagged values (1–24 timestamps) of conductivity sensor 1 were used as features to capture short-term dynamics and the daily cycle.

Hyperparameters for the RF model were tuned using Bayesian optimisation with five-fold cross validation. The resulting hyperparameters for the RF model were as follows:

- **Bootstrap:** True
- **Max. features:** 1.0
- **Max. depth:** 22
- **Min. sample split:** 4
- **Min. sample leaf:** 27
- **Max. leaf nodes:** None

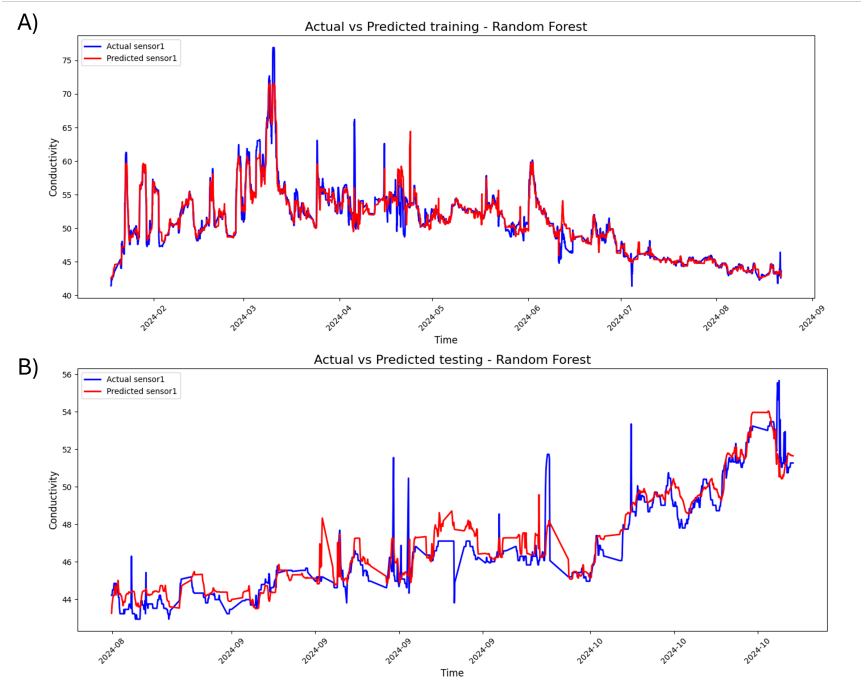

Figure 4. Predicted and actual values for a) the train and b) the test set of the RF model for the Dutch model.

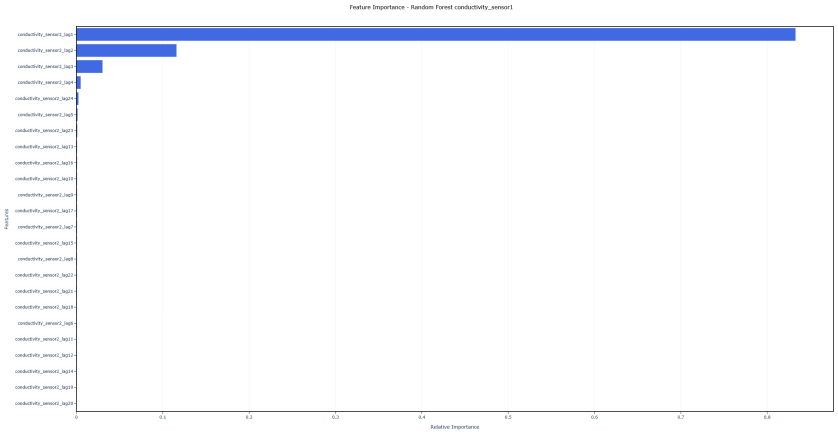

Figure 5. Feature importance for the RF model for the Dutch model.

- **n estimators:** 34
- **Criterion:** squared error
- **Random state:** 42

The resulting model performance for the RF model is given in Table 2. Both train and test R^2 scores are high, indicating that with only cross-sensor lagged inputs the model adequately captures the relevant dynamics. Figure 7 shows the actual and predicted values for both the train and the test set. As described in the main text, the first 80% of the time series was used for training and the remaining 20% for testing in order to respect temporal ordering.

Figure 8 presents the feature importance. Cross-sensor lags within the last few hours and the diurnal component (lag 24) contribute most to the predictions of the model.

Figure 9 shows the detected anomalies, the actual values measured by conductivity sensor 2 and the by the RF model reconciled values for the complete dataset.

A)

B)

Figure 6. a) Detected anomalies for conductivity sensor 1 together with the reconciled values for the complete dataset, and b) zoom in two weeks period.

A)

B)

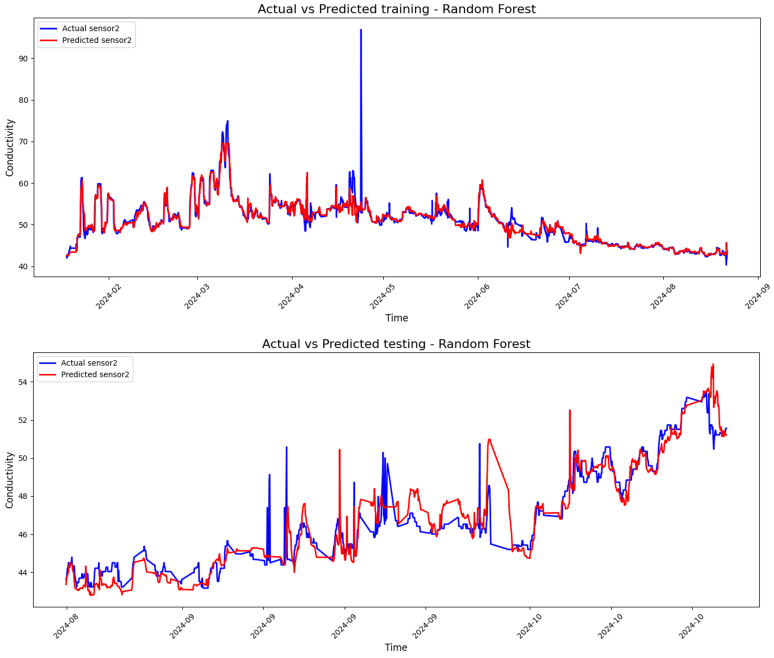

Figure 7. Predicted and actual values for a) the train and b) the test set of the RF model for the Dutch model (conductivity sensor 2).

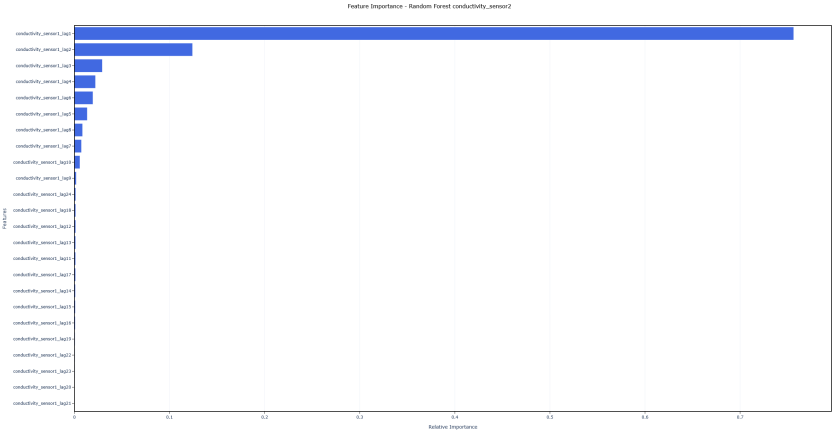

Figure 8. Feature importance for the RF model for the Dutch model (conductivity sensor 2).

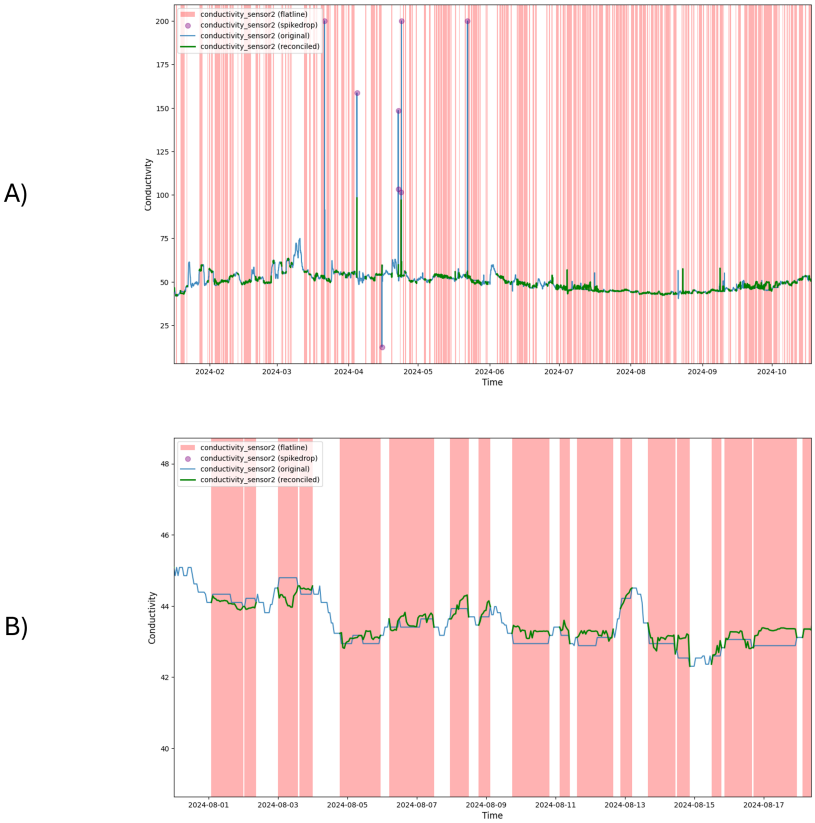

Figure 9. a) Detected anomalies for conductivity sensor 2 together with the reconciled values for the complete dataset, and b) Zoom in two weeks period.