

CISSAN

Collective intelligence supported by security aware nodes

D4.5 Methods and Software for Data Integration and Data Quality Verification

Editor: Klaus Chmelina, Geodata

Abstract

This document describes the Data Quality Verification System developed for evaluating the believability of data. The system is based on combining an existing cloud-based data management platform and a newly developed software analyzing the data within the platform by using empirical Data Quality Rules.

The document describes the system's objective, the conceptual approach of using believability scores, the architecture, principal workflow, and the depicted methods (Data Quality Rules) and concepts (clustering, windowing, aggregation) used for data analysis. It further presents a prototype implemented for the use-case of tunnel construction and illustrates evaluation examples.

Additionally, the capabilities of two AI-supported approaches are explored to assess data believability and the presence of potential anomalies.

The developed system enables organisations to assess the perceived believability of their data and, thus, contributes to data security.

Project CISSAN

Public Report

October 2025

Participants in project CISSAN are (in alphabetic order with the project coordinator first):

- University of Jyväskylä (coordinator)
- Affärsverken Karlskrona
- Arctos Labs Scandinavia AB
- Bittium Biosignals Ltd.
- Bittium Wireless Ltd.
- Blekinge Tekniska Högskolan
- Blue Science Park
- Clavister AB
- Councilbox Ltd.
- Geodata ZT GmbH
- Mattersoft Ltd.
- Mint Security Ltd.
- Netox
- Nodeon Finland Oy
- Savantic
- ScopeSensor Ltd.
- Techinova AB
- Wirepas Oy

CISSAN-Collective intelligence supported by security aware nodes

D4.5 Methods and Software for Data Integration and Data Quality Verification

Editor: Klaus Chmelina, Geodata

Project coordinator: Ilgin Safak, University of Jyväskylä

CELTIC published project result

© 2025 CELTIC-NEXT participants in project CISSAN

Disclaimer

This document contains material, which is the copyright of certain PARTICIPANTS, and may not be reproduced or copied without permission.

All PARTICIPANTS have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the PARTICIPANTS nor CELTIC-NEXT warrant that the information contained in the report is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Executive Summary

This report describes the Data Quality Verification System, representing a CISSAN solution for assessing the believability of data in IoT platforms. The system addresses the challenge of detecting data tampering (manipulation, deletion, insertion) by (cyber)attacks as well as operational anomalies. The believability of data is assessed using empirical Data Quality Rules based on mathematical, statistical and AI methods calculating Believability Scores, which are assigned to the data and provided to the user through a cloud-based data management platform.

The report describes the objective, conceptual approach, architecture, principal data flow, and the depicted methods (Data Quality Rules) and concepts (clustering, windowing, aggregation) used to analyse the data and determine their believability. It further presents a prototype implemented for the use case of tunnel construction and illustrates evaluation examples.

List of Authors

In alphabetic order by partner name:

- Klaus Chmelina, Geodata ZT GmbH
- Alois Maierhofer, Geodata ZT GmbH
- Maximilian Krieger, Geodata ZT GmbH
- Ilgin Safak, University of Jyväskylä

Table of Contents

- Executive Summary 3
- List of Authors 4
- Table of Contents 5
- List of Figures 6
- Abbreviations 7
- 1 Introduction 8
 - 1.1 Objective of this document 8
 - 1.2 Content and structure of this document 8
- 2 Data Quality Verification System 10
 - 2.1 System Objective and Conceptual Approach 10
 - 2.2 System Architecture and Principal Workflow 11
- 3 Method Description 12
 - 3.1 Believability Scores 12
 - 3.2 Aggregation of Believability Scores 12
 - 3.3 Windowing 14
 - 3.4 Data Quality Rules 15
 - 3.4.1 DQ rule - Timeliness 15
 - 3.4.2 DQ rule - Frequency 16
 - 3.4.3 DQ rule - Noise 17
 - 3.4.4 DQ rule - Trend Similarity 19
 - 3.4.5 DQ rule - Change Point Similarity 22
 - 3.5 AI approach 25
 - 3.5.1 Prediction Transformer 25
 - 3.5.2 Large Language Models (LLMs) 29
- 4 Software Prototype 34
 - 4.1 Components and Their Software Implementations 34
 - 4.1.1 GeodataHub 34
 - 4.1.2 DQ Verification Component 34
 - 4.1.3 DQ Repository 35
 - 4.1.4 JSON Structure 35
 - 4.2 Software Package Summary 37
- 5 Conclusion 39
- References 40

List of Figures

Figure 1. Data Quality dimensions and the addressed dimension “Believability”	10
Figure 2. System architecture of the Data Quality Verification System. Green parts are newly developed, other parts belong to the existing IoT platform GeodataHub that was further extended.	11
Figure 3. Main workflow, software components, development environments, platforms and communication interface of the software prototype.....	11
Figure 4. Granularity levels of Believability Scores.	12
Figure 5: Example of granularity levels for tunnel displacement monitoring and aggregating Believability Scores by the minimum function.	13
Figure 6: Tunnel with monitoring cross sections and 3D targets.....	13
Figure 7: Illustration of the different windowing options for a time series.....	14
Figure 8: Visualization of a non-overlapping windowing with the size 10 measurements.	15
Figure 9: Visualization of a non-overlapping windowing with the size 20 days.....	15
Figure 10. Example of a complex processing pipeline.	15
Figure 11. Time differences and their respective Believability Scores in color, sorted by time difference. Id = running number of time difference.....	16
Figure 12. Differences of sensor values to their linear regression and the expected (orange) and maximum (black) variance thresholds.	17
Figure 13. Believability Scores of different time series for the believability aspect Noise.....	18
Figure 14. Believability Scores of different time series pairs for the believability aspect Trend Similarity.	20
Figure 15. Believability Scores of a time series cluster consisting of three time series and for selected windows of the time series cluster.	21
Figure 16. Believability Scores of four time series when compared with a reference time series. ...	21
Figure 17. Principle of change point detection using the P&C method.	23
Figure 18. Change points of a time series from the tunnel construction use case detected by the P&C method.	23
Figure 19. Change points and their change point scores for a cluster of three time series from the tunnel construction use case.	24
Figure 20. Predicted (upper image) and observed data (lower image) using a Prediction Transformer.	25
Figure 21. A time series with a change in the function at point 35.....	26
Figure 22. Prediction for window 18 and window 35.	27
Figure 23. RMSE values and the thresholds in the top plot. The bottom plot shows the believability score of each point. The color represents each point's score in both plots. The dashed grey line represents the point where the change happened in both plots.	28
Figure 24. TS1 values with the color representing the believability score for each point.	29
Figure 25. DQ Verification System, software prototype architecture, used communication protocols, data formats, and networks.	37
Figure 26. Sensor time series (upper diagram) and its Believability Scores (lower diagram) displayed in a dashboard of the IoT platform GeodataHub.....	38

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AR	Autoregression
ARIMA	Auto-Regressive Integrated Moving Average
CUSUM	Cumulative Sum
DQ	Data Quality
DTW	Dynamic Time Warping
IoT	Internet of Things
JSON	JavaScript Object Notation
LLM	Large Language Model
MA	Moving Average
MCP	Model Context Protocol
RMSE	Root Mean Squared Error
3D	Three-Dimensional

1 Introduction

This document describes the Data Quality Verification System that has been developed in the CISSAN project. It aims to estimate the believability of data managed by the Internet of Things (IoT) platforms and enable confident decision-making based on trusted data.

In particular, this is achieved through:

- providing a tamper protection mechanism ensuring that only untampered and high-quality data is used for decision making, and where data with low credibility is excluded,
- providing a quick overview of the data health within the IoT platform, and
- automatically informing, alarming and guiding the user to potentially problematic data.

Particular motivation behind the development is the current situation of the tunnel construction use case, where huge and rapidly growing volumes of monitoring data are acquired every day during tunnel construction and used as a basis for manifold decisions. Examples include the daily optimization of construction methods and support measures, the realtime control of tunnelling machines, and the initiation/triggering of safety measures. Decisions based on tampered/wrong data may lead to critical situations ranging from construction delays to damages and even catastrophes.

Thus, the developed system aims to avoid such problems in the future.

1.1 Objective of this document

The primary objective of this document is to provide a comprehensive overview of the Data Quality Verification System. The document aims to:

- **Detail the Architecture:** Describe the components, including an existing IoT platform for sensor data acquisition, integration and provision, and the newly developed Data Quality Verification component, and their roles in assessing data believability.
- **Explain the Functionality:** Outline, how data is exchanged between the components, analysed by empirical Data Quality Rules, and how the final results, the Believability Scores, are presented and exploited.
- **Provide Examples:** Present examples of data sets and the results of their quality analysis.
- **Present a software prototype:** Describe the implementation of a software prototype and a demonstrator used to showcase the system in operation.

1.2 Content and structure of this document

To achieve the above objective, the document is structured as follows:

- **Introduction**
 - Objective of this document
 - Content and structure of this document
- **System Objective and Conceptual Approach**
 - Goal of the system
 - Clarification of data quality terminology and cyber-security context
 - Description of overall concept and approach
- **System Architecture and Principal Workflow**
 - Description of system components and data flow
 - Description of used development environments, languages and platforms
- **Method Description**

- Description of empirical rules for data believability analysis
- Examples of data and their believability analysis
- **Software Prototype**
 - Description of the developed software prototype
- **Conclusion**

Each section is designed to progressively build upon the reader's understanding, from high-level concepts down to specific technical implementations, ensuring clarity on how the Data Quality Verification System operates to assess the believability of IoT data.

2 Data Quality Verification System

2.1 System Objective and Conceptual Approach

The Data Quality Verification System is designed to determine the data quality dimension “**Believability**” (Figure 1), which refers to its trustworthiness, credibility, and perceived truthfulness. Believability is a crucial aspect of data quality closely related to data security, and only a high degree of it enables confident data-driven decision-making.

In our concept, data believability is determined by calculating **Believability Scores** assigned to the data. The calculation is carried out by applying domain-specific **Data Quality Rules**. The rules are based on empirical methods representing exclusive domain knowledge and employing different concepts ranging from simple statistics, plausibility and threshold checks to special mathematical algorithms and Artificial Intelligence (AI) / Large Language Models (LLMs). Each Data Quality Rule outputs a Believability Score. Several such scores can be aggregated to form a total Believability Score for a data record, a cluster of data records or even the health of all the data within an IoT platform or data system.

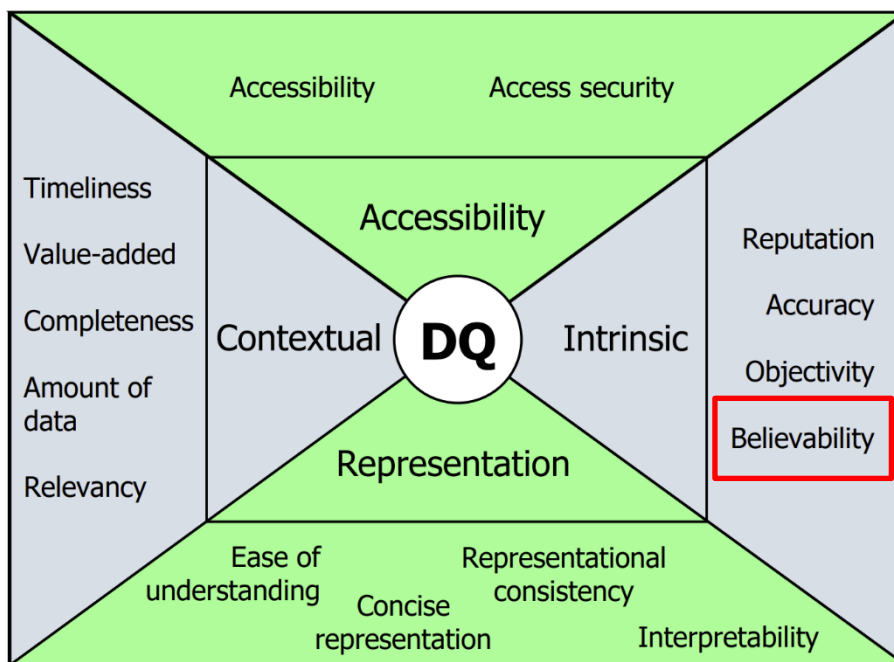


Figure 1. Data Quality dimensions and the addressed dimension “Believability”.

Depending on the choice and parameterization of the Data Quality Rules, the obtained Believability Scores may have different meanings and cross correlations with other data quality dimensions:

- *Believability* in an operational sense
defines the degree the data is believed to be correct in a purely operational, technical sense. (e.g., free of errors, accurate, complete).
- *Believability* in a data security-related sense
defines the degree the data is believed to be free of tampering (e.g., free of manipulation, deletion, insertion), be it intended (by an attacker) or unintended.

As an example, a temperature sensor sending “9999999°C” clearly sends unbelievable data. This may indicate an outlier, a damaged sensor (operational sense) or an attack (data security-related sense). The root cause may not always be identifiable.

The system prototype has been developed and implemented specifically for the CISSAN tunnel construction use case and for dealing with time series data (monitoring data, sensor data). However, the general concept was observed to be also applicable to other data types, domains and use cases. The system addresses challenges in IoT data management, enabling organizations to exploit their

data to find operational and security-related data anomalies, and other data-related issues. In the narrower sense, the system can be seen as an empirical Data Believability Verification system.

2.2 System Architecture and Principal Workflow

The architecture of the developed Data Quality Verification System is composed of several components, including (Figure 2)

- the existing **IoT platform GeodataHub** providing basic functions and services for Sensor Data Gathering, Integration and Storage, Provisioning and Presentation,
- a newly developed **Data Quality (DQ) Verification Component** applying the Data Quality Rules to time series data and exchanging data with the IoT platform,
- a **DQ repository** allowing the configuration, parameterization, and storage of Data Quality Rules according to the needs of the domain, use case, and user, and
- an optional **Data Quality Improvement module** allowing data correction.

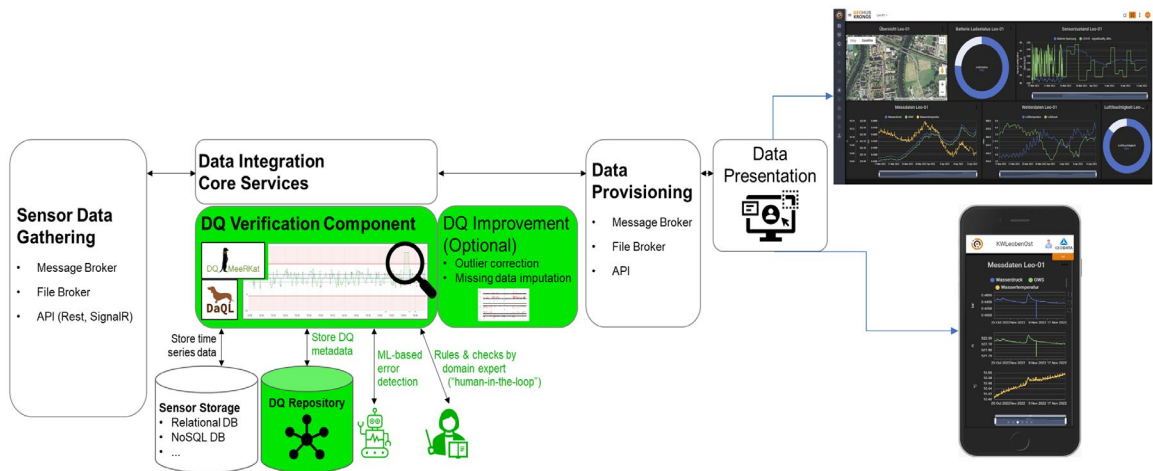


Figure 2. System architecture of the Data Quality Verification System. Green parts are newly developed, other parts belong to the existing IoT platform GeodataHub that was further extended.

Figure 3 depicts the development environments and principal communication interfaces. The existing IoT platform (GeodataHub) and the newly developed DQ Verification Component represent the main system/software components interacting. To request a DQ verification, GeodataHub sends a JavaScript Object Notation (JSON) request to the DQ Verification Component containing the storage address of the data to be explored, the IDs of the rules to be applied, and other control parameters. The DQ Verification Component then queries the data from the IoT platform, applies the rules, processes the Believability Scores, aggregates them and returns the results to the IoT platform where they are further used. The process can be started on demand (by the user), periodically, and/or may be event-triggered.

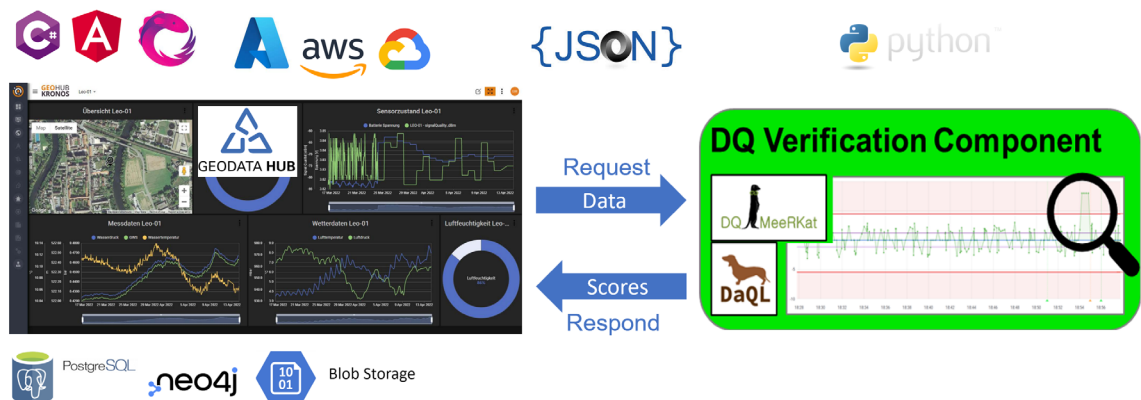


Figure 3. Main workflow, software components, development environments, platforms and communication interface of the software prototype.

3 Method Description

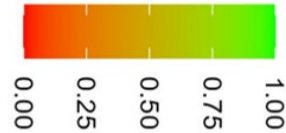
3.1 Believability Scores

In our approach, data believability is represented by **Believability Scores**. The Believability Score is defined as a calculated numerical value between zero (0) and one (1) for a certain **granularity level** and a certain **believability aspect** of the data.

“0”: data is unbelievable

“1”: data is believable

(0,1): believability is between unbelievable and believable



The lowest **granularity level** (Figure 4) of a Believability Score in the given use case is per sensor value, the highest level is per project, giving the believability of the whole monitoring data in the server/of a project (= overall project health score). In between, further granularity levels exist like per time series window (= a sequence of sensor values of a time series), per whole time series (= all sensor values of a time series which can also be regarded as sensor believability or sensor health) and per time series cluster (= several time series, which can also be regarded as a sensor cluster believability or health).

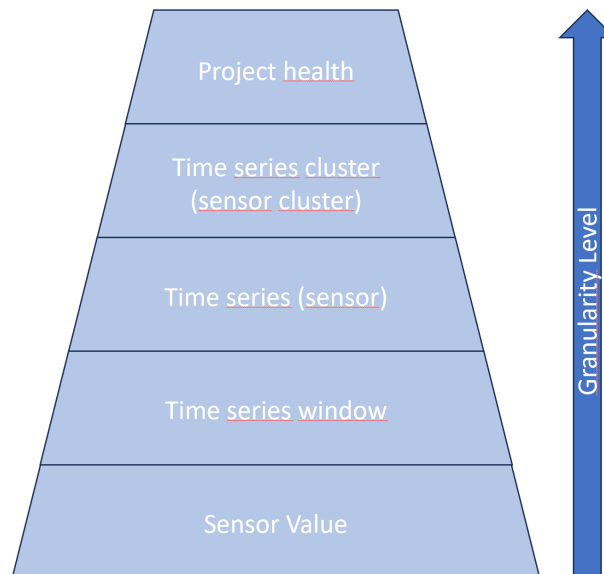


Figure 4. Granularity levels of Believability Scores.

3.2 Aggregation of Believability Scores

By aggregating Believability Scores of lower granularity levels, we can assign Believability Scores to higher, coarse-grained levels and establish a hierarchical structure. As an example, the believability of a time series can be aggregated from the Believability Scores of single sensor values of the time series. Believability Scores can be aggregated to the next higher granularity levels by using aggregation functions which may be a simple ratio, a (weighted) average, or a minimum-maximum operation. The aggregation function can be selected depending on the domain and use case, depending on the user preferences. Figure 5 highlights an example of a hierarchical granularity level structure suitable for tunnel displacement monitoring shown in Figure 6 where several three-dimensional (3D) targets (sensors) are installed in regularly spaced cross sections (sensor clusters) every 10m inside the tunnel. To aggregate from lower granularity level to a higher level, the min-function is selected, evaluating a worst-case scenario where the worst believability score is always passed forward.

The number of granularity levels that can be established, and the aggregation functions used can be configured and combined in a flexible manner.

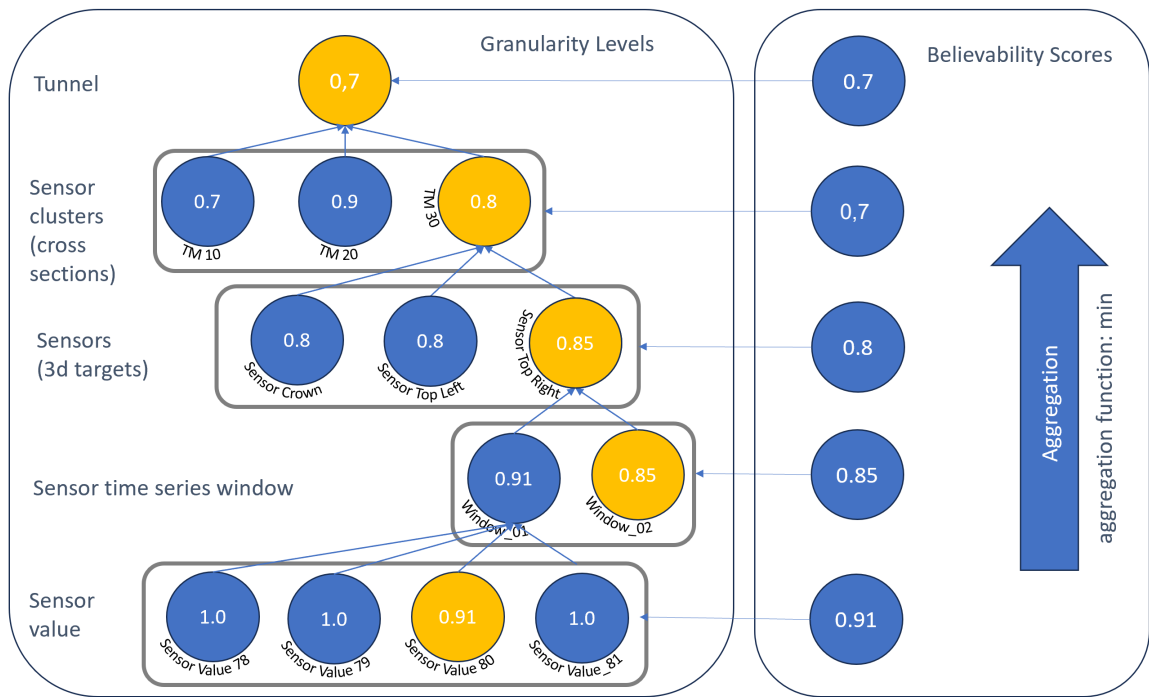


Figure 5: Example of granularity levels for tunnel displacement monitoring and aggregating Believability Scores by the minimum function.



Figure 6: Tunnel with monitoring cross sections and 3D targets.

For decision making, one needs not only the reports on the current data believability, but also on its change with time. Hence, the Believability Score is a value, which needs to be tracked on all granularity levels by continuously monitoring the new sensor data. One Believability Score can therefore be considered as a snapshot of the data's believability at a fixed point in time. Every new

sensor value may increase or decrease sensor believability, and consequently the Believability Scores at higher granularity levels.

3.3 Windowing

Aggregating Believability Scores for all available sensor data results in an overall Data Believability Score of a project/system (e.g., a tunnel). However, for a more detailed analysis, looking at the scores for parts of the project/system is important. Therefore, in addition to calculating the score for the whole time series, Windowing is used to enable a more detailed time series analysis. The size/length of the window is either defined by the number of measurements (n) or a considered time interval (e.g., 5 days). Another option considered is a combination of both, where the window length is defined by the newest/last n measurements in the considered time interval. Windowing can be applied differently (see Figure 7):

- a. sliding window without overlapping
- b. sliding window with overlapping windows
- c. growing window

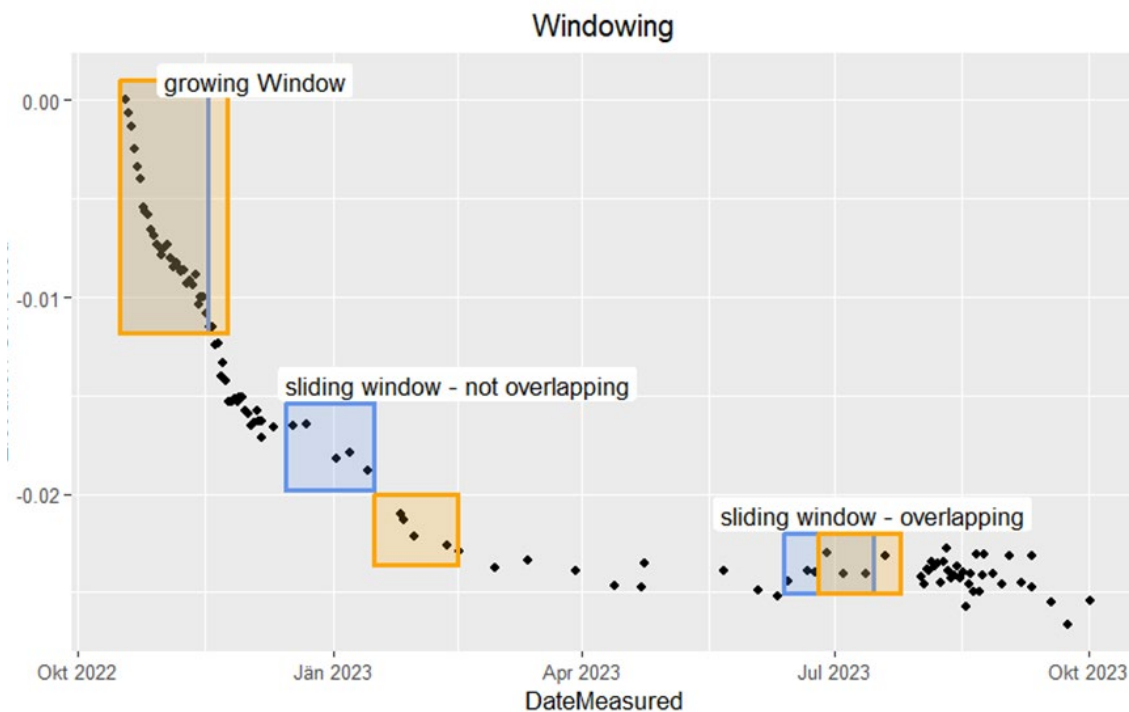


Figure 7: Illustration of the different windowing options for a time series.

The first two options (a and b) split a given time series into smaller parts (sub-clustering) where the user can define the window size by either the number of measurements/sensor values (Figure 8) or the time interval (Figure 9). Furthermore, those two options can also be combined into a hybrid approach. Option c enables the analysis of the whole time series over time. As time passes, more data becomes available, leading to a growing window.

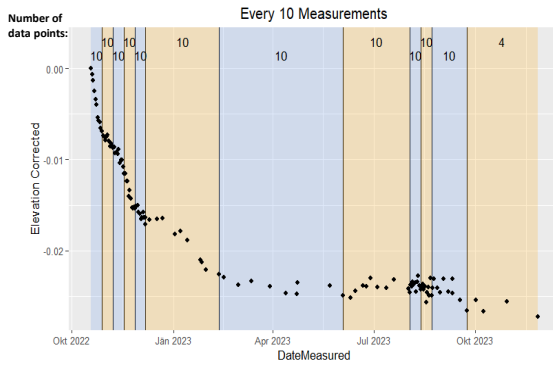


Figure 8: Visualization of a non-overlapping windowing with the size 10 measurements.

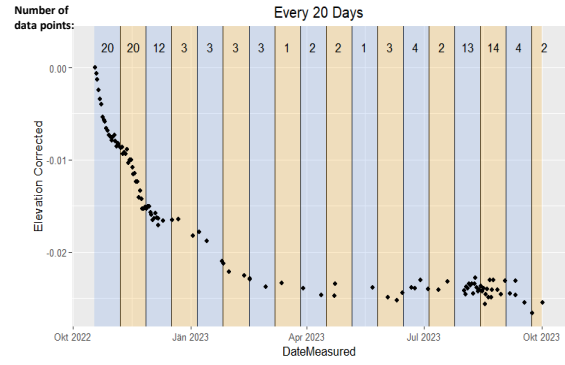


Figure 9: Visualization of a non-overlapping windowing with the size 20 days.

3.4 Data Quality Rules

Data Quality Rules form the fundamental concept of the method. They represent and apply the specific knowledge of the domain. For the given use case, the knowledge has been acquired from, and the rules have been designed with tunnelling experts. Each rule is designed to analyse one specific **believability aspect** such as timeliness, frequency plausibility, noise plausibility, etc., which are explained in the following chapters in more detail. The Believability Scores are calculated by applying rules of various types for different granularity levels.

While some rules are rather simple, some others may require more extensive data processing. Figure 10 shows an example of a more complex processing pipeline consisting of multiple processing steps and applying various transformations to the raw data. Each rule has a specific set of parameters controlling the calculation of the Believability Scores.

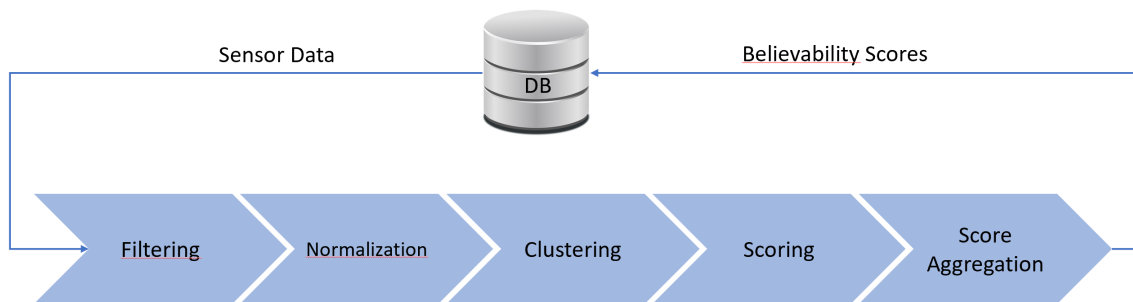


Figure 10. Example of a complex processing pipeline.

3.4.1 DQ rule - Timeliness

The ‘Timeliness’ rule checks if the time difference between two data columns is between a predefined minimum and maximum threshold. It results in “0” if the time difference is bigger than a maximum threshold and smaller than a minimum threshold. In between the two thresholds it results in scores ranging from “0” to “1” right in the middle of the two thresholds using a linear model (Figure 11).

The rule is used to analyse if the upload time (= time between data creation at the sensor and data import to the server) is believable/plausible/within a reasonable time span. In the given use case, an example is the manual upload of sensor data stemming from a sensor that must be read out manually. Assuming the sensor to be in a remote area where it takes 30min for the engineer to reach the next data upload point (a PC with internet connection), a much quicker data upload is unbelievable and indicates issues (maybe the engineer invented the data to avoid data acquisition effort). On the other hand, it might be suspicious when the data upload took longer than 6 hours because then the question arises why the engineer took so long (maybe the engineer had the need to manipulate the data before upload because it was erroneous/of low quality).

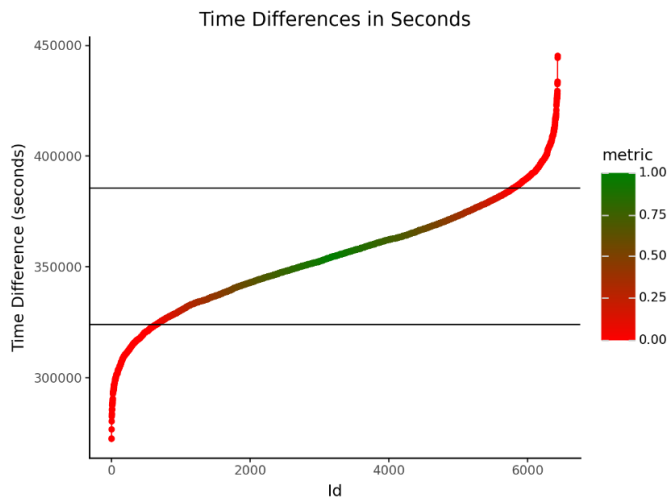


Figure 11. Time differences and their respective Believability Scores in color, sorted by time difference. Id = running number of time difference.

The parameters of the implementation are provided in Table 1.

Table 1. Parameters of the implementation in DQ rule - Timeliness

Parameters	Datatype	Description
threshold_low	float	Every result produced by the calculation that is lower than the threshold value is treated as 0 for the final score.
threshold_up	float	Every result produced by the calculation that is higher than the threshold value is treated as 0 for the final score.
window_size	int	Size of the data window used to calculate the rule. If omitted, the whole time series is used.

3.4.2 DQ rule - Frequency

The Frequency rule checks if the time difference between two adjacent data samples (measurements) is believable/plausible/within a reasonable time span. We assume a time window, in which a specific number of measurements need to take place. If a measurement is outside this window because the time difference to the previous sample is longer than expected, this measurement is of “0” data believability. Same is with a data sample (measurement) that has a too small time difference to its preceding sample. While the first case may indicate the deletion of a measurement may the second be an indication for data ingestion (e.g., by an attacker).

The parameters of the implementation are provided in Table 2.

Table 2. Parameters of the implementation in DQ rule - Frequency

Parameters	Datatype	Description
threshold_low	float	Every result produced by the calculation that is lower than the threshold value is treated as 0 for the final score.
threshold_up	Float	Every result produced by the calculation that is higher than the threshold value is treated as 0 for the final score.
window_size	int	Size of the data window used to calculate the rule. If omitted, the whole time series is used.

3.4.3 DQ rule - Noise

The Noise rule calculates if the variance of a time series is believable/plausible within a reasonable range between a predefined expected and maximum threshold. The rule calculates the differences of the data samples (sensor values) to a linear regression for a (small) window of the time series. Then the variance of the differences is calculated and assumed as “noise” (Figure 12). This calculation is done window-wise. Every data sample having a difference above the maximum threshold gets the Believability Score “0”. Starting from the expected variance threshold down to a difference of “0”, every data sample gets the Believability Score “1”. Every value between the two thresholds has a believability between “0” and “1” (linear interpolated).

The rule allows for detecting noisy measurements which may be caused by a physical attack on the sensor or an attacked sensor configuration software. The rule is to be used carefully as operational causes and security issues might not be distinguishable. Also, one needs to be cautious about the use of a linear regression model, which might not be correct in case of non-linear data behaviour/development within the time series window used for variance calculation.

The Noise rule has a second variant where an initial window is used to calculate the expected variance threshold instead of defining it as an absolute parameter by the user. This variant has the advantage that the expected variance doesn't need to be known in advance but can be determined from the data itself.

Figure 13 displays 15 different time series showing the elevations (settlements) of tunnel monitoring points (3D targets, see Figure 6). Each row shows 3 points of a cross section forming a cluster. The first column is the crown point, the second is the top left point and the third the top right point. The Believability Scores derived by the Noise rule are shown colour coded. The three time series in row 4 show a much higher noise than expected (as can be recognized by the different scale of the vertical axis) and therefore have been given a low believability.

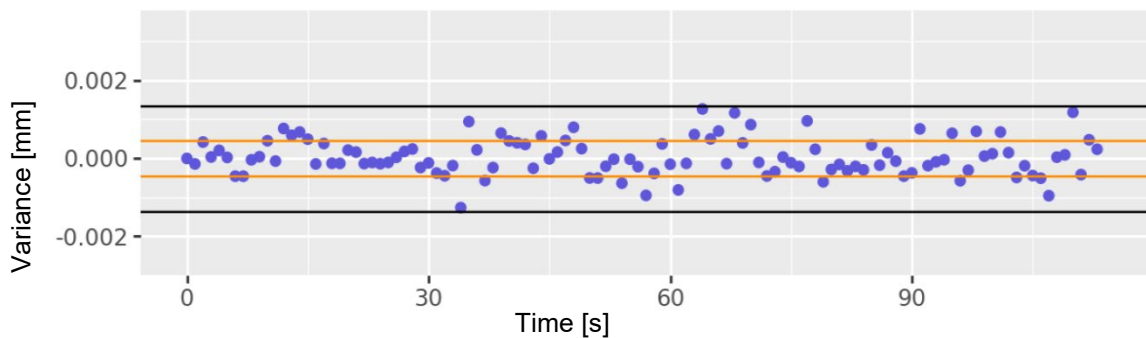


Figure 12. Differences of sensor values to their linear regression and the expected (orange) and maximum (black) variance thresholds.

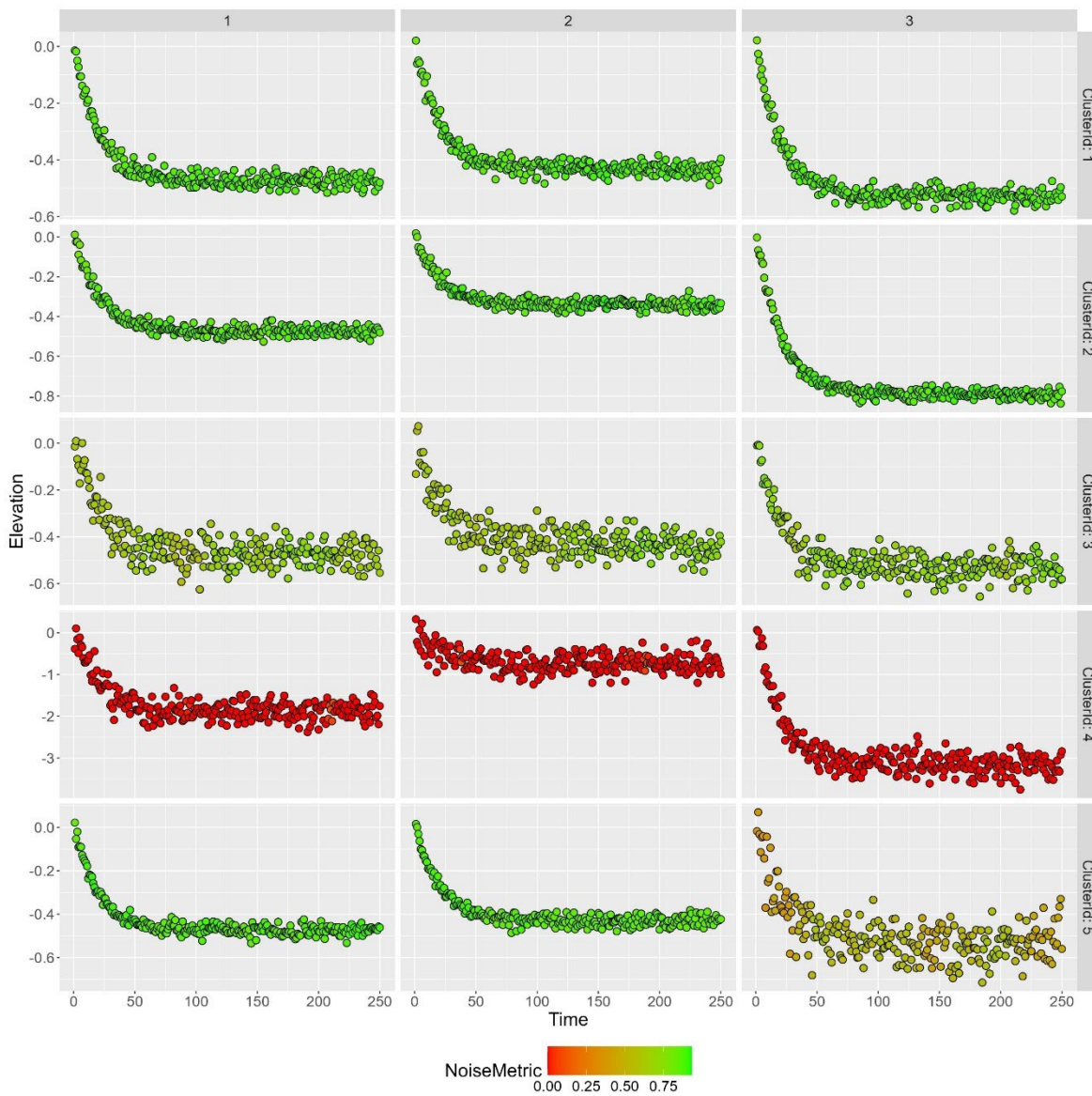


Figure 13. Believability Scores of different time series for the believability aspect Noise.

The parameters of the implementation are provided in Table 3.

Table 3. Parameters of the implementation of DQ rule - Noise

Parameters	Datatype	Description
Min threshold (= expected variance)	float	Every result produced by the calculation that is lower than the <i>min threshold</i> value is treated as 1 for the final score. Every result that exceeds the <i>min threshold</i> and lower than <i>max threshold</i> is scaled between 1 and 0. The parameter can be determined by the user or calculated using the first window of the time series.
Max threshold	float	Every result higher than the <i>max threshold</i> value is treated as 0 for the final score.
window_size	int	Size of the data window used to calculate the noise rule. If omitted, the whole time series is used.

3.4.4 DQ rule - Trend Similarity

This rule compares two time series with another to check if they have similar or different trends. The rule applies to cases where similar trends are expected or assumed logical and different trends must be assumed implausible. For example, let us compare the data from two air temperature sensors mounted close to each other. A temperature rise measured by one sensor should also be measured by the other sensor; otherwise the situation is strange/implausible. The use of this rule may make sense for sensors located close to each other, mounted on the same object, measuring the same process, etc.

The comparison of time series is carried out by **Dynamic Time Warping (DTW)**, a method for calculating the distance between time series. The method applies to entire time series, or windows of time series, but not to singular data samples. The distance between two time series is thresholded and normalized to be in an interval between 0 and 1 (= the Believability Score), where 1 represents the same time series, and 0 represents completely different series with no match whatsoever. The resulting scores are assigned to both time series, so the one having the actual problem cannot be identified by the rule. This becomes possible when applying the rule to time series clusters.

The rule can be used to compare two real time series, and to compare a real time series with a reference/pattern time series, e.g., a predicted time series (if available). A predicted time series must be provided externally, as prediction is not part of the rule.

Introduction to DTW:

DTW was first introduced by Sakoe and Chiba [1] for a speech recognition task. It is a method for estimating the distance between two sequences that do not need to be of the same length and might have significant features at different points in time (shifted on the x-axis).

This is achieved by calculating a $n \times m$ dimensional cost matrix C between two time series $X = (x_1, x_2, \dots, x_n) \in R^n$ and $Y = (y_1, y_2, \dots, y_m) \in R^m$. The elements of C are computed using a local distance function $f(x_i, y_j)$ which returns the (Euclidean) distance between two points. If a pair of points (x_i, y_j) is similar, the result of f should be small. On the other hand, if the pair is dissimilar, the result should be large.

Usually, the Euclidean distance is used for f . After calculating C , the next step is to find all possible warping paths $P = (P_1, \dots, P_k)$. One warping path is defined as $P_k = (p_1, \dots, p_l)$ where an element $p_l = (i_l, j_l)$ corresponds to an element in C . To reduce the possible number of warping paths, the search space can be limited by multiple constraints:

- **Boundary condition** makes sure that every warping path P_k starts at the beginning of the time series ($p_1 = (1,1)$) and ends with the last element ($p_l = (n, m)$).
- **Step size condition** defines how the warping path traverses through the cost matrix. This condition limits long jumps in a warping path. One simple condition may be $p_{(l+1)} - p_l \in (1,1), (1,0), (0,1)$.
- **Monotonicity condition** ensures that the time-ordering of the data is preserved.

Once the search is complete, the cost of each warping path is calculated by summing up the individual costs for each element in the warping path.

The DTW score is the cost of the warping path with minimum cost among all warping paths.

Clustering concept:

To avoid the comparison of just two time series, a clustering concept has been developed calculating the DTW distance between all time series of a user-defined cluster. A cluster may be a group of sensors or time series that are close to each other or share similar environmental or semantic conditions. A sensor can be part of multiple clusters by default, depending on the project needs.

Based on DTW, two distance-based cluster rules are implemented.

1. Inter-cluster distance, the DTW distance between all time series in a cluster.
2. Reference distance, the DTW distance between each time series in a cluster and a reference time series.

Each distance between two time series is normalized, where 1 symbolizes a small distance and 0 symbolizes an atypical distance. The minimum value for the normalized distance is 0 (the smallest DTW distance), and the maximum value is defined by the user.

All time series are pre-processed before calculating the DTW distances. The preprocessing consists of several steps, e.g., including the subtraction of the first value from all the values in the time series. For calculating the distances, the time information is ignored. Since this metric aims to compare the trends, the assumption is that, in the ideal case, the trend should be the same, even if one time series starts at day x and another one starts at day y.

Figure 14 shows examples of different time series pairs showing elevations of monitoring points (settlements) coloured by their Believability Score. As the rule can be applied using the DQ windowing technique, the scores are different depending on the windowing technique used. While the Growing-Window variant (Figure 14 left) detects a starting divergent trend rather late because the time series show similar trends over a long period in the past, the Non-Overlapping-Sliding-Window variant can detect divergent time series much quicker as it disregards the past (Figure 14 right).

Figure 15 shows the DTW results for a time series cluster of three time series. The top plot visualizes the sensor values of the whole cluster. The three plots below show the sensor values for a specific data window. The Believability Scores of the displayed data can be found above the plots. They indicate a high but slightly decreasing believability in the given windows (0.99 -> 0.97 -> 0.90).

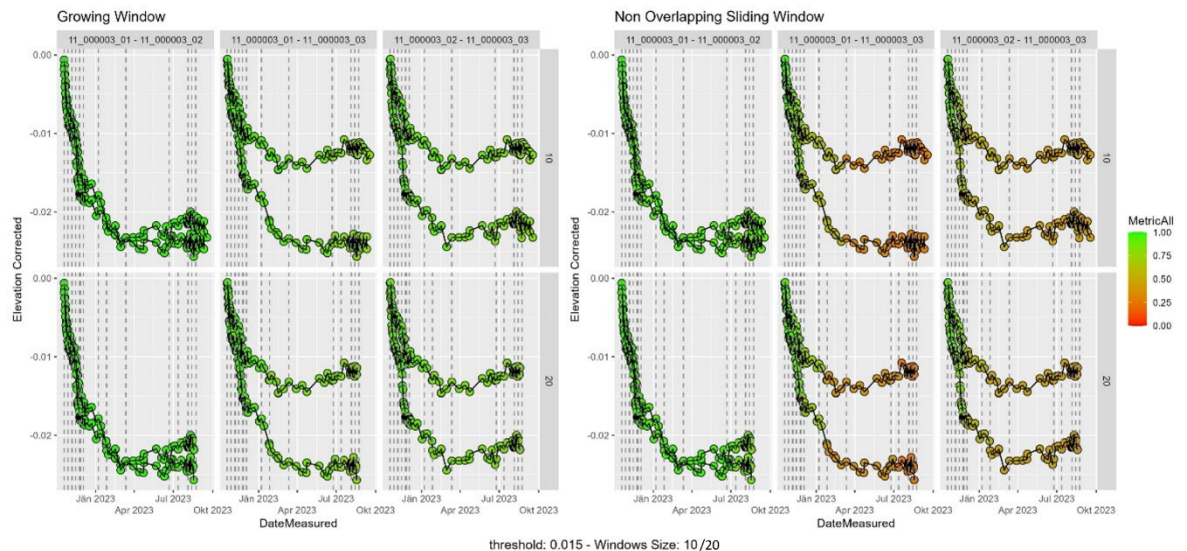


Figure 14. Believability Scores of different time series pairs for the believability aspect Trend Similarity.

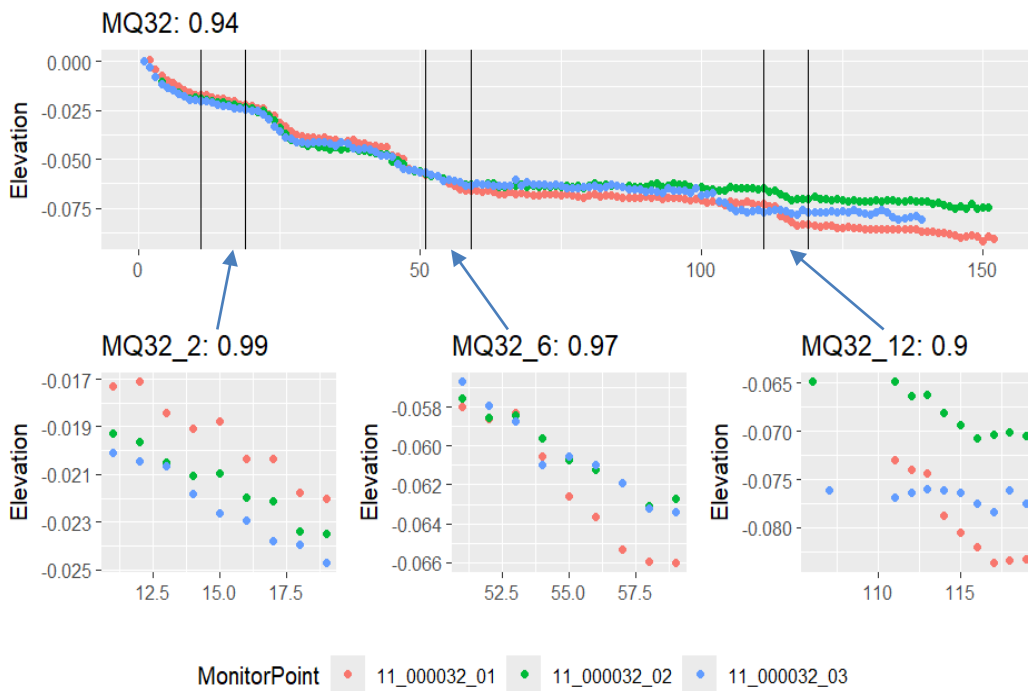


Figure 15. Believability Scores of a time series cluster consisting of three time series and for selected windows of the time series cluster.

Figure 16 shows the Believability Scores of four different time series after applying the rule comparing four real time series with a reference/pattern time series (plotted as a black line).

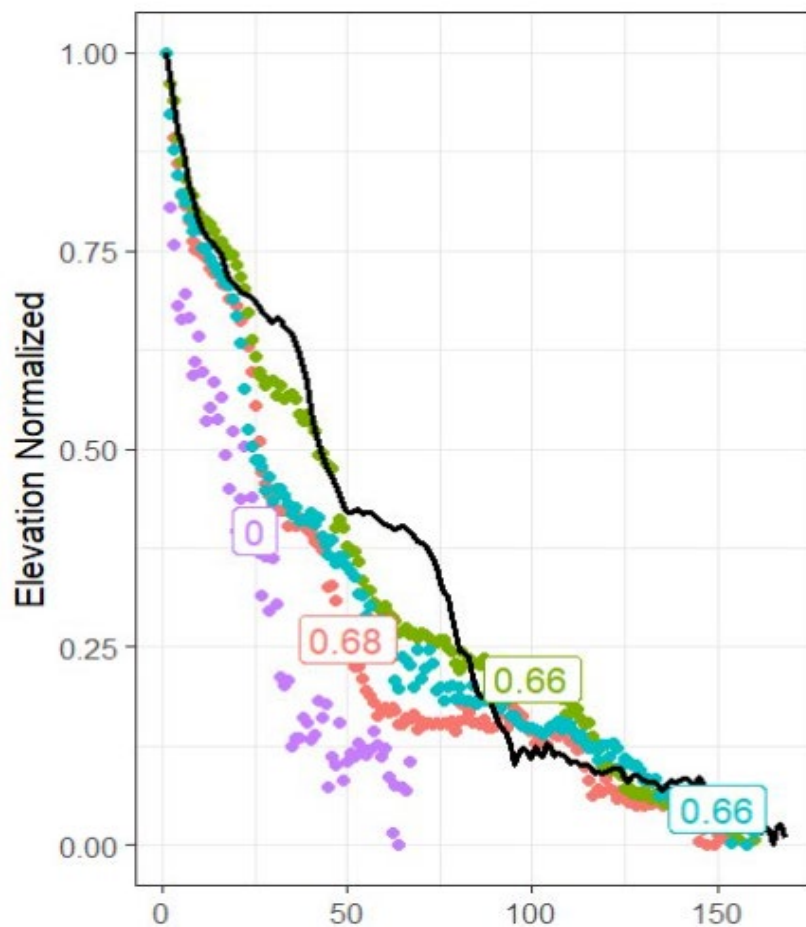


Figure 16. Believability Scores of four time series when compared with a reference time series.

The parameters of the implementation are provided in Table 4.

Table 4. Parameters of the implementation in DTW

Parameters	Datatype	Description
Threshold	float	Based on the calculation, every result higher than the threshold value is treated as 0 for the final score.
window_size	int	Size of the data window used to calculate the rule. If omitted, the whole time series is used.
ReferenceTS	data_id	This parameter is required if the Metric should be calculated between the cluster time series and a reference/pattern time series; otherwise, it can be ignored.

3.4.5 DQ rule - Change Point Similarity

Like the trend similarity rule above, this rule compares two or more time series (= a cluster) or time series windows with each other. But instead of comparing their trends, the time series are analysed if they have similar or different change points. The idea is to apply the rule in cases where change points in different time series are expected or assumed logical to occur at the same point in time, or where it must be assumed implausible if this is not the case. A simple example is the comparison of the data of a tilt sensor mounted on a train track and a nearby sensor measuring wind speed in a railway tunnel. It is plausible that a passing train always causes a small vibration/tilt change of the track and a gust of wind at the same time. The rule can be applied to time series of different sensor types and different physical parameters.

The rule first calculates the change points of the time series. Here we use the Predict and Compare (P&C) method (Figure 17 and Figure 18). The method consists of two steps. In a first step (Predict) the time series is predicted, and in a second step, the actual time series is compared to the predicted one. For the first step (Predict) we use the ARIMA method (Auto-Regressive Integrated Moving Average). The method is a statistical technique for analysing and forecasting time series data by combining three components: Autoregression (AR), Integration (I), and Moving Average (MA). For the second step (Compare), we apply the Cumulative Sum (CUSUM) method, which is a statistical quality control technique for monitoring sequential data to detect small, sustained shifts in a process's mean.

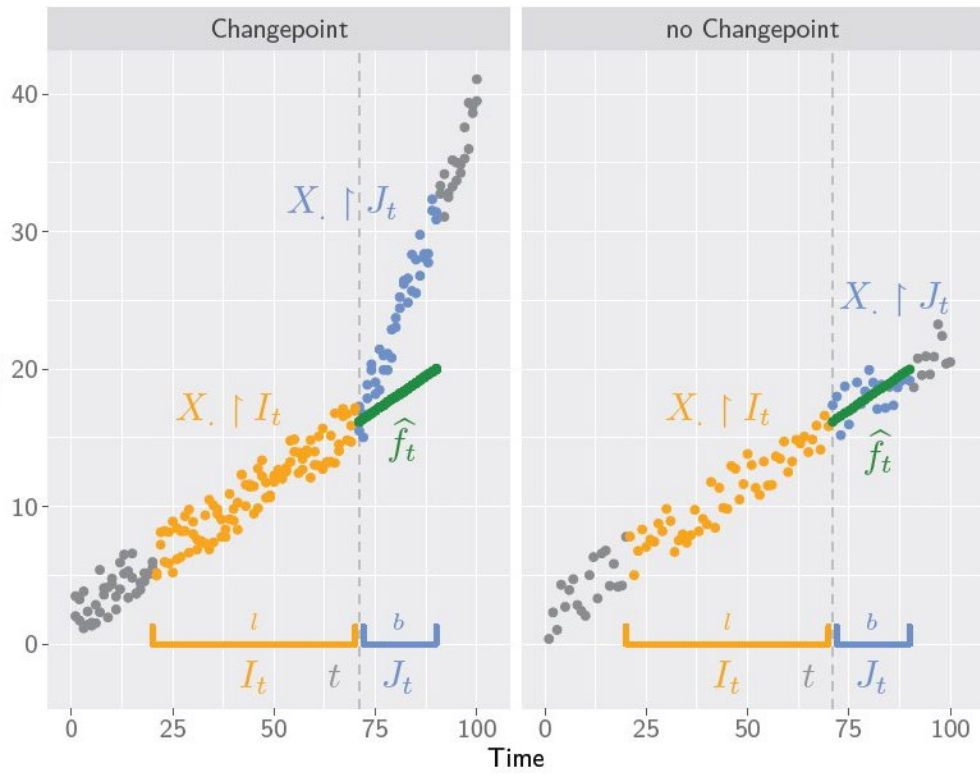


Figure 17. Principle of change point detection using the P&C method.

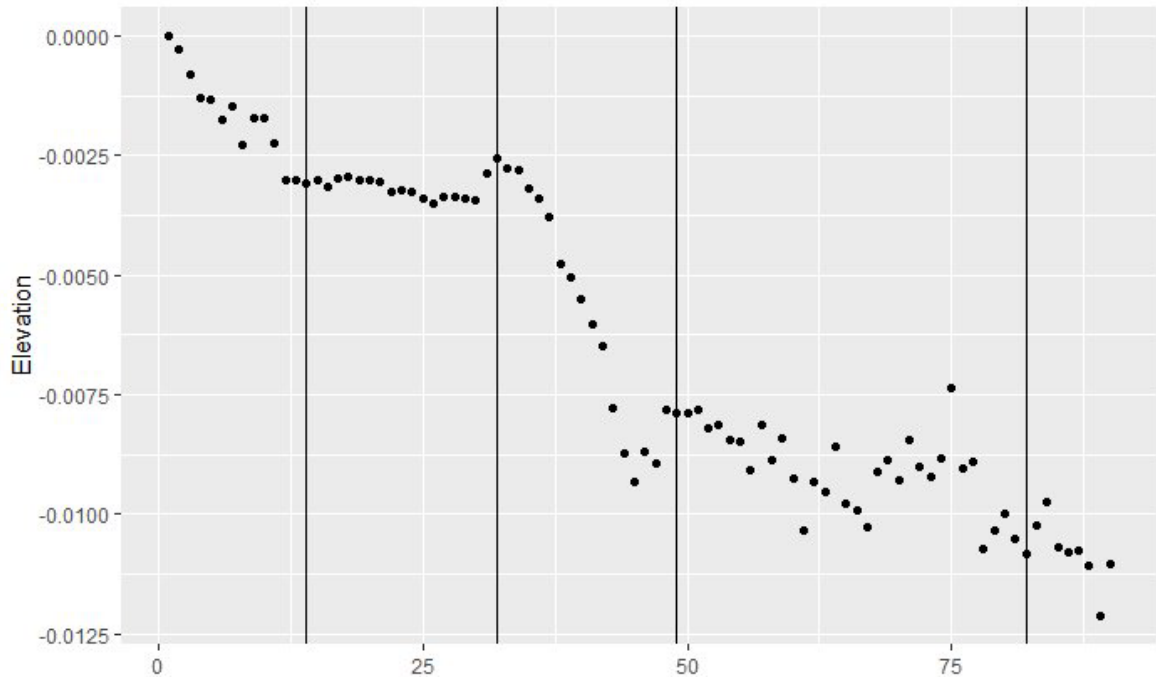


Figure 18. Change points of a time series from the tunnel construction use case detected by the P&C method.

After identifying the change points in all the time series, the rule searches for corresponding change points in these time series and calculates the time differences between them. These time differences are transformed into change point scores (one per detected change point) that are then aggregated to a Believability Score for the cluster of time series.

If there is a change point in one time series that has no correspondences in the other time series, its change point score is 0. Similarly, the same applies to time differences that are higher than a threshold.

Figure 18 presents an example of a settlement time series of a tunnel monitoring point with four detected change points.

Figure 19 illustrates an example calculation for a cluster of three time series from the tunnel construction use case. It displays the detected change points in each time series and their scores (colour coded lines). Change points without correspondences in the other time series get low scores.

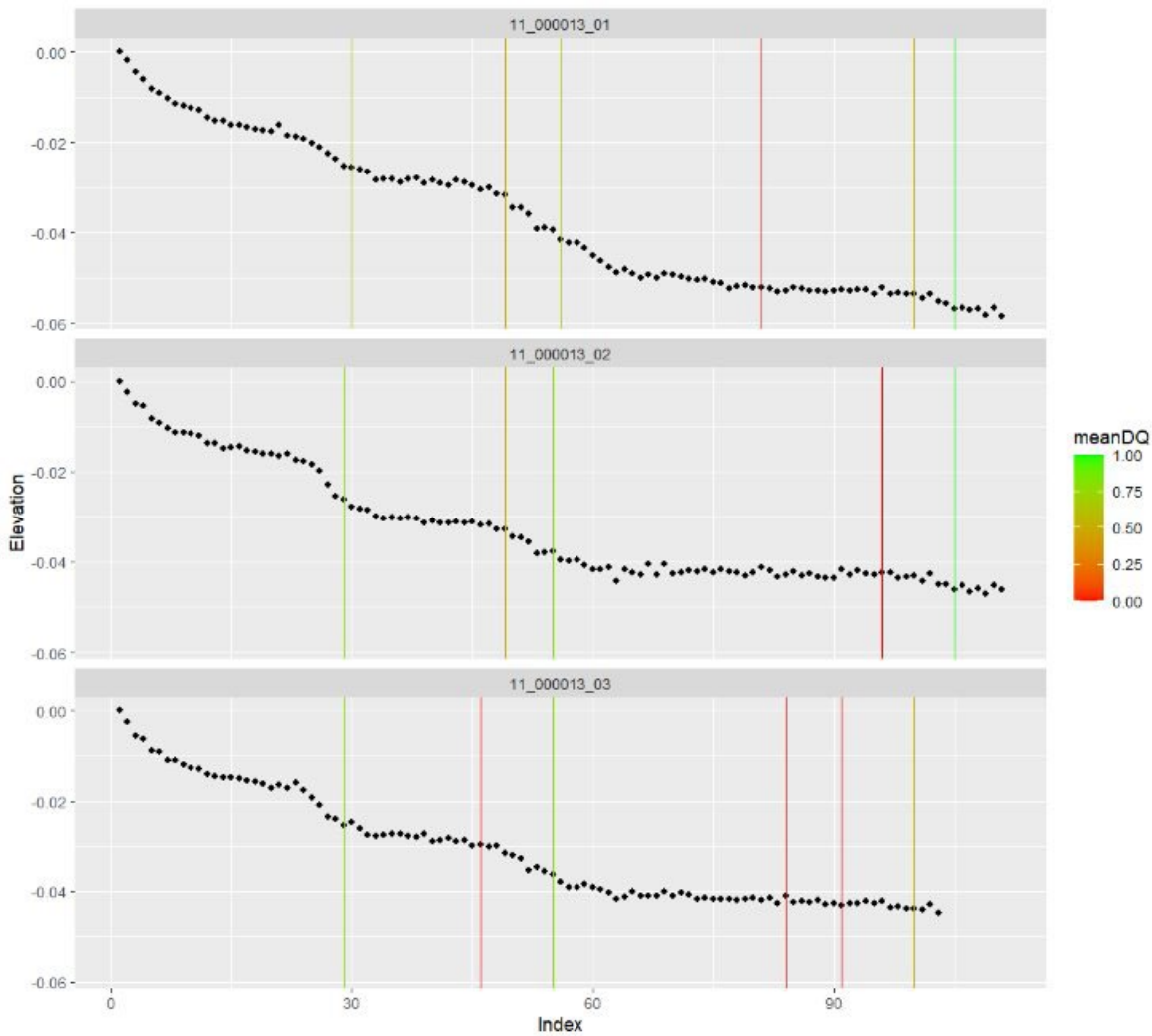


Figure 19. Change points and their change point scores for a cluster of three time series from the tunnel construction use case.

Parameters of the implementation:

Parameters	Datatype	Description
Threshold	float	Based on the calculation, every result higher than the threshold value is treated as 0 for the final score.
window_size	int	Size of the data window used to calculate the rule. If omitted, the whole time series is used.
Predict	-	Set of parameters required to control the Predict method ARIMA
Compare	-	Set of parameters required to control the Compare method CUSUM

3.5 AI approach

The rules presented in the previous chapters are based on classical, empirical mathematical/statistical models. In what follows, two AI approaches that have been researched to explore the believability of data are discussed.

3.5.1 Prediction Transformer

In Section 3.4.5, the rule Change Point Similarity makes use of the ARIMA method for the prediction of time series. However, several other mathematical/statistical methods exist and can be applied for forecasting sequential data.

An AI-supported option to do so is the use of Prediction Transformers, which is a class of deep learning models that use self-attention mechanisms to learn temporal dependencies and patterns in sequential data. Transformers can predict future values based on historical time series inputs as they are trained to capture and understand patterns and features of time series data. Transformers do not necessarily outperform basic forecasting tools like ARIMA or entirely data-driven approaches. They require high-quality training data of a sufficient amount, which is not always available in a domain.

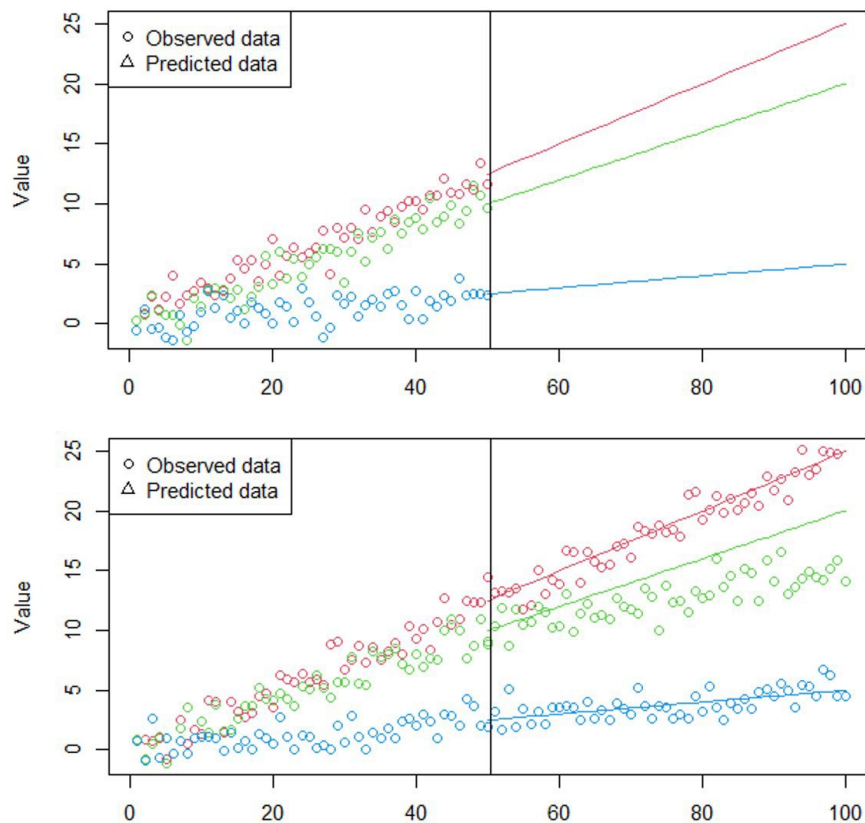


Figure 20. Predicted (upper image) and observed data (lower image) using a Prediction Transformer.

For the tunnel construction use case, the approach has been researched using available/already pre-trained transformers (e.g., TiRex [2,3,4], Lag-Llama [5,6] and N-BEATS [7,8]). **TiRex** is a pre-trained, 35-million-parameter foundation model for time series forecasting, developed by NX-AI and introduced at NeurIPS 2025. Its key appeal is zero-shot forecasting: you can point it at any new time series data and get predictions immediately, without any fine-tuning or retraining on your own dataset.

Under the hood, TiRex is built on xLSTM — an enhanced version of the classic LSTM architecture — rather than the transformer models that dominate most recent forecasting research. This is a deliberate choice: transformers struggle with the kind of long-range state tracking that good time series forecasting demands, while LSTMs excel at it but historically lacked strong in-context learning. xLSTM closes that gap, giving TiRex the best of both worlds.

The model outputs both point estimates and quantile predictions, giving you uncertainty ranges alongside a central forecast. It performs well across both short- and long-horizon tasks, topping the GiftEval and Chronos-ZS benchmarks — outperforming much larger models from Amazon, Google, and Salesforce.

For the TiRex transformer an AI Data Quality Rule has been developed. Let's assume, we have a time series TS_1 consisting of 50 data points, as shown in Figure 21. The time series is a simple linear function

$$x * k + d$$

with white noise added. k stays the same for all 50 points. From point 0 to 34, d is 1, from point 35 onward, d is 5, generating a sudden shift/jump along the Value axis. In Figure 21 two windows are marked, which we will use to show how the metric calculation works in case of an error (W_{35}) or maybe a data tampering attack and a case without an error (W_{18}). One of the parameters of the metric is the length of the context window (which the prediction model uses to forecast), another one is the prediction length (which is the length of the forecast). The data provided to this metric must be at least as long as the context length and the prediction length. If the data is longer than the context + prediction length, a sliding window approach is used to analyze all data points. In this example the context length is 10 and the prediction length is 3.

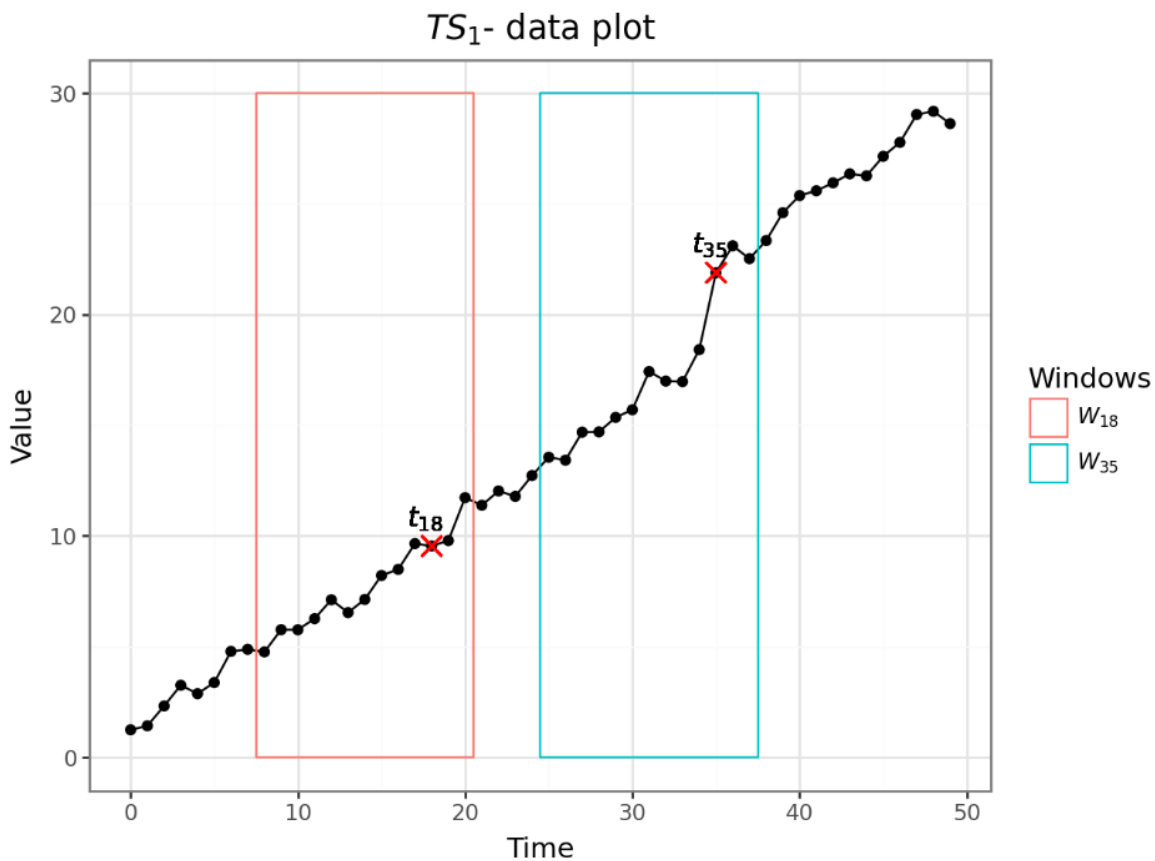


Figure 21. A time series with a change in the function at point 35.

After obtaining a window, the context part is used to predict the next 3 data points (in our example). Figure 22 shows the prediction for W_{18} and W_{35} . The red points are the context window, the blue points are the true values in the prediction window and the green points are the forecast of the prediction model.

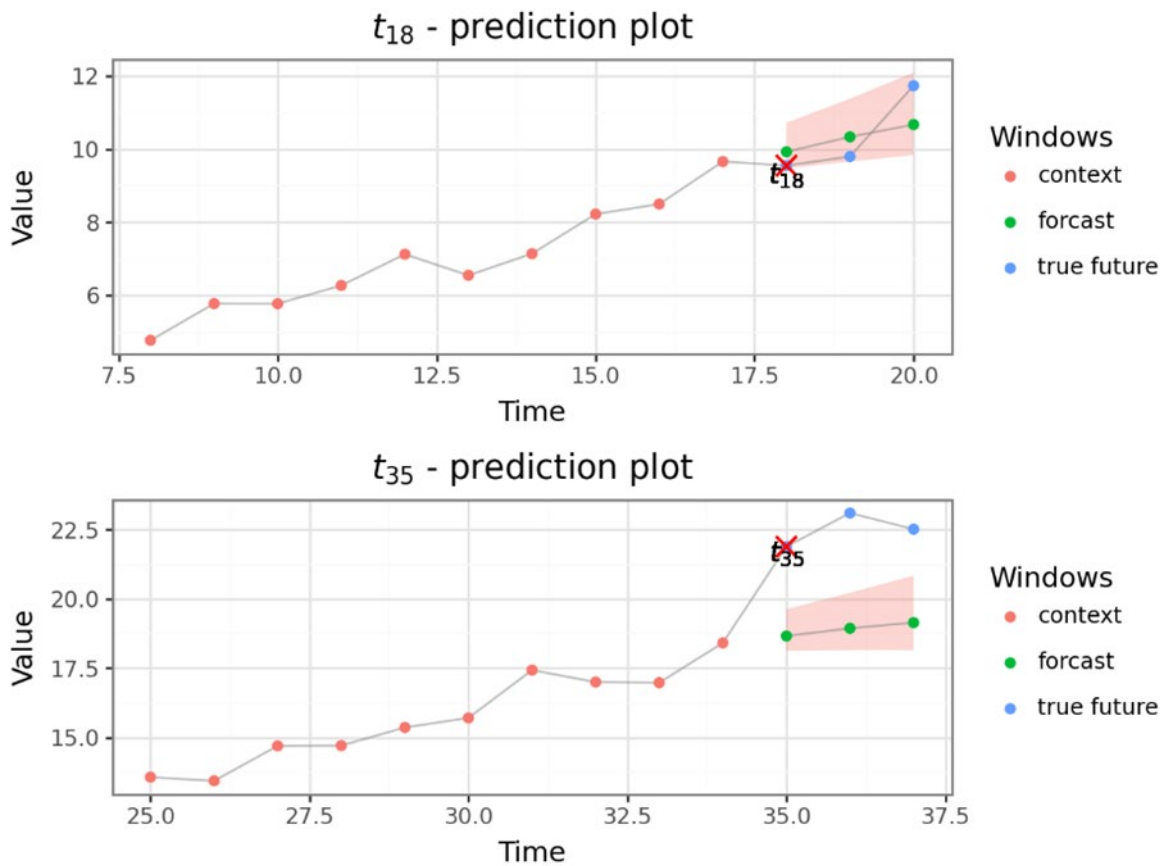


Figure 22. Prediction for window 18 and window 35.

TiRex is the prediction model we use, and it provides the mean value along with a convenient interval around it. After the general explanation we look at TiRex in more detail. This confidence interval is shown in transparent red around the forecast values and accounts for potential noise in the data. After obtaining the forecast from the prediction model, it is compared with the real data points for the same time period. This comparison can be made using Root Mean Squared Error (RMSE) or DTW. For our example, we will use RMSE. RMSE is a typical measure of the error a predictive model makes.

$$RMSE: \sqrt{\frac{\sum_{i=1}^n (true_{values_i} - prediction_{values_i})^2}{prediction\ window\ length}}$$

Figure 23 shows the results for the whole time series and the thresholds that indicate if the RMSE value indicates a problem in the data or not. There are two thresholds: a lower one (a dashed line in violet) and an upper one (a dotted line in blue). All values above the upper threshold indicate an error in the data and result in a metric value of 0. Between the upper and lower thresholds, the RMSE indicates that there might be an error in the data. The closer to the upper border, the likelier and the more significant the error is. RMSE values between the two borders are mapped to values between 0 and 1. Below the lower border, everything is fine, the time series behaves as expected, all values are assigned 1, the best possible result. The value for each window is assigned to the first point in the prediction window.

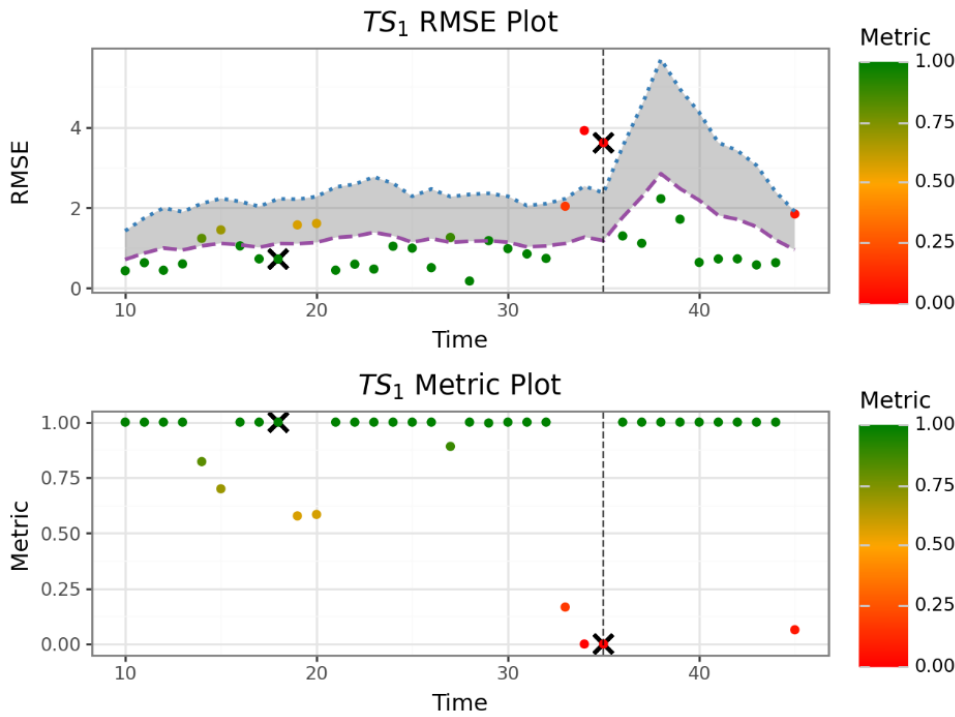


Figure 23. RMSE values and the thresholds in the top plot. The bottom plot shows the believability score of each point. The color represents each point's score in both plots. The dashed grey line represents the point where the change happened in both plots.

Figure 23 shows the RMSE values and their thresholds for TS₁, along with the believability scores. The values of W_{18} and W_{35} are marked by a black X. The value of W_{18} shows that everything is ok at that point in the time series. The W_{35} metric value is 0, indicating that something unexpected happened. This was already indicated by the scores for W_{33} and W_{34} . W_{33} to W_{35} contains the jump in the prediction window but not in the context window. In contrast, W_{36} onwards have lower scores, as in these context windows the jump in the d value is already included.

Figure 24 shows how each point is rated by the AI metric.

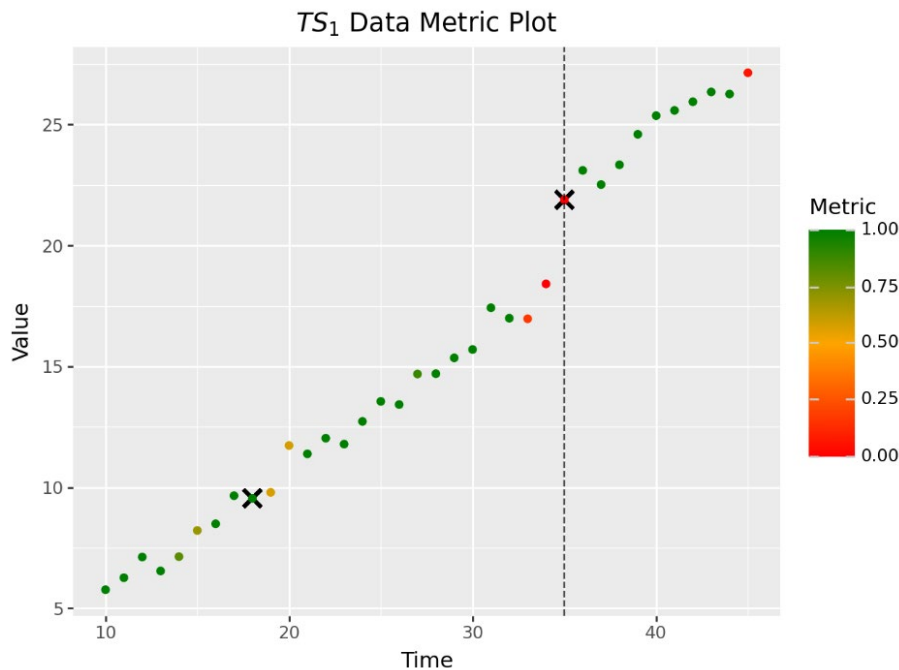


Figure 24. TS₁ values with the color representing the believability score for each point.

The given example shows that time series predictions can be done by pre-trained prediction transformers like TiREX and used in data quality rules comparing predicted with true data and scoring their differences. They have been tested to be useful but not necessarily ‘better’ than classical prediction models like ARIMA. From a practical point of view, pre-trained transformers have a huge advantage compared to, e.g., machine learning approaches as they do not require training data. Especially for shorter tunnel projects that might already be finished before sufficient training data has been collected, this is seen advantageous.

However, the behaviour of time series data of a tunnel project (e.g., tunnel deformation data) is influenced by numerous external factors like geology, excavation phases and progress, tunnel dimension, and tunnel support, that might change dynamically and abruptly, not be known at all. All these factors lead to many trend changes/change points, shifts and jumps in tunnelling time series that are hard to predict (as an example see Figure 18).

3.5.2 Large Language Models (LLMs)

LLMs are generative AI models trained with self-supervised machine learning on a vast amount of text, designed for natural language processing tasks. The currently largest and most capable LLMs are generative pre-trained transformers (GPTs), based on a transformer architecture, which are largely used in generative chatbots such as ChatGPT, Gemini and Claude.

In our research we used the LLM **Claude Sonnet 3.5 and 3.7** by Anthropic to query time series data for the presence of “suspicious” patterns. As LLMs in general are not good at calculations but learn to predict the most probable next text/character, the given task clearly is a challenge for an LLM as it requires the calculation of huge amounts of numerical data.

A further restriction is imposed by the rather small amount of data (context length) that can be sent to the LLM to be used for reasoning as well.

To overcome these problems, we use the **Model Context Protocol (MCP)**. MCP is an open-source standard created by Anthropic to enable LLMs and AI agents to seamlessly integrate with and access external tools, data, and systems, like how an Application Programming Interface (API) connects services. By providing a standardized, two-way communication framework, MCP allows AI systems to go beyond their initial training data, incorporate real-time information, perform actions, and access specialized features.

Despite its advantages, MCP also has its own challenges. A significant problem lies in dealing with failed queries. If a request cannot be processed successfully, the system initiates a new query,

possibly several times in succession. This can quickly lead to increased costs, especially if paid services or API calls are involved.

Another limiting factor is the latency in complex queries. Depending on the type and scope of the requested data or the utilization of the connected systems, the processing time can vary considerably. These delays can impair the user experience and reduce the overall system efficiency. In addition, the fundamental problem of context length is still an issue in certain scenarios. For queries that generate extensive responses, the returned data might exceed the available context length of the LLM.

In our case, the MCP allows the LLM to work even with a huge time series database by sending it to external services for processing numerical parameters, collecting them, and processing them with other services. In this way, the LLM serves more as a data handler and translator of results to natural language that are processed by external services in numerous sequential steps.

Communication with the LLM is realized by natural language via **LLM prompts**. LLM prompts are natural-language instructions that guide the LLM to generate a specific output, like a text, code, or answer. The quality of an LLM's response is often determined by the clarity and detail of the prompt, so crafting effective prompts through techniques like prompt engineering is crucial for achieving desired results. Prompts can range from simple questions to detailed instructions, contain domain knowledge or even assigned roles for the LLM. When the LLM's answer(s) exceed the context length of the LLM, this would lead to an error message. Therefore, a suitable prompting asking for concise answers is recommended.

For our use case, we decided on a three-prompt approach. Prompt 1 clarifies the role and topic, Prompt 2 provides domain knowledge, Prompt 3 asks to solve a problem (the actual question).

Below is shown a concrete example of the three prompts asking for the exploration of a tunnel monitoring database of a past tunnel project containing 347 settlement time series of 347 monitoring points to find peculiar or unbelievable settlements.

Prompt 1: Role and topic clarification

You are an AI assistant that uses MCP to access a Tunnel Settlement Measurement SQLite database. Your topic is: Tunnel Settlement Monitoring. Remember to maintain consistency throughout the scenario and ensure that all elements (tables, data, queries, dashboard, and solution) are closely related to the given topic. Implore to make all outputs as human-readable as possible but remain concise

Prompt 2: Domain knowledge, background information

Here is some background information about the linked database: The data represent the accumulated settlements of monitoring points located in a tunnel. The monitoring points are designed as geodetic prism targets installed in the roof of the tunnel. The monitoring points were measured during the construction of the tunnel by a surveyor applying the optical 3D displacement monitoring method. Zero readings were measured for the first time and therefore have a value of zero. Most settlement values will be negative; however, users often imply absolute settlement values. For example, the biggest increase between two settlement measurements is typically the absolute value $|x_1 - x_0|$, where x_0 is the first measurement and x_1 is the second measurement. Always call measurements x_0, x_1, \dots in your queries

Prompt 3: The task, the question

Do you find any notable anomalies or outliers in the settlements?

After starting, the LLM first analyses the structure of the database, queries it several times, and after about 7 minutes comes back with the following answer:

LLM answer:

*Based on my analysis of the settlement data, I've identified several notable anomalies in the settlement pattern of **monitoring point 161**:*

1. Positive Settlement Spike on August 31, 1991: *The most significant outlier appears on August 31, 1991, where there's a sudden positive settlement change of +0.0544m (from -0.8044m to -0.7500m). This represents an upward movement (heave) with a daily rate of +0.0568m/day, which stands in stark contrast to the overall settlement trend. This appears to be a measurement anomaly or error, as it deviates dramatically from the consistent settlement pattern observed before and after this date.*

2. Initial Rapid Settlement Period (June 12-15, 1991): *At the beginning of measurements, there's an unusually high settlement rate with daily rates between -0.1695m/day and -0.0356m/day. This initial rapid settlement occurs during the first few days of monitoring, suggesting either an initial adjustment period or potentially construction-related settlement.*

3. Irregular Measurement Gap and Settlement Recovery After August 31, 1991: *Following the anomalous positive settlement on August 31, there's a 17-day gap until the next measurement on September 17, where the settlement jumps back to -0.8161m. This represents a significant negative change (-0.0661m) after the previous positive spike, suggesting a correction or return to the expected settlement pattern.*

4. Accelerated Settlement in Final Period (September 2-25, 1992): *Towards the end of the monitoring period, there are several accelerated settlement events:*

** September 1-8, 1992: -0.0129m over 9 days (-0.00143m/day)*

** September 8-16, 1992: -0.0262m over 8 days (-0.00328m/day)*

** September 16-25, 1992: -0.0346m over 9 days (-0.00383m/day)*

This represents a significant acceleration compared to the previous months, with the daily settlement rate increasing by approximately 3-4 times. The most prominent outlier is clearly the August 31, 1991 positive settlement spike, which is inconsistent with the overall settlement trend and likely represents a measurement error or anomaly. The subsequent correction on September 17 suggests this was indeed an anomalous reading rather than a genuine settlement behavior.

The accelerated settlement in the final monitoring period (September 1992) could be indicative of a changing ground condition, potentially related to seasonal factors, groundwater changes, or additional construction activities in the vicinity that might have influenced the settlement behavior at position 161.

To provide more information the LLM is prompted again to provide a visualization of the findings:

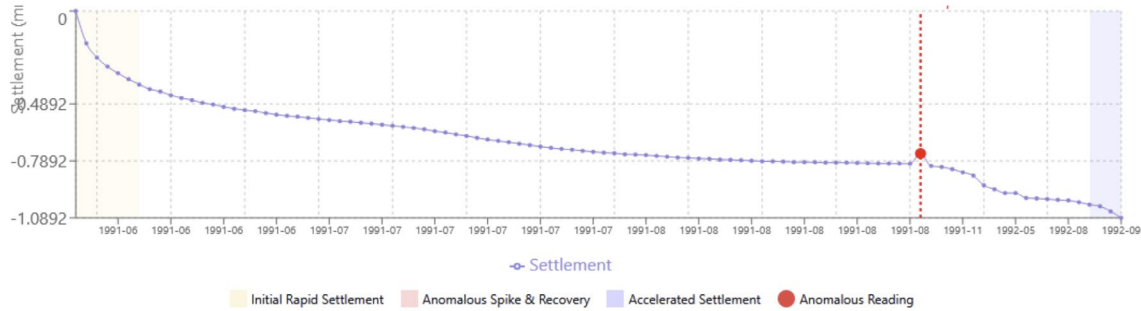
Another Prompt: Requesting visualization

Can you plot the findings?

The LLM takes about 3 minutes to come back with the following answer:

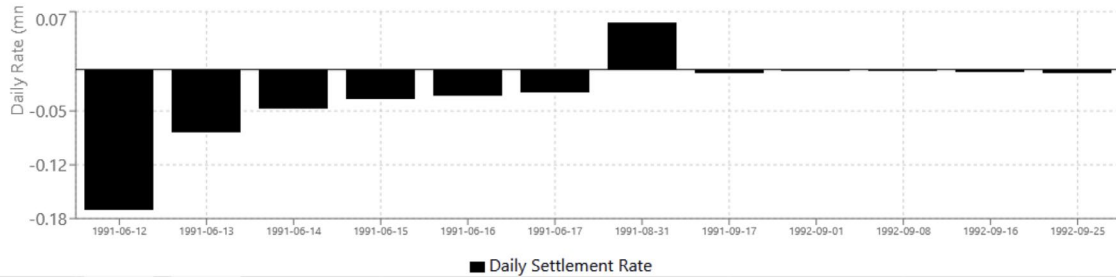
Settlement Over Time

This chart shows the complete settlement history for position 161, highlighting the anomalous readings and key periods of interest.



Daily Settlement Rate

This chart shows the daily rate of settlement for key periods, highlighting the initial rapid settlement, anomalous spike, and final acceleration.



Key Findings

- **Major Anomaly (August 31, 1991):** A significant positive settlement spike of +0.0544mm with a daily rate of +0.0568mm/day, inconsistent with the overall trend.
- **Initial Rapid Settlement (June 11-17, 1991):** Extremely high settlement rates at the beginning of monitoring, ranging from -0.1695mm/day to -0.0275mm/day.
- **Measurement Gap & Recovery (Aug 31 - Sep 17, 1991):** Following the anomalous spike, there's a 17-day measurement gap and then a return to the expected negative settlement pattern.
- **Accelerated Final Settlement (Sep 1-25, 1992):** The settlement rate increased in the final period, with the daily rate approximately 3-4 times higher than previous stable periods.

I've created a detailed visualization of the settlement data for position 161, which clearly highlights the anomalies we identified. The dashboard includes:

1. Settlement Over Time Chart:

- * This main chart shows the complete settlement history for position 161 (located 3274m from the tunnel ridge)
- * The anomalous reading on August 31, 1991 is highlighted in red, showing the sudden positive spike
- * Three key periods are highlighted with colored zones:
 - Yellow zone: Initial rapid settlement (June 11-17, 1991)
 - Red zone: Anomalous spike and recovery period (Aug 30 - Sep 18, 1991)
 - Blue zone: Accelerated settlement period (Sep 1-25, 1992)

2. Daily Settlement Rate Chart:

- * This bar chart shows the daily rate of settlement (mm/day) for key periods
- * The extreme initial settlement rates and the anomalous positive spike are clearly visible

3. Key Findings Summary:

- * A concise list of the main anomalies is identified in our analysis

The visualization confirms that the most prominent outlier is indeed the August 31, 1991 reading, which shows an unexpected upward movement (positive settlement) of +0.0544m. This is completely inconsistent with the overall negative settlement trend and likely represents a measurement error.

Other notable patterns include the extremely rapid initial settlement in June 1991, and the accelerated settlement in the final monitoring period of September 1992. These patterns might reflect genuine changes in ground conditions or construction activities affecting this monitoring point.

The received answers and created visualizations prove that LLMs already can provide a surprisingly high level of expertise and knowledge depth when dealing with time series data in the tunnelling domain. The given answers of the given example are correct.

In our research we conducted several more tests (other data, other promptings) that did not all deliver such promising results. Context length errors and wrong answers happened occasionally, and hallucinations were experienced.

An optimal prompting causing the LLM to focus on security-related anomalies or to clearly distinguish between operational and security-related anomalies is still an issue to be answered, same is with the transfer of LLM answers to Believability Scores. Therefore, further research efforts are needed to implement the AI approaches into the Data Quality Verification System prototype developed in the framework of CISSAN.

4 Software Prototype

This chapter presents a comprehensive overview of the software architecture, task distribution, programming languages, platforms, and selected software packages implemented on the Data Quality Verification System.

4.1 Components and Their Software Implementations

The prototype consists of three main components, each with distinct tasks, roles, and technical implementations.

4.1.1 GeodataHub

Role & Tasks:

- Provides a central platform for managing and visualizing the measurement data (time series).
- Provides user interfaces for project configuration, data exploration and visualisation, reporting, and analytics.
- Maintains long-term storage, data integrity, and comprehensive access control.
- Sends a JSON request to the DQ Verification Component specifying and triggering the DQ analysis
- Receives a JSON answer from the DQ Verification Component containing the DQ analysis results (Believability Scores).
- Manages (stores, visualizes) the DQ analysis results (Believability Scores)

Software Platform & Language:

- Frontend developed in Angular/TypeScript as Single Page Web App (SPA)
- Backend developed in .NET/C# (REST/jsonRPC API and business logic) running as Azure Web App
- Databases: PostgreSQL (relational data), Neo4J (graph and relationship data) and Blob Storage on Azure Cloud infrastructure.

Key Implementation Details:

- Implements advanced data models for measurement types, provenance, and time-series analytics.
- Supports integration with external systems via standard APIs.
- Role-based access and authentication leveraging OAuth/OpenID.

4.1.2 DQ Verification Component

Role & Tasks:

- Receives JSON request from GeodataHub and runs DQ analysis
- Loads DQ rules from DQ Repository and data from GeodataHub
- Sends JSON answer with DQ analysis results (Believability Scores) back to GeodataHub

Software Platform & Language:

- Python 3.1

Key Implementation Details:

- Virtual environment to manage the dependencies. The following table shows the used dependencies and a basic description of why they are included.

• Dependency	• Description
• Pandas	• Data management engine
• scikit-learn	• Machine learning library, used for clustering
• Pytest	• Unit-Testing framework
• Mkdocs	• Doc generation framework

4.1.3 DQ Repository

Role & Tasks:

- Stores DQ rules
- Provides DQ rules to DQ Verification Component

Software Platform & Language:

- .NET/C#

Key Implementation Details:

- Implemented as a module inside of GeodataHub
- All DataQuality Clusters are stored in a neo4J graph database

4.1.4 JSON Structure

The DQ analysis is activated when sending a JSON request from GeodataHub to the DQ Verification Component in the following structure (example).

```
{
  "Cluster": {
    "Root": {
      "contents": [
        "Branch_A",
        "Branch_B"
      ],
      "data": [],
      "metrics": [],
      "aggregation": [
        {
          "aggFuction": "weight_avg",
          "parameter": [
            {
              "ClusterName": "Branch_A",
              "value": 0.4
            },
            {
              "ClusterName": "Branch_B",
              "value": 0.6
            }
          ]
        }
      ]
    }
  },
  "Branch_A": {
    "contents": [
      "Leaf_AA",
      "Leaf_AB"
    ],
    "data": [],
    "metrics": [
      {

```

```

        "metric": "FrequencyMetric",
        "parameter": [
            {
                "window_size": 2,
                "threshold": 2,
                "target_column": "DateMeasured"
            }
        ]
    },
    ],
    "aggregation": "avg"
},
"Branch_B": {
    "contents": [
        "Leaf_BA",
        "Leaf_BB"
    ],
    "data": [],
    "metrics": [
        {
            "metric": "FrequencyMetric",
            "parameter": [
                {
                    "threshold": 1,
                    "target_column": "DateMeasured"
                }
            ]
        },
        {
            "metric": "Noise",
            "parameter": [
                {
                    "threshold": 2,
                    "target_column": "DateMeasured"
                }
            ]
        }
    ],
    "aggregation": "min"
},
"Leaf_AA": {
    "contents": [
        "ID_AA_1",
        "ID_AA_2"
    ],
    "data": [],
    "metrics": [
        {
            "metric": "FrequencyMetric",
            "parameter": [
                {
                    "threshold": 1.6,
                    "target_column": "DateMeasured"
                }
            ]
        }
    ]
},
"Leaf_AB": {
    "contents": [],
    "data": [
        "ID_AB_1",
        "ID_AB_2",
        "ID_AB_3"
    ]
},
"Leaf_BA": {
    "contents": [],
    "data": [
        "ID_BA_1"
    ]
}

```

```

    },
    "Leaf_BB": {
      "contents": [],
      "data": [
        "ID_BB_1",
        "ID_BA_2"
      ]
    }
  },
  "Data": {
    "ID_AB_1": {
      "channelId": "a4947842-407e-4617-b574-697f898f1fb0",
      "monitorPointId": "f4a91ecb-6fa3-4086-bc38-d5e8192b6d8d",
      "returnValueDataTypeId": "b23d6ab9-22c3-4d54-8bbb-
13231b8b953a",
      "singleValueDimensionId": null
    },
    "ID_AB_2": {
      "channelId": "a4947842-407e-4617-b574-697f898f1fb0",
      "monitorPointId": "f4a91ecb-6fa3-4086-bc38-d5e8192b6d8d",
      "returnValueDataTypeId": "b23d6ab9-22c3-4d54-8bbb-
13231b8b953a",
      "singleValueDimensionId": null
    }
  }
}

```

4.2 Software Package Summary

- Programming Languages: .NET/C# (custom modules), Python 3.1 and Bash (auxiliary scripts).
- Databases: PostgreSQL (structured data), Neo4J (graph data).
- Cloud Platform: Azure Cloud (for GeodataHub backend and databases deployment).

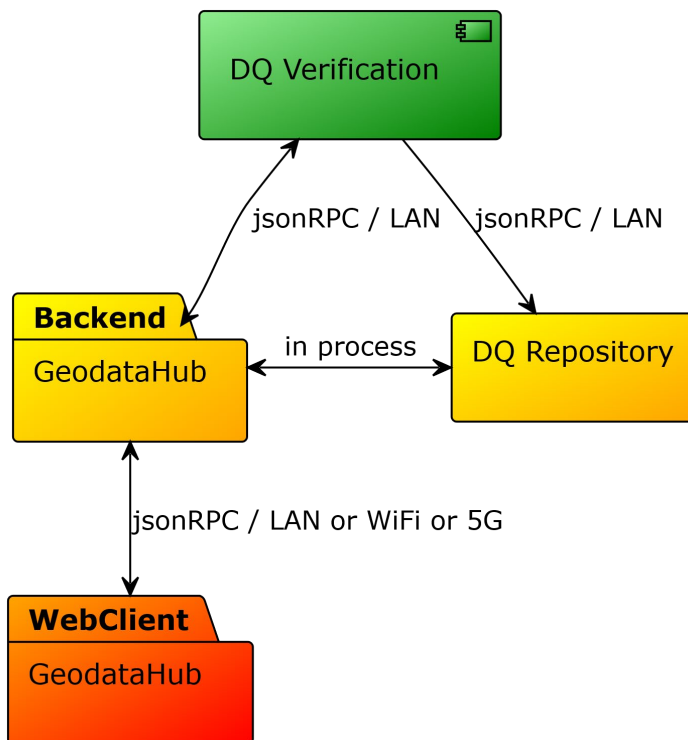


Figure 25. DQ Verification System, software prototype architecture, used communication protocols, data formats, and networks.

Figure 26 displays the Believability Scores processed by the developed software prototype for a time series in the IoT platform GeodataHub. In this example two Data Quality Rules, the Frequency Rule and the Noise Rule have been applied, and a total Believability Score has been aggregated from the two rules. The time series presented in Figure 26 (upper diagram) clearly shows a time period where data is missing, followed by a noisy period and an outlier sequence at the end. As a result, its believability, shown in Figure 26 (lower diagram), has decreased with time. The lower diagram displays the scores of the Frequency Rule (blue line), the Noise Rule (green line), and an overall score aggregated from both using the average function (yellow line). The scores are calculated at four points in time, indicated by black dots in the upper diagram and the corresponding rectangles in the lower diagram. They are (1.) at a beginning part of the time series where all seems right and all scores are high (>0.8), (2.) shortly after the missing data part where the Frequency Rule causes the blue score to decrease from 1 to 0, (3.) shortly after the noisy part, where the Noise Rule causes the green score to decrease from 0.8 to 0.6, and (4.) after the outlier part where the Noise Rule another time causes the green score to decrease from 0.6 to 0.17. The rules were applied using the growing-windowing technique meaning that the time series is analysed from its start to the given points in time (black dots).

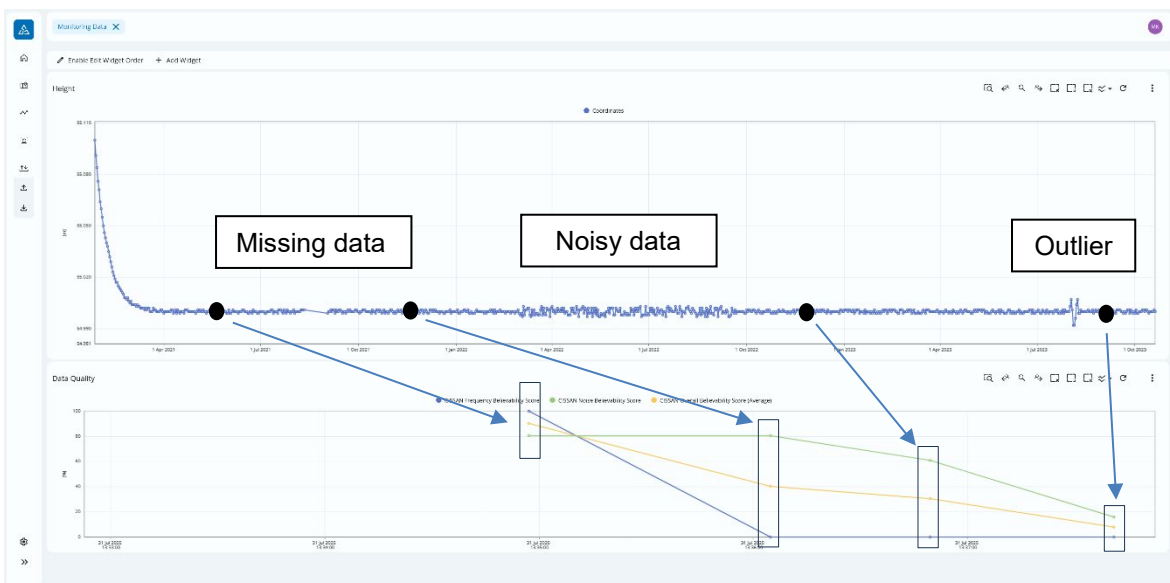


Figure 26. Sensor time series (upper diagram) and its Believability Scores (lower diagram) displayed in a dashboard of the IoT platform GeodataHub.

5 Conclusion

The Data Quality Verification System described in this report demonstrates a systematic and empirical approach to determine the data quality dimension 'believability'. By further developing and connecting an existing IoT platform used for data management with a newly developed DQ Verification software, the system can determine Believability Scores and disseminate them to the platform and its users. The resulting scores are used for alarming, reporting, and other required actions.

The Believability Scores are processed by domain-specific Data Quality Rules that apply special mathematical/statistical methods and sophisticated concepts such as dynamic time warping, clustering, windowing and aggregation. They are assigned at different granularity levels to single sensor values, a sequence of sensor values (= a time series window), an entire time series, a group/cluster of several time series and all the time series of a given monitoring project (total data health, total believability).

Rules have been developed for analysing aspects like the timeliness, noise, frequency, trend similarity and change point similarity of sensor time series and tested with various data sets.

A software prototype has been implemented for the use case of tunnel construction based on time series sensor data.

Based on first experiments with real and simulated data, the solution was observed to provide a valuable contribution for ensuring data security and operational security. It enables detecting both anomalies caused by attacks (e.g., data tampering) and operational issues (e.g., sensor errors). The approach is assumed to be universally applicable and beneficial for many domains of interest.

Subsequent studies have investigated various AI approaches to support the data quality analysis of time series. One approach aims at predicting time series to use the prediction for later comparison with real data. For this purpose, pre-trained transformer models (TiREX, Lag-Llama, N-BEATS) have been evaluated and compared. The prediction results have been of similar quality to the ones obtained from classical, statistical prediction methods. The alternative of developing own AI models based on training data stemming from tunnelling projects remains an idea for future research.

In a second AI approach, available Large Language Models (LLMs), e.g., Claude Sonnet, have been fed with tunnel monitoring sensor time series and queried for the presence of anomalies and peculiar data patterns. The obtained answers have been of surprising correctness and proved a high level of domain knowledge and data understanding. Further research is required to evaluate and exploit the capabilities of LLMs for the detection of security-related data anomalies.

References

- [1] Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1), 43–49.
- [2] Auer, A., Podest, P., Klotz, D., Böck, S., Klambauer, G., & Hochreiter, S. (2025). TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning. <https://arxiv.org/abs/2505.23719>
- [3] Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., & Hochreiter, S. (2024). xLSTM: Extended Long Short-Term Memory. <https://arxiv.org/abs/2405.04517>
- [4] TiRex model: <https://huggingface.co/NX-AI/TiRex>
- [5] Rasul, K., Ashok, A., Williams, A. R., Ghonia, H., Bhagwatkar, R., Khorasani, A., Bayazi, M. J. D., Adamopoulos, G., Riachi, R., Hassen, N., Biloš, M., Garg, S., Schneider, A., Chapados, N., Drouin, A., Zantedeschi, V., Nevmyvaka, Y., & Rish, I. (2024). Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting. <https://arxiv.org/abs/2310.08278>
- [6] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. <https://arxiv.org/abs/2302.13971>
- [7] Oreshkin, B. N., Carпов, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. International Conference on Learning Representations. <https://openreview.net/forum?id=r1ecqn4YwB>
- [8] <https://github.com/ServiceNow/N-BEATS>