

Article

An Adaptive IoT-Based Forecasting Framework for Structural and Environmental Risk Detection in Tailings Dams

Raul Rabadán-Arroyo ¹, Ester Simó ², Francesc Aguiló-Gost ^{2,*}, Francisco Hernández-Ramírez ³
and Xavier Masip-Bruin ²

¹ Department Arquitectura de Computadors, Universitat Politècnica de Catalunya (UPC), 08034 Barcelona, Spain; raul@rabadan.dev

² CRAAX@UPC, Advanced Network Architectures Lab (CRAAX), UPC BarcelonaTech, 08800 Vilanova i la Geltrú, Spain; ester.simo@upc.edu (E.S.); xavier.masip@upc.edu (X.M.-B.)

³ Worldsensing, 08014 Barcelona, Spain; fherandez@worldsensing.com

* Correspondence: francesc.aguilo@upc.edu

Abstract

Tailings dams represent one of the most environmentally sensitive infrastructures in the mining industry. To address the need for continuous and accurate monitoring, this paper presents an adaptive forecasting framework that combines Internet of Things (IoT) technologies with machine learning (ML) models to detect early signs of structural and ecological risks. The proposed system architecture is modular and scalable and enables the automated training, selection, and deployment of predictive models for multivariate sensor data. Each sensor data flow is independently analyzed by using a configurable set of algorithms (including linear, convolutional, recurrent, and residual models). The framework is deployed via containers with a CI/CD pipeline and includes real-time visualization through Grafana dashboards. A use case involving tiltmeters and piezometers in an operational tailing dam shows the system's high predictive accuracy, with mean relative errors below 4% across all variables (in fact, many of them have a mean relative error below 1%). These results highlight the potential of the proposed solution to improve structural and environmental safety in mining operations.

Keywords: IoT; deep learning; risk detection

1. Introduction

The mining industry is a key engine of growth in many countries, since it is essential to obtain raw materials for the industrial, technological, and agricultural sectors, among others. However, it usually has a negative image mainly due to the environmental impact strongly linked to the mining operations, such as those created by the generated waste and acid mine drainage, which if not properly treated can contaminate water, soil, and air. The mining industry relies heavily on Tailings Storage Facilities (TSFs) for the long-term confinement of mineral waste. In this context, the tailings dam, the physical barrier that retains the waste, is a critical aspect because its failure can cause serious environmental and social damage [1]. To support continuous supervision, TSFs are increasingly instrumented with geotechnical sensors and integrated monitoring systems [2,3]. Indeed, IoT systems enable real-time data collection and analysis, which is crucial to optimize operations and ensure personnel safety and asset efficiency [4,5]. However, processing data from tailings dam sensors, including inclinometer and piezometer measurements, remains a major challenge. In operational settings, these sensor streams are often non-stationary, with



Academic Editors: Zijian Li, Zihan Chen, Jiahui Li and Longyu Zhou

Received: 3 December 2025

Revised: 28 January 2026

Accepted: 30 January 2026

Published: 3 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

outliers and missing values due to harsh environments and intermittent communication, which complicates reliable forecasting and automated supervision.

Given these challenges, machine learning and deep learning (ML/DL) methods have been explored for predictive modeling, forecasting, and anomaly detection to support dam monitoring and early-warning applications [6,7]. Related IoT research has also leveraged deep reinforcement learning to maintain timely data delivery under energy and bandwidth constraints [8]. However, to the best of our knowledge, end-to-end solutions that unify sensing and analytics while automating per-sensor model selection, supporting retraining, and ensuring robust containerized CI/CD deployment on noisy, non-stationary operational streams remain scarce.

To address these limitations, this paper proposes an end-to-end framework whose main contributions are

1. An adaptive IoT-based forecasting framework integrated into a modular monitoring architecture for TSFs.
2. An automated training and model-selection workflow that evaluates multiple time-series models (linear and deep learning families) and deploys the best-performing option for each sensor stream.
3. A container-based deployment approach (with CI/CD) that enables reliable operation and real-time visualization dashboards.
4. Validation on data from operational tailings dams using tiltmeters and piezometers, showing low forecasting errors (below 4%). Forecasts provide a baseline, and deviations between observed and predicted values can be used as early indicators to support risk-oriented supervision.

The rest of the paper is organized as follows: Section 2 reviews the state of the art. Section 3 introduces the proposed system architecture, and Section 4 describes the deployment strategy. Results are included in Section 5, and finally, conclusions are mentioned in Section 6.

2. Review of the State of the Art

In recent years, IoT systems have been utilized to improve the safety of facilities and personnel in many different mining environments. In this regard, Zhang et al. [5] provide an exhaustive review of recent advances in IoT applications in underground mines for environmental parameter monitoring. Other mining IoT deployments also address workforce safety and health monitoring [9,10]. In particular, the application of this type of technology in tailings dams represents a significant advance in monitoring. A relevant example may be found in [11], where the authors implement a safety monitoring system in a tailings dam environment to collect real-time information on key aspects, such as tailings dam deformation, seepage, water level, and rainfall, thus changing the traditional procedure for information collection and transmission. Han et al. [12] establish a real-time monitoring system based on IoT technology in a tailings dam at Okdong Mine to observe the structural deformation and leachate outflow characteristics over time.

Tailings dams generate continuous data from many different sensors (pore pressure, displacement, tilt, seepage, water level, rainfall, etc.). However, in practice, this data can contain noise, extreme values, gaps, irregular sampling, communication outages, and also sensor drift or changes due to maintenance [13]. All of this means that a trained model can lose accuracy when real-world conditions change [14]. Furthermore, these signals often follow regular cycles tied to reservoir level and temperature, but they can also shift quickly when conditions change. That can be triggered by external drivers like rainfall and temperature, or by internal changes in the system—especially evolving seepage conditions and how the structure responds over time [15]. Therefore, for analytics to work effectively in

operation, a pipeline with consistent preprocessing and proper time evaluation is typically required, as well as a mechanism to adapt/retrain the model when the data distribution changes [14,16].

Beyond monitoring, the challenge lies in exploiting the large amount of data generated by IoT systems to enable rapid detection and response to possible failures in critical infrastructures. Time series analysis is a fundamental predictive analytics tool that enables the identification of patterns, anomalies, and trends in physical variables evolving over time. In critical infrastructures, time-series analysis is often used to detect faults, plan maintenance, and improve operations, and it has worked well in many real cases [17,18]. However, traditional time-series methods often struggle when the system behavior is complex and non-linear.

To overcome these limitations, deep learning (DL) models are particularly effective at learning complex temporal dependencies from large time-series datasets and at detecting subtle patterns. An example of this can be found in Chriki et al. [19], where the authors use Long Short-Term Memory (LSTM) models to detect and predict anomalies for safety monitoring, validating the approach on a railway section. Park et al. [20] also propose an approach that combines an autoencoder and an LSTM model to detect and classify faults in an industrial process, reducing detection latency compared to traditional statistical approaches. In addition, Gasparin et al. [21] conduct a comprehensive benchmarking study of deep learning models for time series forecasting in the context of electric load prediction.

Recent works in geotechnical and dam/TSF monitoring show a trend toward ML/DL for forecasting key variables and detecting anomalies in continuously monitored instrumentation flows. Recent review papers in this field make the same point: monitoring data is often nonlinear, noisy, and nonstationary, which is exactly why data-driven methods have become so important for predicting system behavior and informing safety decisions [6,22]. In the forecasting field, recent studies of dams/TSFs report improved predictive performance with hybrid pipelines that combine signal decomposition and learning models. Recent literature offers concrete examples. On the one hand, supervised machine learning models have been compared to predict dam displacement when operating conditions change [23]. Along the same lines, approaches are also emerging that do not rely on a single value, but rather provide interval forecasts of TSF deformation, explicitly incorporating uncertainty [24]. Furthermore, digital twin methods are gaining traction, where real-time sensor data are combined with the finite element method (FEM) and transfer learning techniques to support near real-time assessments [25]. In addition to improving prediction models, monitoring systems are increasingly incorporating robust mechanisms to handle anomalous readings. For example, Ren et al. [26] combine detailed seepage and hydraulic loading prediction with explicit outlier detection (Isolation Forest and LOF) and add safety thresholds based on residues. In practice, this provides a clear way to distinguish between truly anomalous signals and the normal variability that appears in field data. Finally, TSF-oriented frameworks that combine ML with real-time monitoring architectures (including digital twin perspectives) further support the operational relevance of these approaches [27].

In contexts where data is generated in real time from multiple sensors, adaptive models that respond to changes in system conditions are essential. While the literature shows significant progress in IoT monitoring of TSFs and the use of ML/DL for prediction and anomaly detection in geotechnical time series, a practical problem arises in practice: each sensor behaves differently, and the data is rarely static. As time goes on, drift, maintenance-driven shifts, and fluctuations in signal quality can creep in, sometimes enough to make a model that once worked well break down [13,14,28]. That is why it is important to move beyond one-off training and toward smarter model selection

and adaptive learning approaches, such as AutoML and online learning, which can reduce manual re-engineering by automating model selection and model updates under drift [29,30]. In this paper, we introduce a modular framework that evaluates performance over time and automatically picks the best model for each sensor based on recent predictive accuracy. When it detects changes in the data, it can retrain and update the selected model, which ultimately makes real-time monitoring in tailings storage facilities more robust and reliable.

Our main objective in this work is to provide an easily implementable system for monitoring one or more tailings dams. Furthermore, we prioritize easy scalability, meaning that sensors can be added without compromising the predictions of those already in production. To this end, each sensor's data is predicted using a machine learning model with a related optimal error among a set of different models trained using the same sensor's historical data.

This scalability is possible thanks to the independence of the tasks associated with each individual sensor. The different processes of data reading, model training, and prediction depend solely on each sensor, allowing sensors with different data streams to be predicted using different models if necessary. In addition, the retraining of the different models associated with each sensor is also carried out independently and without affecting the predictions being generated simultaneously.

This independence makes it easy to add new sensors by modifying the system's configuration file. In this way, monitoring a new tailings dam is implemented by adding their sensors to this configuration file and deploying the corresponding monitoring dashboard to the control building of the new dam.

There is another noteworthy feature that can be used if needed. The different system modules can be independently deployed to different locations depending on local requirements. For example, the Trainer module, which requires computing power, can be dockerized in the cloud, while the Predictor module can operate closer to the edge, providing information to any dashboards implemented, whether on the same tailings dam or over different ones. The architecture of this component is detailed in the following section.

3. Architecture of the Forecasting Component

The following proposal is a multimodal adaptive forecasting system per sensor. There are several reasons for this election. On the one hand, having several models that have been trained with the same dataset ensures better results than having only one. On the other hand, as this system pretends to be installed in different tailings dams, as the behavior of the related datasets will eventually be different, one model will forecast better for some datasets while other model for others.

Although this proposal should yield better results, it has a weak point compared to the unimodal format, the retraining phases require more computation time. Although this task can be performed in the background while the system continues in production.

3.1. Overview

The functional architecture of the proposed system is shown in Figure 1. It consists of a set of highly connected components implemented in Python 3.11. Key components are the Trainer and the Predictor, while the other components are responsible for allowing external interaction and persistence. The Broker Client connects with an external message broker through MQTT. This client, when receiving updates on transactions, stores them into the database and requests the Predictor for forecasts (which can then be published back to the broker). It is able to do reads/writes from the database to obtain more metadata regarding the specifics on the device to generate predictions and retrieve training results (selected models, accuracies recorded, etc.). Grafana shows data in postgres, allowing to

monitor the behavior of the time series in a dashboard. The Trigger waits and requests the Trainer to start the process. The Trainer retrieves information of historical measures from the database, runs the training and selection process, and stores the results in the file system.

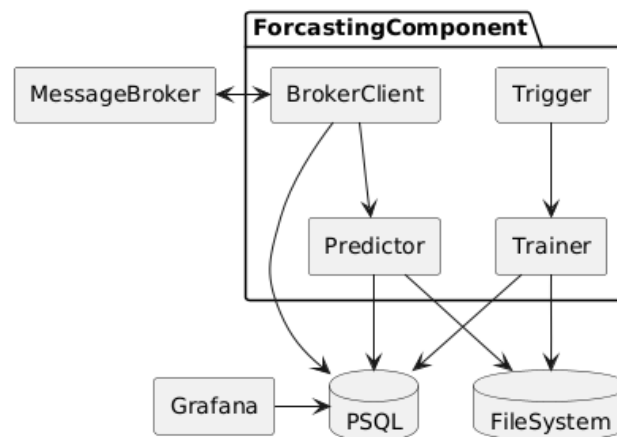


Figure 1. High-level block diagram of the forecasting component.

The Trainer module generates both models based on historical data for each device and a configuration of different models. The configuration can specify which algorithm to use and the hyperparameters' settings for training. Then, it selects the model showing the best performance according to a set of well-specified metrics (later described in Section 3.2). This training process is triggered by the Trigger at the start of the system. All models are retrained after a certain period of time. When historical datasets collected by sensors are small, that is, when the system begins to work from zero, the models tend to be overtrained. It is a transient state. When datasets turn out to be larger, containing enough data to do correct trainings, the models produce correct forecasting. It turns out to be a steady state. The Trainer module is independent of the Predictor, and can be easily adapted to run as a separate background process or distributed in different servers.

The Predictor module is triggered when a new measure is received. It loads a selected model for a given device, the previous N measures, and generates the M future measures (more details are explained in Section 3.3). As this module reacts to every measure received from a sensor, it can be adapted to run on separate servers if different instances of BrokerClient are able to consume different sets of messages.

The BrokerClient connects to a given MessageBroker and translates the messages received to store the obtained values to the local database. In the proposed implementation, we use a local Eclipse Mosquitto™ broker bridged to external endpoints.

3.2. Trainer

The purpose of the Trainer is to retrieve the available training configurations, apply them to the historical time series related to every device, and select the trained model performing the best for a given configuration. This loop is expressed in Algorithm 1, where the device information is retrieved from the database. The training configurations from the file system and the marking for usage are performed by creating a symbolic link in the file system.

Algorithm 1 Training loop

```

for all device ∈ GET_DEVICES_INFORMATION( ) do
  R ← ∅
  for all config ∈ GET_TRAINING_CONFIGS( ) do
    R ← R ∪ RUN_TRAINING_WITH(device, config)
  end for
  r ← SELECT_BEST_RESULT(R)
  MARK_FOR_USAGE_ON_PREDICTION(device, config)
end for

```

For every pair of a given device's time series summary and a training configuration loaded from disk, the Trainer applies a procedure similar to Algorithm 1. This procedure is defined in Algorithm 2. First, the configuration is checked to be compatible with the given device. Then, it follows the usual procedure where the data is loaded, split, preprocessed and used to train and evaluate a model. To allow a wide range of configurations while the system executes in the background, these steps are made dependent on the given configuration.

Algorithm 2 Training run for a device and configuration pair

```

function RUN_TRAINING_WITH(device, config) :
  if ARE_INCOMPATIBLE(device, config) then
    return ∅
  end if
  dataset ← RETRIEVE_TIMESERIES(device)
  t, v ← SPLIT(dataset, config)
  t, v ← PREPROCESS(t, v, config)
  m ← CREATE_TRAIN_MODEL(config, t)
  r ← EVALUATE_MODEL(config, v, m)
  PLOT_SAMPLES(config, t, v, m)
  return {r}
end function

```

The available preprocessing options are defined as follows:

- (a) The number of standard deviations from the mean required to consider the movement of a measure as an outlier. If this value is provided, the difference with the previous values is calculated to obtain the movements of each measure. Then, the mean of the movements of the measure is calculated, and in case there is a measure that deviates more than the specified number of standard deviations, it is dropped. If the value is not defined, this process is not performed.
- (b) Whether to work with differences or not. Activating this option makes each value be converted into the difference with the previous one. By doing so, the model works with the differences instead of the absolute values. If there is no previous measure, 0 is selected.
- (c) Whether to enrich the dataset with the sin and cos values for the day-relative time. This option was added because a daily cycle of sensor values was detected.
- (d) The size of the input and output time windows used during training.

The base models to train were based on [31], and summarized as follows:

1. *MultiStepLastBaseline*: Baseline model that repeats the last time step of the window
2. *RepeatBaseline*: Baseline model that repeats the input of the window as an output
3. *MultiLinearModel*: Model that makes a linear combination of the last time step of the window. It is implemented using a *Dense* layer without an activation function or hidden layer.

4. *DenseModel*: Model that takes the last time step of the window and passes it through a hidden dense layer with a ReLU activation function and configurable number of dense units.
5. *CNNModel*: Model that takes the last conv_width time steps of the window and passes it through a one-dimensional convolutional layer with a ReLU activation function and configurable number of units.
6. *RNNModel*: Model that takes the input window and passes it through an LSTM layer with a configurable number of units.
7. *ResidualMultiLinearModel*: A variation in the MultiLinearModel where the result is added to the extended last time step.
8. *ResidualDenseModel*: Variation in the DenseModel where the result is added to the extended last time step.
9. *ResidualCNNModel*: Variation in the CNNModel where the result is added to the extended last time step.
10. *ResidualRNNModel*: Variation in the RNNModel where the result is added to the extended last time step.

3.3. Predictor

The purpose of the *Predictor* each time it is triggered is (i) to determine whether a selected model exists for the given device; (ii) to retrieve previous measures to build an input window; (iii) to run the inference on the model; and (iv) to store and broadcast the result. This loop is expressed in Algorithm 3. Loading the model yields both the model and the training configuration used, stored in the file system. Moreover, the measures are loaded from the database. After the inference, the predictions are stored in the database and published to allow access by authenticated parties.

Algorithm 3 Predictor handling of a message

```

function HANDLE_TRIGGER(device, ts) :
  if NOT_TRAINED(device) then
    return
  end if
  model, config ← LOAD_SELECTED_MODEL(device)
  data ← LOAD_LAST_MEASURES(device, config)
  in_window ← PREPROCESS(data, config)
  out_window ← RUN_INFERENCE(model, data)
  tss ← GENERATE_TIMESTAMPS(ts)
  STORE_PREDICTIONS(device, tss, out_window)
  PUBLISH_PREDICTIONS(device, tss, out_window)
end function

```

4. A Tailing Dam's Monitoring Use Case

4.1. Scenario Description

This section includes a tailings dam monitoring scenario where the proposed system is to be implemented. Many different IoT devices and sensors have been deployed in order to properly measure key geotechnical parameters related to dam stability and environmental safety. Distinct environmental and operational conditions are considered to validate the system, thus ensuring its robustness, adaptability, and effectiveness under real-world conditions. The monitored variables include the following data: slope measured using inclinometers; pore pressure measured using piezometers; and absolute surface displacements measured using GNSS sensors. These parameters are pivotal to identify anomalies and/or structural changes that could drive possible dam instabilities. The collected data

enable the early detection of potential structural problems and serve as the basis to build predictive models aimed at both improving safety and informing maintenance strategies.

4.2. System Deployment Overview

The deployment includes four modular and scalable layers which are described in the following items:

- **Sensor Layer:** This layer includes GNSS nodes that are equipped with Real-Time Kinematic (RTK) positioning, tiltmeters, piezometers, and other geotechnical instruments.
- **Communication Layer:** It uses LoRa-based wireless communication connected to IoT nodes for long-range, low-power data transmission between IoT nodes and gateways.
- **Data Management Layer:** It consists of a centralized, cloud-compatible database that aggregates historical and real-time data, ensuring data integrity, scalability, and secure access.
- **Visualization and Analytics Layer:** This layer enables real-time monitoring as an interactive software platform. The forecasting module included in this work is integrated within it.

The installation and configuration of GNSS nodes at critical points across the TSFs are included in each deployment, also complemented by additional sensor types. The GNSS-Meter's millimetric precision enables the detection of multiple displacement phenomena that are essential for a comprehensive dam safety assessment as follows:

- Seasonal and thermal movements.
- Long-term creep detection.
- Early warning of anomalous movements.

GNSS technology also offers several fundamental advantages compared to conventional dam monitoring approaches as follows:

- **Absolute 3D positioning:** This capability is particularly valuable for identifying regional ground movements that may affect the dam foundation [32].
- Continuous automated operation.
- Cost-effective dense coverage.
- Integration with early warning systems.

Dam displacements are typically small, generally reaching up to 6 mm per year under normal operating conditions. Research across different dam types shows that embankment dams may experience vertical movements of approximately 6.0 mm per year at critical monitoring points [33].

The Worldsensing GNSS-Meter achieves sub-centimeter precision through advanced RTK technology and edge processing algorithms. The system provides three levels of data aggregation, with accuracy improving as the aggregation period increases. Its 24-h aggregated accuracy—2–4 mm horizontally and 3–7 mm vertically—demonstrates that the GNSS-Meter operates within the same order of magnitude as typical annual dam displacements. This resolution is not only sufficient but also well suited for several critical monitoring objectives.

It may seem paradoxical that GNSS errors (2–3 mm) can be comparable to annual dam displacements (1–6 mm). This is resolved by recognizing that dam monitoring is fundamentally a trend-detection problem, rather than a single-point absolute measurement problem. A single GNSS measurement with 2 mm uncertainty does not precisely define the dam's position. However, when hundreds or thousands of measurements are collected over weeks and months, statistical analysis can reveal displacement trends with significantly higher confidence.

For example, consider a dam with an annual displacement of 3 mm/year. Over a 30-day period, the expected displacement is approximately 0.25 mm, which appears impos-

sible to detect with 2 mm measurement accuracy. However, with 24 hourly measurements per day (around 720 measurements per month), the statistical strength of the dataset enables the detection of sub-millimetric trends. Techniques such as the remove–restore approach, which models deterministic components (thermal and hydrostatic effects) and applies collocation for stochastic analysis, can achieve RMSE errors well below 1 mm when processing long time series. Larger displacements can be detected more easily and can trigger alarms immediately.

In practice, dam monitoring relies on multiple data sources (e.g., tiltmeters, piezometers, total stations). GNSS provides an additional and highly valuable layer of information that, before the adoption of IoT-based solutions, was often expensive and labor-intensive to deploy and maintain.

The effectiveness of a GNSS-based dam monitoring system depends critically on the strategic placement of sensors to capture the three-dimensional deformation behavior of the structure. The sensor network must provide sufficient spatial coverage to detect localized anomalies, while also maintaining redundancy for reliable data validation. The placement strategy is guided by the following engineering principles derived from dam structural behavior and GNSS measurement requirements:

- Structural critical points.
- Spatial deformation patterns.
- RTK base–rover geometry.
- Redundancy and validation.

The exact locations of monitoring systems on dams are typically highly restricted by both operators and national regulations, as dams are considered critical infrastructure. Therefore, we aren't able to include a graphical location of sensors by confidentiality reasons.

In the domain of geotechnical and structural monitoring, the precision, reliability, and longevity of the deployed sensors are paramount. In this use case Worldsensing's Tiltmeters are considered as they offer a combination of superior specifications, advanced operational modes, and a focus on long-term, autonomous operation. They boast impressive technical specifications, including a resolution of $\pm 0.0001^\circ$ and a repeatability of $< 0.0003^\circ$ for its high-precision model (Tilt-XHP). Moreover, it can use a lower power consumption, enabling an exceptional battery life of up to several years [34]. Because of the strategically selected location of the base stations, uninterrupted communication with all devices and gateways is granted. Once the data acquisition process is operational, the system is able to access historical datasets. This fact ensures access to a context for baseline analysis and predictive model training.

The MQTT protocol is used to transmit real-time sensor data that is processed by dockerized analytics modules. This choice of communication protocol is a critical decision that directly impacts the efficiency, reliability, and scalability of a monitoring system. While other protocols' architectures are not inherently optimized for the unique constraints of IoT environments, the Message Queuing Telemetry Transport (MQTT) protocol has emerged as the de facto standard for IoT data transmission, particularly in the context of transferring data from an IoT gateway to a cloud server. The advantages of MQTT over alternatives like HTTP are numerous and significant, centering on efficiency, reliability, and a flexible architecture. MQTT's design as a lightweight, publish-subscribe protocol makes it exceptionally well-suited for resource-constrained devices and networks with low bandwidth and high latency. Unlike HTTP's request-response model, where a client sends a request and waits for a server's response, MQTT's pub-sub model decouples the data producer (the IoT device or gateway) from the data consumer (the cloud application). This asynchronous communication, combined with a minimal message overhead (as low as 2 bytes compared to HTTP's 8-byte minimum), results in significantly more efficient use of

network resources. As demonstrated in performance benchmarks, the difference in data consumption becomes increasingly stark as the number of messages grows, making MQTT a far more scalable solution for long-term monitoring projects [35,36].

This data is also made visible through user-friendly dashboards adapted to the needs of dam engineers and safety officers. A standalone Python-based application implements these dashboards, with the only requirement of an internet connection for real-time updates.

4.3. Forecasting Framework Integration

The deployment proposed to test the forecasting module has been performed using containers and a CI/CD pipeline. Moreover, to ease the verification of the models and the system functionality, we also added a local Grafana dashboard.

The containerization of the Python code has been performed using Docker and the base `python:3.11-slim` image. Besides the deployment, it has also been used to run the tests and as a development container. Additionally, we used Docker Compose to control versions and also to deploy the local database, the local message broker, and the local Grafana services together with the Python code.

The repository used for the source code was contained in the GitLab instance hosted by UPC. For each commit, we ran the CI/CD pipeline in a private runner on a server located at the CRAAX Lab in UPC. It consisted of first running unit tests, a linter, type checking, and scanning for common security errors. Once passed, the system container is built and deployed to the test server. We made sure to do it in a highly available way to avoid missing message updates.

The Grafana configuration is relatively bare-bones, as it was only used to be able to have an easy way to verify that the system was working, what the data was like, and the behavior of the models. It only contains one dashboard with the database as its data source. The dashboard allows selecting the device and which of the variables of its respective time series to display. The data is expressed in two panels. The first contains the values of the selection and all predicted values contained in the database. As we generate a window every time a measure is received and the older ones overridden, the historic prediction trend only contains the next expected value. The second panel contains the mean relative error between the prediction of the next value and the real value of the last 24 measures.

5. Results

Although this section includes results related to two types of sensors, tiltmeters and piezometers, any other type can be added. The system is not limited in this respect. In this use case, only these two types were tested, and many sensors were strategically placed. By doing so, we avoid data reception failures due to redundancy in the event of unforeseen circumstances such as battery drain, accidental failure of a sensor, etc. It doesn't matter how many sensors are placed; our component accepts all of them. Every added new sensor needs some time for its associated models to predict results reasonably (when its related dataset is large enough, letting the model not be overtrained).

A tiltmeter measures three types of angle of inclination of a pole; these angles are denoted by x , y and z . A piezometer measures the pore pressure in the ground and frequency values. It is worth mentioning that, as stated in the literature, relative errors below 10% are considered very good predicted values; from 10% to 20%, good ones; from 20% to 30%, acceptable; from 35% to 50%, regular; and over 50%, unacceptable [37,38].

The training task has been performed using the loss function MAE (Mean Absolute Error); however, the MSE (Mean Squared Error) has also been included for metrical consid-

erations. The percentage volume of data has been taken as 20% for training and 80% for validating. These two metrics have the expression:

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

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

where y_i is a real value and $\hat{y}_i$ is the related predicted value. For practical reasons, we were asked to plot the curve of MRE (Mean Relative Error) which makes sense when no real data attains the value zero. The expression of MRE is

$$\text{MRE} = \frac{1}{N} \sum_{i=1}^N \frac{|y_i - \hat{y}_i|}{|y_i|}. \quad (3)$$

For the tiltmeter data stream, the most suitable model appears to be the RNN and its variants RNN_Diff and RNN_Enrich. These models can be trained using different configurations of their hyperparameter values. We will symbolically denote these differences as 8u, 16u, 32u, 64u, 256u or 1024u. The other model types either obtain worse values for the chosen error metrics or tend to overtrain, and we filter them out directly.

Table 1 is included here showing only the models with the best error metrics values obtained in one training process of the angle x . This table contains the symbolic information of hyperparameters (first column), two versions of the RNN model (the plain model and the enriched one), and their respective measures of error metrics (MAE and MSE). The first row of each hyperparameter u includes the metric values of training, and the second row the validation metrics.

Table 1. Some numerical figures related to one tiltmeter training process.

Hyp	RNN		RNN_Enrich	
u	MAE	MSE	MAE	MSE
8			0.1598	0.0718
			0.1720	0.0727
16	0.1714	0.0740	0.1726	0.0729
	0.1902	0.0764	0.1963	0.0749
32			0.1801	0.0784
			0.2102	0.0843
64			0.1957	0.0842
			0.2210	0.0851
1024	0.1669	0.0682		
	0.2042	0.0784		

One of the training processes is commented on here. Their best error metric values are contained in Table 1. The best of them was attained by the RNN_Enrich-8u model, with a related MAE of 0.1598 and a related MSE of 0.0718. Notice that many more models have been trained, but they have not been considered because either they are overtrained or they have worse MAE results.

Each predicted value is shown in two subfigures. The first subfigure includes the real values in green and the predicted values in yellow. The second subfigure shows a red graph that represents the mean of the last 24 relative errors between the next real value and his corresponding prediction.

The figures included in this work contain about 200 predictions for each parameter analyzed. Figures 2–4 show the predictions of a tiltmeter angles values x , y and z . You can notice how their behavior is different from the one of Figure 5, related to the pressure values predictions of a piezometer. The values from the first sensor zigzag, while those from the second one have a different behavior, as you can see in Figure 5.

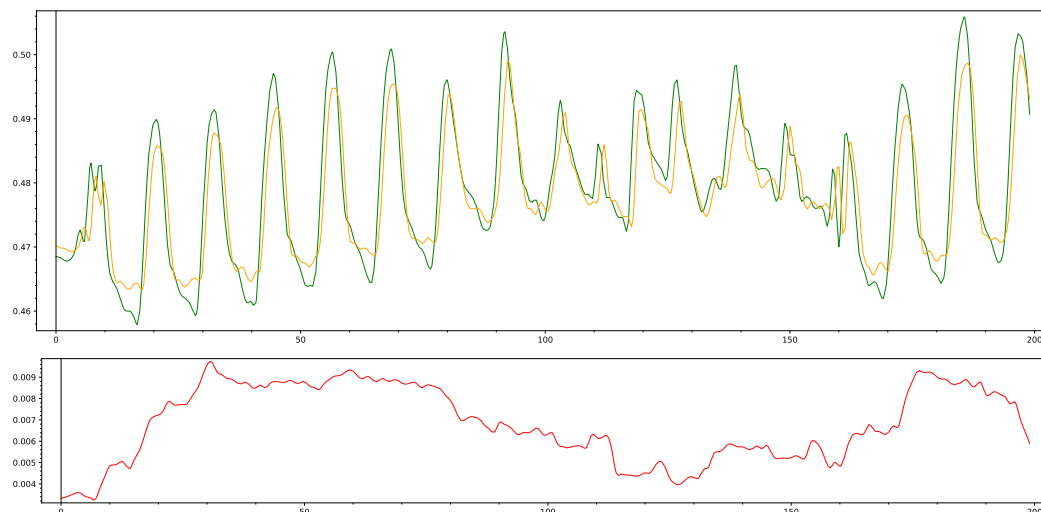


Figure 2. Tiltmeter's x -angle forecasting. Real values in green, predicted values in yellow and error values in red.

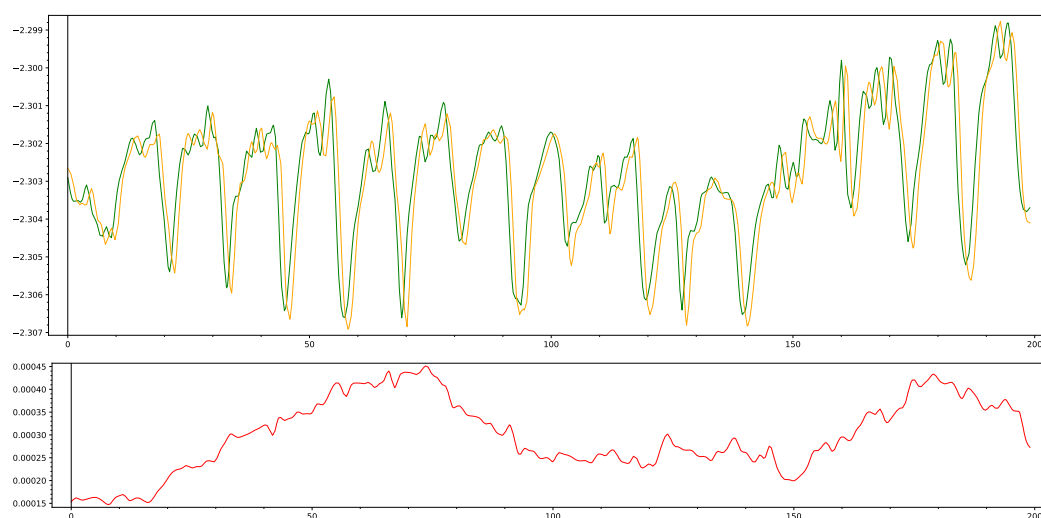


Figure 3. Tiltmeter's y -angle forecasting. Real values in green, predicted values in yellow and error values in red.

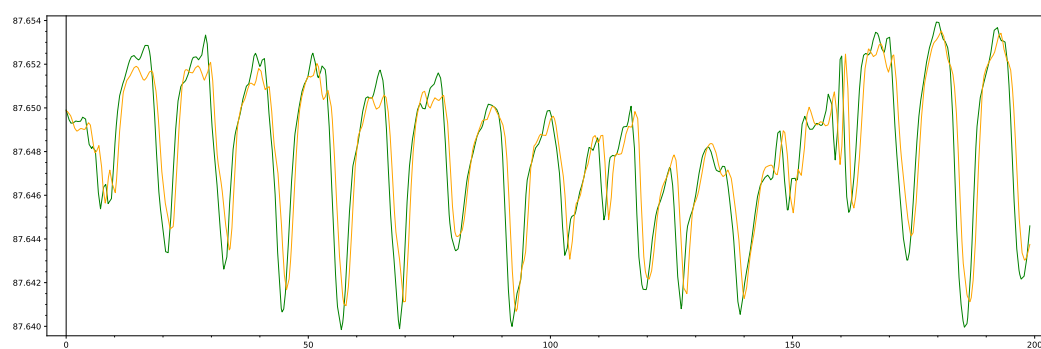


Figure 4. Cont.

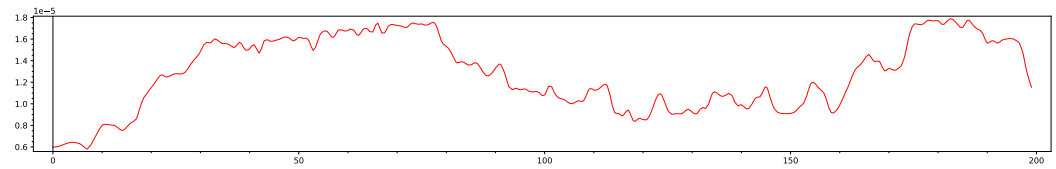


Figure 4. Tiltmeter's z-angle forecasting. Real values in green, predicted values in yellow and error values in red.

We can see that the relative error curve remains below 1% in all cases. This is not always the case; in some tilt meters, we have obtained error values greater than 1%, but always below 10% (few cases exceed 5%, and all do not reach 10%). These relative errors are more than acceptable.

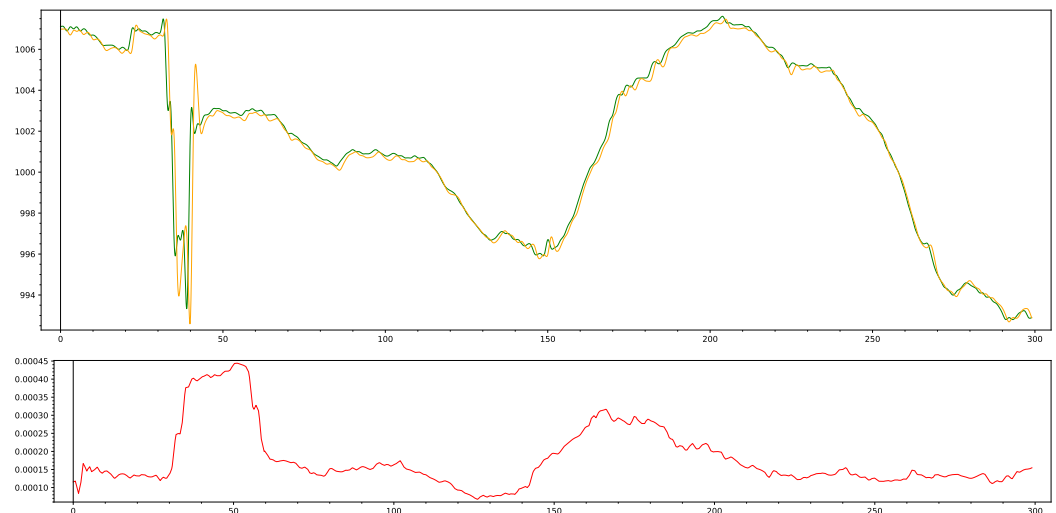


Figure 5. Piezometer's pressure forecasting. Real values in green, predicted values in yellow and error values in red.

The piezometer forecasting is performed by model `Residual_CNN-8u` the entire time period plotted in Figure 5 with MAE 0.1663 and MSE 0.0710.

6. Conclusions

The previous section shows how the multimodel forecasting framework proposed in this paper gives predicted values from good to excellent. The main advantages of this environment can be summarized in the following items:

- The system is flexible and modular enough to automatize the selection of the best model among several options for each sensor independently. Thus, there are more chances of obtaining successful results even when different behaviors of data sensors are present.
- Retraining all models from time to time makes the system more robust with respect to eventual behavior changes in the collected sensors' data.
- In general, the mean of relative errors in the predicted values is considered good enough when it is below 10%. Here, the largest mean of the relative errors is below 4%, corresponding to the x -angle (Figure 2), and all other predictions have a mean of relative errors well below 1%.
- The proposed prediction component can be dockerized and ported to the main computer of the tailings dam station. Once installed, models related to all configured sensors begin to learn themselves. After some time of collecting data measures and ensuring the errors turn out to be small, the system can be put into production.

- Another different working mode is the centralization of this service, that is, when the mine company has several tailings dams placed in different locations, far apart from each other. In this case, the component can be installed in a server where all different tailings dams send their collected data. The system accepts this kind of configuration as simple sensor additions.
- Environmental changes will be detected whenever the sensors can measure it. Depending on the magnitude of the change, the system will treat it in two different ways. As all models (related to all sensors) are retrained from time to time, little environmental changes will be learned by the models. Of course, a great change in a short time period will be raised as an anomaly. Other more radical changes, such as earthquakes or accidents that directly affect the tailings dam, such as a heavy vehicle collision that damages its structure, will not be detectable.
- This item isn't an advantage, although it is not very bad. The computation time of the retraining process is larger than if the system were unimodal. However, as the time between consecutive retraining can be configured, it is not an issue. Moreover, while retraining, the system continues forecasting (using the current models).

These characteristics show key features of the proposed framework, namely flexibility, scalability, and modularity, making its adoption by other scenarios a relatively easy process, only requiring minor changes in the current development.

7. Limitations and Future Improvements

Although this proposal offers a practical and quick way (through dockerization) to monitor a tailings dam, there are some limitations that suggest some future work that will add robustness to the system.

Although the next sensor's value is reasonably well predicted, it needs a measure of the maximum window prediction output with a reasonable error. Presumably, a high computational time cost will be required. The application of this proposal to public datasets should be performed, and the comparison with other systems' responses could allow the improvement of it.

Tracking systems like this one rely on statistical properties (deviation from the median compared to the variance, etc.) to detect potential anomalies. The system must raise an alert if a heavy vehicle accidentally approaches a sensor, or if any other similar circumstance occurs. A specialist then has to decide about a potential weakness risk of the dam and what action to take. This process costs time and money. A worthwhile future task is to distinguish these outliers from true anomalies by detecting certain patterns that occur in the former but not the latter.

Unexpected sensor events like depleted batteries, accidental breakage, communication failure, etc., are detected now by a lack of historical data in the specific timestamps. However, a more complex and detailed analysis of this topic is necessary.

Author Contributions: Conceptualization, all authors; methodology, all authors; software, R.R.-A.; validation, all authors; formal analysis, R.R.-A., E.S. and F.A.-G.; investigation, R.R.-A., E.S. and F.A.-G.; resources, Hernández; data curation, R.R.-A. and E.S.; writing—original draft preparation, R.R.-A., E.S. and F.A.-G.; writing—review and editing, all authors; visualization, all authors; supervision, X.M.-B.; project administration, X.M.-B.; funding acquisition, X.M.-B. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by the SEC4TD project, funded by EIT with Project Agreement Number 21123, and for UPC authors by the Spanish Ministry of Science and Innovation under grant PID2024-156150OB-I00, as well as by the Catalan Government under contract 2021 641 SGR 00326.

Data Availability Statement: Data sets used in this work are private and confidential.

Conflicts of Interest: Author F. Hernández-Ramírez was employed by the company Worldsensing. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Garcia, F.F.; Camilo-Cotrim, C.F.; Caramori, S.S.; Bailão, E.F.L.C.; Nabout, J.C.; Gitirana Junior, G.d.F.N.; Almeida, L.M. Mine Tailings Dams' Failures: Serious Environmental Impacts, Remote Solutions. *Environ. Dev. Sustain.* **2025**, *27*, 18179–18201. [\[CrossRef\]](#)
2. Cacciuttolo, C.; Guzmán, V.; Catriñir, P.; Atencio, E. Sensor Technologies for Safety Monitoring in Mine Tailings Storage Facilities: Solutions in the Industry 4.0 Era. *Minerals* **2024**, *14*, 446. [\[CrossRef\]](#)
3. Zvarivadza, T.; Onifade, M.; Dayo-Olupona, O.; Said, K.O.; Githiria, J.M.; Genc, B.; Celik, T. On the Impact of Industrial Internet of Things (IIoT)—Mining Sector Perspectives. *Int. J. Min. Reclam. Environ.* **2024**, *38*, 771–809. [\[CrossRef\]](#)
4. Ersan, M.; Irmak, E. Development and Integration of a Digital Twin Model for a Real Hydroelectric Power Plant. *Sensors* **2024**, *24*, 4174. [\[CrossRef\]](#)
5. Zhang, H.; Li, B.; Karimi, M.; Saydam, S.; Hassan, M. Recent Advancements in IoT Implementation for Environmental, Safety, and Production Monitoring in Underground Mines. *IEEE Internet Things J.* **2023**, *10*, 14507–14526. [\[CrossRef\]](#)
6. Mojtahedi, F.F.; Yousefpour, N.; Chow, S.H.; Cassidy, M. Deep Learning for Time Series Forecasting: Review and Applications in Geotechnics and Geosciences. *Arch. Comput. Methods Eng.* **2025**, *32*, 3415–3445. [\[CrossRef\]](#)
7. Hariri-Ardebili, M.A.; Mahdavi, G.; Nuss, L.K.; Lall, U. The Role of Artificial Intelligence and Digital Technologies in Dam Engineering: Narrative Review and Outlook. *Eng. Appl. Artif. Intell.* **2023**, *126*, 106813. [\[CrossRef\]](#)
8. Zheng, K.; Luo, R.; Liu, X.; Qiu, J.; Liu, J. Distributed DDPG-Based Resource Allocation for Age of Information Minimization in Mobile Wireless-Powered Internet of Things. *IEEE Internet Things J.* **2024**, *11*, 29102–29115. [\[CrossRef\]](#)
9. Vlachos, M.; Pavlopoulos, L.; Georgakopoulos, A.; Tsimiklis, G.; Amditis, A. A Robust End-to-End IoT System for Supporting Workers in Mining Industries. *Sensors* **2024**, *24*, 3317. [\[CrossRef\]](#)
10. Jiang, Y.; Chen, W.; Zhang, X.; Zhang, X.; Yang, G. Real-Time Monitoring of Underground Miners' Status Based on Mine IoT System. *Sensors* **2024**, *24*, 739. [\[CrossRef\]](#)
11. Wang, L.; Yang, X.; He, M. Research on Safety Monitoring System of Tailings Dam Based on Internet of Things. *IOP Conf. Ser. Mater. Sci. Eng.* **2018**, *322*, 052007. [\[CrossRef\]](#)
12. Han, S.H.; Jung, M.C.; Kim, J.W.; Jeon, S.W.; Yoon, K.W.; Hwang, I.H.; Park, J.H.; Koh, K.R. Evaluation of Applicability of Monitoring System Using IoT in a Tailings Dam. *J. Korean Soc. Miner. Energy Resour. Eng.* **2019**, *56*, 645–653. (In Korean with English abstract) [\[CrossRef\]](#)
13. Goknil, A.; Nguyen, P.; Sen, S.; Politaki, D.; Niavis, H.; Pedersen, K.J.; Suyuthi, A.; Anand, A.; Ziegenbein, A. A Systematic Review of Data Quality in CPS and IoT for Industry 4.0. *ACM Comput. Surv.* **2023**, *55*, 327. [\[CrossRef\]](#)
14. Hinder, F.; Vaquet, V.; Hammer, B. One or Two Things We Know about Concept Drift—A Survey on Monitoring in Evolving Environments. Part A: Detecting Concept Drift. *Front. Artif. Intell.* **2024**, *7*, 1330257. [\[CrossRef\]](#)
15. Deng, Z.; Gao, Q.; Huang, M.; Wan, N.; Zhang, J.; He, Z. From Data Processing to Behavior Monitoring: A Comprehensive Overview of Dam Health Monitoring Technology. *Structures* **2025**, *71*, 108094. [\[CrossRef\]](#)
16. Dong, K.; Yang, D.; Yan, J.; Sheng, J.; Mi, Z.; Lu, X.; Peng, X. Anomaly Identification of Monitoring Data and Safety Evaluation Method of Tailings Dam. *Front. Earth Sci.* **2022**, *10*, 1016458. [\[CrossRef\]](#)
17. Cook, A.A.; Mısırlı, G.; Fan, Z. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet Things J.* **2020**, *7*, 6481–6494. [\[CrossRef\]](#)
18. Abbasi, A.R.; Mahmoudi, M.R.; Arefi, M.M. Transformer Winding Faults Detection Based on Time Series Analysis. *IEEE Trans. Instrum. Meas.* **2021**, *70*, Art. no. 3516210. [\[CrossRef\]](#)
19. Chriki, F.Z.; Simó, E.; Aguiló, F.; Garcia-Milà, I.; Masip, X. A Deep Learning Framework for Safety Monitoring of a Railway Section. In *Proceedings of the 2024 IEEE International Conference on Cyber Security and Resilience (CSR), London, UK, 2–4 September 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–6. [\[CrossRef\]](#)
20. Park, P.; Di Marco, P.; Shin, H.; Bang, J. Fault Detection and Diagnosis Using Combined Autoencoder and Long Short-Term Memory Network. *Sensors* **2019**, *19*, 4612. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Gasparin, A.; Lukovic, S.; Alippi, C. Deep Learning for Time Series Forecasting: The Electric Load Case. *CAAI Trans. Intell. Technol.* **2022**, *7*, 1–25. [\[CrossRef\]](#)
22. Najih, H.; Aboulhassane, A.; Moustachi, O.E.K. Contributions of Artificial Intelligence to Dam Monitoring: A Literature Review. *Asian J. Civ. Eng.* **2025**, *26*, 4925–4940. [\[CrossRef\]](#)

23. Babadi, A.M.; Mirzabozorg, H.; Baharan, K. Predictive Modeling of Concrete Arch Dam Behavior: Evaluating the Efficacy of Random Forest and Radial Basis Function Networks. *AI Civ. Eng.* **2025**, *4*, 22. [\[CrossRef\]](#)
24. Hao, T.; Xu, K.; Zheng, X.; Li, J.; Wang, H. A Tailings Dam Displacement Interval Prediction Model Based on Time Series Decomposition and an Improved Elman Neural Network Model. *Stoch. Environ. Res. Risk Assess.* **2025**, *39*, 5959–5976. [\[CrossRef\]](#)
25. Tian, J.; Chen, C.; Zhang, L.; Chen, J.; Huang, H.; Zhang, P. A Digital Twin Method for Real-Time Analysis of Structural Deformation and Failure for High Arch Dams. *Eng. Struct.* **2025**, *345*, 121426. [\[CrossRef\]](#)
26. Ren, J.; Nan, S.; Zhang, J.; Zhang, S. A Deep Learning Approach Driven by Raw Monitoring Data for Earth/Rockfill Dam Seepage Prediction and Safety Assessment. *J. Civ. Struct. Health Monit.* **2025**, *15*, 2017–2036. [\[CrossRef\]](#)
27. Mwanza, J.; Mashumba, P.; Telukdarie, A. A Framework for Monitoring Stability of Tailings Dams in Realtime Using Digital Twin Simulation and Machine Learning. *Procedia Comput. Sci.* **2024**, *232*, 2279–2288. [\[CrossRef\]](#)
28. Alsharef, A.; Sonia; Kumar, K.; Iwendi, C. Time Series Data Modeling Using Advanced Machine Learning and AutoML. *Sustainability* **2022**, *14*, 15292. [\[CrossRef\]](#)
29. Kalaitzidis, E.; Diamantopoulos, T.; Michailoudis, A.; Symeonidis, A.L. AML4S: An AutoML Pipeline for Data Streams. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 87. [\[CrossRef\]](#)
30. Kovalevsky, V.E.; Zhukova, N.A. Building a Model for Time Series Forecasting Using AutoML Methods. In *Proceedings of the 2024 XXVII International Conference on Soft Computing and Measurements (SCM), Saint Petersburg, Russia, 22–24 May 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 308–311. [\[CrossRef\]](#)
31. TensorFlow. Time Series Forecasting. Available online: https://www.tensorflow.org/tutorials/structured_data/time_series (accessed on 18 December 2024).
32. Reguzzoni, M.; Rossi, L.; De Gaetani, C.I.; Caldera, S.; Barzaghi, R. GNSS-Based Dam Monitoring: The Application of a Statistical Approach for Time Series Analysis to a Case Study. *Appl. Sci.* **2022**, *12*, 9981. [\[CrossRef\]](#)
33. Yépez-Rincón, F.D.; Ferriño Fierro, A.L.; Escobedo Tamez, A.N.; Guerra Cobián, V.H.; Pinedo Sandoval, O.E.; Chávez Gómez, J.H.; Alatorre Cejudo, L.C.; Pirasteh, S. Mapping Longitudinal and Transverse Displacements of a Dam Crest Based on the Synergy of High-Precision Remote Sensing. *Adv. Civ. Eng.* **2024**, *2024*, 6220245. [\[CrossRef\]](#)
34. Worldsensing. Loadsensing G7 Tiltmeter. Available online: <https://www.worldsensing.com/discover-loadsensingg7/> (accessed on 9 January 2026).
35. Callen, J.M. Worldsensing CMT Edge MQTT Connection. 2024. Available online: https://bentleysystems.service-now.com/community?id=kb_article_view&sysparm_article=KB0108462 (accessed on 9 January 2026).
36. Craggs, I. MQTT vs. HTTP for IoT. 2024. Available online: <https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iiot/> (accessed on 9 January 2026).
37. Lewis, C. *Industrial and Business Forecasting Methods: A Practical Guide to Exponential Smoothing and Curve Fitting*; Butterworth Scientific: London, UK, 1982.
38. Montaña-Moreno, J.; Palmer-Pol, A.; Sesé-Abad, A.; Cajal-Blasco, B. Using the R-MAPE Index as a Resistant Measure of Forecast Accuracy. *Psicothema* **2013**, *25*, 500–506. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.