ESCAP/WMO
Typhoon Committee

TECHNICAL REPORT ON HYDROLOGICAL DATA QUALITY CONTROL AND FLOOD FORECASTING USING AI TECHNIQUES

DECEMBER 2025

UNITED NATIONS
ESCAP
Economic and Social Commission for Asia and the Pacific

WMO

TC/TD-No. 0024

Image source: Front cover image was generated using ChatGPT (OpenAI) based on author-designed prompts.

HYDROLOGICAL DATA QUALITY CONTROL AND FLOOD FORECASTING USING AI TECHNIQUES

Authors

Dr. Chung-Soo Kim, Ms. Cho-Rong Kim, Dr. Kah-Hoong Kok (Korea Institute of Civil Engineering and Building Technology, Republic of Korea);
Ms. Hye-Jin Park, Ms. Myung-Suk Hyun, Mr. Yoo-Han Cho (Han River Flood Control Office, Republic of Korea)

Special appreciation is extended to the four target Typhoon Committee member countries (Department of Meteorology & Hydrology of Lao P.D.R., Department of Irrigation & Drainage of Malaysia, Philippine Atmospheric, Geophysical and Astronomical Services Administration of Philippines, and Royal Irrigation Department of Thailand) and to the Typhoon Committee Secretariat (TCS), especially Dr. Jinping Liu, for their strong support and collaboration.

NOTE

TC/TD-No. 0024

ISBN 978-99981-833-4-6

ESCAP/WMO Typhoon Committee Secretariat
Avenida 5 de Outubro, Coloane
Macao, China
Tel.:(+853) 88010531 Fax: (+853) 88010530
E-mail: info@typhooncommittee.org

Published in February 2026

Digital printing, Macao, China in February, 2026

TABLE OF CONTENTS

FOREWORD

Hydrological data is fundamental for understanding and managing water resources, predicting extreme weather events, and supporting sustainable development. However, ensuring the quality of hydrological data is challenging due to the frequent presence of errors caused by sensor malfunctions, communication failures, human errors, and extreme natural events. Such anomalies in hydrological datasets can lead to inaccurate analyses, unreliable models, and flawed decision-making in areas such as flood forecasting, drought management, and water resources planning.

Flood forecasting is a critical component of disaster management, aimed at minimizing the loss of life, property, and infrastructure caused by flooding. Conventional forecasting methods rely on hydrological and meteorological models, which are built on physics-based equations and statistical analyses. While these models have provided valuable insights, they often struggle with capturing the complex and non-linear relationships between various hydrological and meteorological variables, especially under changing climate and land-use conditions.

The emergence of Artificial Intelligence (AI) provides a transformative approach not only to hydrological data quality control, but also to flood forecasting, enabling the automation of anomaly detection and correction with unprecedented accuracy and efficiency. AI-driven flood forecasting also offer several advantages, including improved accuracy, faster computation, adaptability to real-time data, and reduced dependency to domain specific assumptions.

This technical report presents the concept of AI techniques adopted for development of the hydrological data quality control and flood forecasting systems in the target TC member countries (Laos, Malaysia, Philippines, Thailand). An unsupervised machine learning algorithm, Isolation Forest, and a supervised machine learning algorithm, XGBoost, are employed to identify abnormal values in rainfall and water level time-series data. In addition, a deep learning–based Long Short-Term Memory (LSTM) auto-encoder is introduced to correct the identified outliers in water level time series, while an XGBoost regressor is applied to rectify outliers in rainfall time series. Furthermore, a conventional LSTM framework is used to predict future water levels or streamflow in rivers, supporting early flood warning applications.

Two PC-based desktop programs, the hydrological data quality control system and the flood forecasting system based on the discussed AI techniques have been successfully developed. The functions and implementation procedures for each system module are clearly elaborated in this technical report. The systems are currently undergoing final revision and tuning prior to distribution to the target TC member countries for practical real-world applications.

Here, I would like to express my sincerest appreciation to all experts of Han River Flood Control Office, Republic of Korea (HRFCO) and the the Korea Institute of Civil Engineering and Building Technology (KICT) of Republic of Korea, as well as to all related personnel and representatives of the TC Members, for their kind cooperation and valuable contribution to the successful completion of this technical report. I would also like to take this opportunity to thank the

staff of the ESCAP/WMO Typhoon Committee Secretariat (TCS) for their great efforts in coordinating the implementation.

Dr. DUAN Yihong
ESCAP/WMO Typhoon Committee
Secretary
December 2025

1. HYDROLOGICAL DATA QUALITY CONTROL USING AI

The integrity and reliability of hydrological data, particularly rainfall and water level records, are essential for effective water resource management, flood forecasting, ecological assessment, and hydrological modeling. Traditional quality control (QC) procedures have long provided standardized methods to ensure data validity, as outlined in references such as the Hydrology Manual and the World Meteorological Organization's (WMO) Quality Management Framework (QMF), which emphasize systematic validation and operational efficiency. However, the growing volume of telemetered data from precipitation and water level stations has highlighted the need for automation. Recent advancements in artificial intelligence (AI) have enhanced data quality control by overcoming the limitations of conventional methods and improving detection of errors and outliers. Automated quality control (AQC) systems now enable consistent validation across large datasets, minimizing manual intervention and strengthening the overall reliability and usability of hydrological databases for scientific and operational applications.

The detection of outliers in hydrological time series data such as rainfall and water level data is a critical aspect of water resource management, environmental monitoring, and disaster prevention. Recent advancements in machine learning (ML) techniques have significantly contributed to this domain, enabling more accurate and efficient identification of abnormal patterns. Therefore, the unsupervised and supervised machine learning algorithms have been widely applied for outlier detection in rainfall and water level datasets.

Unsupervised learning methods have been prominently utilized for pattern recognition and outlier detection without relying on labeled data. Auto-encoder, a form of deep learning unsupervised models, have been employed for water level outlier detection with notable success. On the other hand, supervised learning approaches require labeled datasets to train models to recognize outliers. Both supervised and unsupervised ML techniques have demonstrated significant potential in identifying abnormal rainfall and water level data. Unsupervised methods excel in scenarios with limited labeled data and are effective in discovering hidden patterns and outliers. Supervised approaches, particularly deep learning models like Auto-encoder, provide high accuracy in classifying outliers when labeled datasets are available. The choice between supervised and unsupervised methods depends on data availability, the complexity of the environment, and specific application requirements.

From traditional regression and classification models to advanced deep learning architectures and hybrid frameworks, ML techniques have demonstrated significant potential in improving outlier detection accuracy, enabling early warning systems by predicting river water levels and issuing early flood warnings several hours or days in advance using ML models trained on historical hydro-meteorological data, and supporting sustainable water resource management. The application of ML in quality controls of hydrological data has further enhanced the capacity for real-time monitoring and outlier detection, paving the way for more resilient hydrological systems in the face of climate variability and complicating water-related disasters and crisis.

It has been proposed to adopt unsupervised Isolation Forest and a supervised XGBoost ML algorithms to establish a framework for detecting outliers in hydrological datasets collected within the Typhoon Committee (TC) member countries. Here, outliers refer to

observations in hydrological records that exhibit significant deviations from the expected range or the underlying statistical behavior of the dataset, potentially indicating abnormal measurements or rare hydrological events. This initiative aims to enhance the capacity in implementing integrated hydrological data quality control and processing. Furthermore, this initiative is designed to assist local authority in shifting from manual, human-dependent processes to AI-enabled and data-driven hydrological data management. It is believed that this initiative can help address the shortage of experienced manpower in target TC member countries by enabling a more holistic and automated approach to hydrological data management.

1.1 Outlier Detection Using Unsupervised Machine Learning Algorithm – Isolation Forest

The main advantage of using an unsupervised machine learning algorithm for outlier detection is it does not require labeled datasets to train a model to tell a hydrological data point whether is normal or abnormal. This makes unsupervised machine learning algorithm a powerful surrogate model for screening out the abnormal hydrological data in the context of especially when no labeled datasets are available. Moreover, Halicki et al. (2024) reported that the majority of unsupervised machine learning algorithms are non-parametric, meaning they do not assume that the underlying dataset follows any specific statistical distribution. This makes them particularly suitable for real-world applications where data often exhibit complex, non-normal patterns. Isolation Forest (IF) is an algorithm that was originated by Liu et al. (2008) where it employs binary trees to identify outliers, resulting in a linear time complexity and low memory usage that is appropriate for processing large datasets as well.

Isolation Forest (IF) is based on a main principle that the abnormal data points are scarce and highly distinguishable in a time series. In an Isolation Forest, randomly sub-sampled data are processed through a tree structure built using randomly selected features. The algorithm isolates observations by recursively partitioning the data. Samples that travel deeper into the tree require more splits to be isolated, indicating that they are more similar to the rest of the data and thus less likely to be outliers. In contrast, samples that appear on shorter branches are isolated with fewer cuts, suggesting that they are outliers, as it was easier for the tree to separate them from the majority of observations.

As noted previously that Isolation Forest is an ensemble-based method for outlier detection, consisting of multiple binary decision trees. Each individual tree within the Isolation Forest is referred to as an Isolation Tree (iTree). When a dataset is provided to the Isolation Forest (IF) model, a random sub-sample of the data is first selected and used to construct a binary decision tree, iTree. The tree grows by recursively selecting a feature at random from the available *N* features. A random threshold value is then chosen within the range of the selected feature's minimum and maximum values. For each node, the data is split based on this threshold: if a data point's feature value is less than the threshold, it is directed to the left branch; otherwise, it goes to the right branch. In this way, each node creates a binary split of the data. This recursive partitioning continues until either each data point is completely isolated in a leaf node or a predefined maximum depth is reached. The process described above is repeated multiple times to build a forest of random binary trees, where each tree contributes to evaluating how easily individual points can be isolated.

During the scoring phase, each data point is passed through all the trained trees in the forest. The model calculates an outlier score for each point based on the average path length—that is, the number of splits (tree depth) required to isolate the point across all trees. Points that are isolated quickly (i.e., with shorter path lengths) are likely to be outliers, as fewer conditions are needed to separate them from the rest of the data. In contrast, normal points tend to have longer path lengths due to their similarity to many other observations. The final outlier score is a normalized function of the average path length, and it typically ranges between 0 (most normal) and 1 (most anomalous). A threshold based on the contamination parameter—which specifies the expected proportion of outliers in the dataset—is used to classify each point. Often, a score close to 1 indicates an outlier, while a score closer to 0 indicates a normal point. The typical Isolation Trees constructed for outlier detection using Isolation Forest algorithm is illustrated in Figure 1.

The mathematical expressions which are used to calculate the outlier score s(x,n) based on the path length of a data point in the tress can be found in Equation (1)-(4):

$$s(x,n) = 2^{-\frac{h(x)}{c(n)}} \quad \text{(Eq.1)}$$

$$c(n) \approx 2H(n-1) - \frac{2(n-1)}{n} \quad \text{(Eq.2)}$$

$$H(i) = \ln(i) + \gamma \quad \text{(Eq.3)}$$

$$h(x) = \frac{1}{t}\sum_{i=1}^{t} h_i(x) \quad \text{(Eq.4)}$$

Where *n* is the number of samples used to build each tree, *c(n)* indicates the normalization factor which represents the average path length of unsuccessful searches in Binary Search Tree (BST) of *n* points, *H(i)* informs the harmonic number with *γ* is Euler constant of approximately 0.5772, and *ht(x)* denotes the path length in each tree, *t*. It has been reported that the only drawback of IF is the users need to specify the contamination parameter manually in which the defined value sets the decision threshold where the ranks of data points of outlier score exceed that specified contamination level are labeled as outliers. In other words, contamination level of 0.01 represents 1% of the data will be treated to be anomalous.

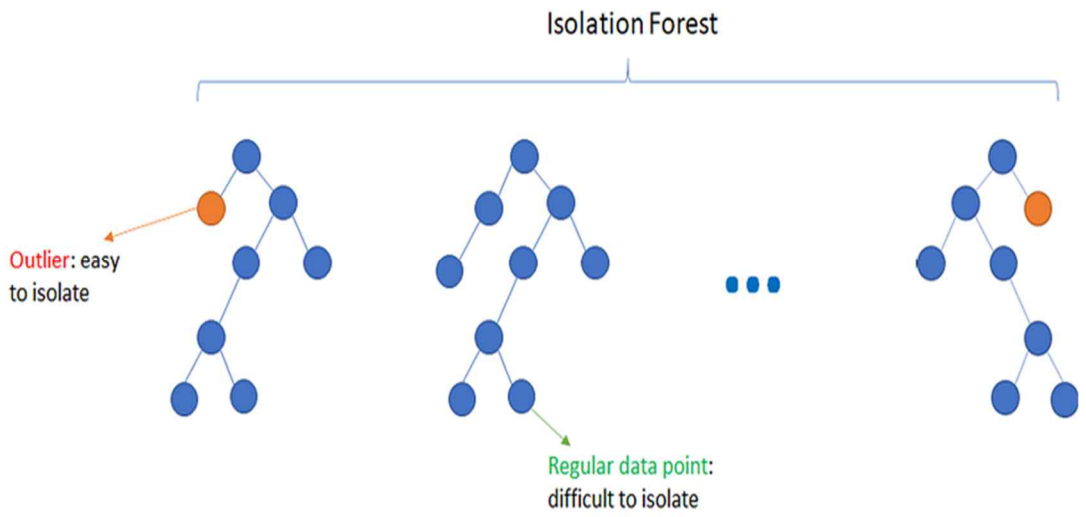

Fig 1. Illustration of outlier detection using Isolation Tree

1.2 Outlier Detection Using Supervised Machine Learning Algorithm – XGBoost

XGBoost (eXtreme Gradient Boosting) is an open-source machine learning algorithm introduced by Chen (2016), widely used for training classification and regression models. It is primarily applied to supervised learning problems, where input features xi are used to predict a target variable y_i. XGBoost has also been successfully adopted for quality control and outlier detection in hydrological time series, provided that sufficient labeled data is available.

As a supervised learning model, XGBoost learns from historical labeled datasets. In the context of outlier detection, it builds an ensemble of decision trees using gradient boosting, where each tree identifies patterns that distinguish between normal and abnormal instances in the hydrological data. This requires a labeled training dataset consisting of input variables (e.g., rainfall or water level time series) and corresponding class labels. Typically, a label of 0 denotes normal data and 1 denotes abnormal data, or vice versa.

During training, XGBoost minimizes classification errors by assigning greater weight to instances that were previously misclassified. This iterative process continues until a strong predictive model is constructed. XGBoost is capable of learning complex non-linear relationships and interactions among features, making it particularly effective at detecting subtle outliers. When new or unlabeled data is introduced to the trained model, it passes through the ensemble of trees. The model then produces an output—either a probability or a binary classification—indicating whether the input is normal or abnormal.

The core mathematical foundations of XGBoost can be expressed as it predicts an output $\hat{y}_i$ as the sum of K regression trees:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i),\ f_k \in F \qquad \text{(Eq.5)}$$

Where f_k are functions from the space of regression trees. Then, XGBoost optimizes a regularized objective function ($L(\phi)$) as depicted in Equation (6)-(7) that balances model fit and complexity:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \quad \text{(Eq.6)}$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \qquad \text{(Eq.7)}$$

Where l is a differentiable loss function and function in Equation (7) penalizes complexity with T being the number of leaves and w being their weights on leaves, while γ, λ are just the regularization parameters. The objective is consequently approximated at each boosting iteration by using a second-order Taylor expansion as presented in Equation (8).

$$\mathcal{L}^t \approx \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + \Omega(f_t)$$

(Eq.8)

Where g_i and h_i are the first and second derivatives of the loss function with respect to previous predictions. The optimal weight for each leaf is then achieved through Equation (9) and the best tree structure minimizes the regularized loss using these weights.

$$w_j^* = \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \qquad \text{(Eq.9)}$$

Where I_j represents set of instances in leaf j. These mathematically efficient frameworks make XGBoost both powerful and effective at maximizing the separation between normal and abnormal patterns through gradient-based optimization and tree-based splits, making it a popular machine learning model for outlier detection in hydrological data.

The performance of a XGBoost model can be significantly enhanced through hyper-parameter optimization, which fine-tunes the model's settings to achieve better accuracy. Usually, the parameters listed as follows are crucial when configuring XGBoost for outlier detection.

1) *Max_depth*: It controls the maximum depth of the trees. Deeper trees can model more complex patterns but may lead to overfitting.

2) *Learning_rate*: It determines the step size at each iteration while moving toward a minimum of the loss function. A lower learning rate can improve model performance but requires more boosting rounds.

3) *n_estimators*: The number of trees in the ensemble. More trees can lead to better performance but also increase computation time.

4) *Sub-sample*: The fraction of samples to be used for fitting individual base learners. This parameter helps prevent overfitting.

The output of XGBoost for each data point is the predicted label (e.g., normal or abnormal) or a probability score. Several metrics are used to evaluate the effectiveness of XGBoost model in detecting outliers. Metric of accuracy informs the proportion of true results among the total number of cases examined. In the context of outlier detection, high accuracy indicates that the model correctly identifies both normal and anomalous instances. Precision tells the ratio of true positive observations to the total predicted positives. High precision means that the model has a low false positive rate. Recall represents the ratio of true positive observations to all actual positives. High recall indicates that the model successfully identifies most of the actual outliers.

1.3 Water Level Outlier Correction Using Deep Learning Algorithm – LSTM Auto-encoder

In time series data, outliers frequently occur due to sensor malfunctions, data loss, or extreme external influences. In the fields of hydrology and meteorology, such anomalies can significantly degrade the accuracy of model training and prediction; therefore, identifying and appropriately correcting or infilling abnormal data is an essential process. Recently, deep learning–based auto-encoders have been widely applied to data quality management, and among them, the LSTM (Long Short-Term Memory) auto-encoder has gained particular attention for its ability to learn long-term dependencies in time series data.

An auto-encoder is a neural network architecture that compresses input data through an encoding process and then reconstructs it through a decoding process, with the input and output having the same structure. Through this process, the network automatically learns latent patterns and essential features embedded in the data. Unlike conventional auto-encoders, an LSTM auto-encoder incorporates LSTM units in its hidden layers, enabling it to capture temporal correlations and both short- and long-term dependencies inherent in time-series data. The encoder compresses the input sequence from a high-dimensional space into a low-dimensional latent space, while the decoder reconstructs the original sequence based on this latent representation. Figure 2 illustrates the typical architecture of a LSTM auto-encoder model.

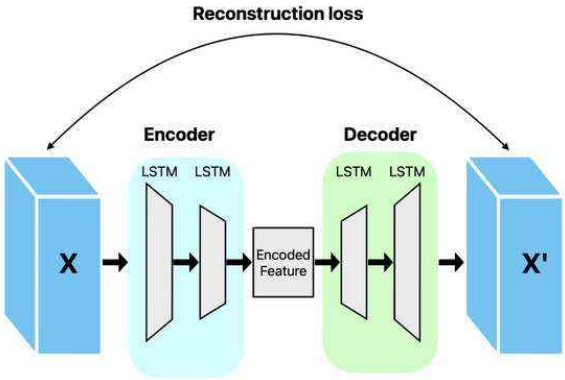

Fig 2. Typical architecture of a LSTM auto-encoder model

1.3.1 Principle of Outlier Correction and Infilling

Outlier correction using an LSTM auto-encoder generally consists of two main steps. First, the model is trained to learn normal time-series patterns. At this stage, the training dataset should contain as few anomalies as possible or have anomalies removed in advance, allowing the model to memorize typical temporal behaviors observed under normal conditions. Second, when new data are provided as input, the auto-encoder reconstructs the corresponding sequences. Even if anomalies are present in the input, the network attempts to restore values that conform to the learned normal patterns. As a result, a large discrepancy between the input and reconstructed output indicates the presence of anomalies, while the reconstructed values themselves can be used for anomaly correction and data infilling.

In other words, the LSTM auto-encoder learns the normal distribution of the data and naturally replaces anomalous segments by reconstructing them in a form that is most consistent with normal patterns. Compared to traditional interpolation methods such as linear interpolation or polynomial regression, this approach has the advantage of capturing complex nonlinear relationships in time series data.

1.3.2 Reconstruction Process and Computation of Values

Reconstruction is the core mechanism of an LSTM auto-encoder. Input data are divided into fixed-length sliding window sequences before being fed into the model. The encoder LSTM accumulates information at each time step through its hidden and cell states to produce a final sequence representation. This compressed vector is then passed to the decoder LSTM, which sequentially reconstructs the output sequence with the same length as the input.

During training, model parameters are optimized to minimize the difference between the input and reconstructed sequences. Mean Squared Error (MSE) is commonly used as the loss function, and a smaller reconstruction error indicates that the model has learned normal patterns more accurately. When anomalous inputs are provided, the model tends to generate outputs that approximate normal patterns rather than reproducing the anomalous values themselves. Consequently, the reconstructed outputs may differ from the original inputs, and this difference can be used for both anomaly detection and correction.

Furthermore, when reconstructing an entire time series, a sliding window approach allows each time step to be reconstructed multiple times. The final reconstructed value at each time point is obtained by taking the average or median of overlapping reconstructed values. Assume that there are 6 data points in a time series and 3 sliding window size has been adopted in a LSTM auto-encoder model to reconstruct these 6 data points. So three data will be reconstructed in each step until all the data are reconstructed. Since more than one data will be reconstructed in each step, so there will be overlapping reconstructed data in particular data point and the average or

median of the overlapping reconstructed data will be taken to ensure that there will be only one reconstructed value in each data point. This strategy reduces the influence of localized anomalies and provides more stable and reliable reconstruction results.

For example, assume that we have 6 original time series data as mentioned in previous paragraph with sliding window size of 3:

- Original time-series data: (1, 2, 3, 4, 5, 6)

- Reconstructed values from the 1st sliding window: (1, 1, 1, 0, 0, 0)

- Reconstructed values from the 2nd sliding window: (0, 2, 2, 2, 0, 0)

- Reconstructed values from the 3rd sliding window: (0, 0, 3, 3, 3, 0)

- Reconstructed values from the 4th sliding window: (0, 0, 0, 4, 4, 4)

- **Final reconstructed data (representative values based on averaging)**: (1, (1+2)/2, (1+2+3)/3, (2+3+4)/3, (3+4)/2, 4) = (1, 1.5, 2, 3, 3.5, 4)

In this process, the data at each time step are reconstructed multiple times. For instance, the third data point is reconstructed in the first window (steps 1–3), the second window (steps 2–4), and the third window (steps 3–5), resulting in a total of three reconstructed values. To consolidate these overlapping reconstructions into a single value, a representative statistic such as the mean (average) or the median is computed, and this value is selected as the final reconstructed value for that time step.

The LSTM auto-encoder based outlier correction technique has high applicability in various time-series–based fields, including hydrology, meteorology, and environmental data management. This approach enables the automatic correction of sensor errors and missing data periods in hydrological observation records and can produce more refined and context-aware values compared to conventional simple regression-based methods. Owing to these advantages, the LSTM auto-encoder is utilized for anomaly and missing data correction and is planned to be incorporated into the development of hydrological data quality control system in TC member countries. However, depending on the learning strategy adopted, LSTM auto-encoder can be classified into unsupervised and supervised approaches.

1.3.3 Unsupervised LSTM Auto-encoder

An unsupervised LSTM auto-encoder refers to a model in which the input and output are identical. In other words, the model is trained to reconstruct the same time-series segment (e.g., water level, discharge, temperature) that is provided as input. Through this process, the model automatically learns how to compress (encode) and reconstruct (decode) the normal patterns embedded in the data.

During training, the model weights are optimized to minimize the difference between the input sequence and the reconstructed sequence. As a result, the trained model exhibits low reconstruction errors for normal patterns and high reconstruction errors for anomalous or abnormal segments. Therefore, the unsupervised approach does not require labeled data and can be trained solely using the observed data itself. It is widely applied in anomaly detection, missing data infilling, and data quality control.

For example, if both the input and output consist of [Water Level, Rainfall], the model is trained to reconstruct the combined time-series patterns of water level and rainfall. Although rainfall influences the reconstruction of water level, the model still aims to reproduce the entire input sequence, and thus this approach is classified as unsupervised learning.

Table 1. Comparisons between unsupervised and supervised LSTM auto-encoder

Category	Unsupervised	Supervised
Input (X)	Original time-series data (e.g., water level, rainfall)	Multivariate time-series data (e.g., rainfall + historical water level)
Output (Y)	Same as input (e.g., water level, rainfall)	Target variable only (e.g., water level)
Learning method	Direct reconstruction of input ($X \approx Y$)	Prediction of target from input ($X \rightarrow Y$)
Label required	Not required (trained using data itself)	Required (ground-truth target variable needed)
Primary purpose	Anomaly detection, missing data infilling, pattern learning	Prediction (regression), input–output mapping
Interpretation of error	Discrimination between normal and anomalous segments	Evaluation of prediction accuracy
Structural characteristics	Encoder–decoder architecture with full reconstruction	Encoder–decoder architecture retained, but output is constrained

1.3.4 Supervised LSTM Auto-encoder

A supervised LSTM auto-encoder is used when the input and output differ. For instance, if the input consists of [Rainfall, Water Level] and the output consists of [Water Level], the model is trained to reconstruct or predict water level values based on rainfall and historical water level data. In this case, since the output is either a subset of the input or an entirely different target variable, labeled data (ground-truth water level values) are required.

In practice, this approach is closer to a sequence-to-sequence prediction model rather than a traditional auto-encoder. During training, the model directly learns the mapping from input X to output Y, which clearly falls under the category of supervised learning. The term “auto-encoder” is used primarily because the model adopts an encoder–decoder architecture; however, the learning paradigm itself is supervised.

In summary, when an LSTM auto-encoder reconstructs the same data provided as input, it is classified as **unsupervised learning**, whereas when it predicts a different target variable, it is classified as **supervised learning**. The comparisons between the unsupervised and supervised LSTM auto-encoders are summarized in Table 1.

1.4 Rainfall Outlier Correction Using Machine Learning Algorithm – XGBoost Regressor

Rainfall outliers can significantly distort statistical analysis, bias model calibration, and degrade the performance of either physical or machine learning-based forecast model. Therefore, robust rainfall data quality control and outlier correction are essential preprocessing steps prior to hydrological modeling and operational forecasting. However, it has been found that the deep learning LSTM auto-encoder algorithm as discussed in previous section is not appropriate for outlier correction and infilling of rainfall time series data. Traditional approaches for rainfall outlier correction such as threshold-based rules, spatial consistency checks, and simple statistical methods (moving averages) are computationally efficient and easy to implement, but they often struggle to capture the nonlinear relationships and complex-temporal dependencies inherent in rainfall processes.

In recent years, machine learning techniques have emerged as powerful tools for rainfall data quality control. Among them, XGBoost (Extreme Gradient Boosting) has gained considerable attention due to its high predictive accuracy, robustness to noise,

and ability to model nonlinear interactions among multiple predictors. The XGBoost regressor, in particular, provides an effective framework for estimating corrected rainfall values by learning from historical observations and auxiliary information such as neighboring station data, meteorological variables, and temporal features. Therefore, XGBoost regressor machine learning algorithm is introduced for correction and infilling of rainfall time series data in TC member countries.

1.4.1 Overview of XGBoost Regressor

XGBoost is an ensemble learning algorithm based on gradient boosting of decision trees. It builds a series of regression trees sequentially, where each new tree is trained to minimize the residual errors of the previous ensemble. The final prediction is obtained by summing the contributions of all trees, weighted by a learning rate.

The XGBoost offers several advantages for rainfall outlier correction: (1) It can model nonlinear and non-additive relationships between rainfall and predictor variables; (2) It is robust to multi-collinearity and missing values; (3) It includes built-in regularization to reduce overfitting; (4) It efficiently handles large datasets and high-dimensional feature spaces. These properties make XGBoost particularly suitable for rainfall correction tasks, where relationships between rainfall observations at different stations and times are often complex and nonlinear.

1.4.2 Methodology of Rainfall Outlier Correction

The application of the XGBoost regressor for rainfall outlier correction generally follows several steps. First, feature construction is performed. Input features may include rainfall measurements from neighboring stations, lagged rainfall values from the target station, temporal indicators (e.g., hour, day, season), and auxiliary meteorological variables such as temperature, humidity, or radar-based rainfall estimates. These features provide contextual information that helps the model infer realistic rainfall values. Second, a training dataset is prepared using historical rainfall data that have been quality-checked or assumed to be reliable. The target variable is the rainfall at the station of interest, while the input features represent the surrounding spatial–temporal conditions.

Third, the XGBoost is trained to learn the mapping between the input features and the target rainfall. During training, the model minimizes a loss function such as mean squared error (MSE), while regularization terms control model complexity. Once trained, the model is applied to periods identified as containing outliers. Instead of using the suspicious observed value, the model generates a predicted rainfall value based on learned relationships. This predicted value is then used as the corrected rainfall value. In practice, correction may be applied selectively only when the difference between observed and predicted values exceeds a predefined threshold, so as to avoid unnecessary modification of valid observations.

Compared to conventional regression or rule-based approaches, the XGBoost-based method provides several key advantages. First, it captures complex nonlinear dependencies between rainfall data collected at different stations, which is particularly important during convective or extreme rainfall events. Second, it adapts flexibly to different climatic regimes without requiring explicit physical assumptions. Third, it can incorporate a wide range of auxiliary data sources, improving robustness and generalizability.

However, due to the relatively complex model structure, hyper-parameter selection has a significant impact on performance, and the training time is comparatively long.

Therefore, optimization may be required when applying the model to real-time operational systems. Nevertheless, the XGBoost regressor is regarded as one of the most powerful models for rainfall outlier correction, offering superior representational capacity and predictive performance. As a result, it demonstrates strong potential as a core algorithm for rainfall anomaly/outlier correction within hydrological data quality control systems.

In conclusion, the machine learning algorithms of unsupervised Isolation Forest and supervised XGBoost will be adopted to identify outliers in hydrological time-series data (rainfall and water level) during the development of the hydrological data quality control system for application in TC member countries. In addition, the system will be equipped with the capability to correct the identified rainfall and water-level outliers using the machine learning algorithm XGBoost regressor and the deep learning algorithm LSTM auto-encoder. In other words, the developed hydrological data quality control system provides comprehensive coverage of hydrological data quality control, ranging from outlier identification to correction, by leveraging various AI-based techniques.

1.5 Demonstration on Hydrological Data Quality Control System Using AI Techniques

A PC desktop version of flood forecasting system using several machine learning algorithms based on Python language has been successfully developed and it is currently under final checking and tuning before distribution to the target TC member countries for applications in real-world hydrological data quality control. This hydrological data quality control system consists of 3 major modules for each type of hydrological data (rainfall and water level), which are:

Module 1: Criteria Establishment

Module 2: Data Correction

Module 3: Report

The user interfaces and functions for each module are illustrated and elaborated respectively in the following section.

(Module 1 for Rainfall Data) Criteria Establishment – Input Data Features

1) The system is designed to allow the user to insert up to 5 rainfall input features by clicking button (Import).

2) The input features' data should be stored in file of *csv*. format where the first column represents the rainfall input data for the station under studied while the remaining columns indicate the rainfall input data for the nearby stations.

3) A graph will be automatically generated by plotting the rainfall time series data and the user is allowed to choose which rainfall data to be displayed in the graph by clicking the button (Station) and select the desired station from the list.

4) The user should also specify the time interval of the inserted input data whether they are daily, hourly, or 10-minute.

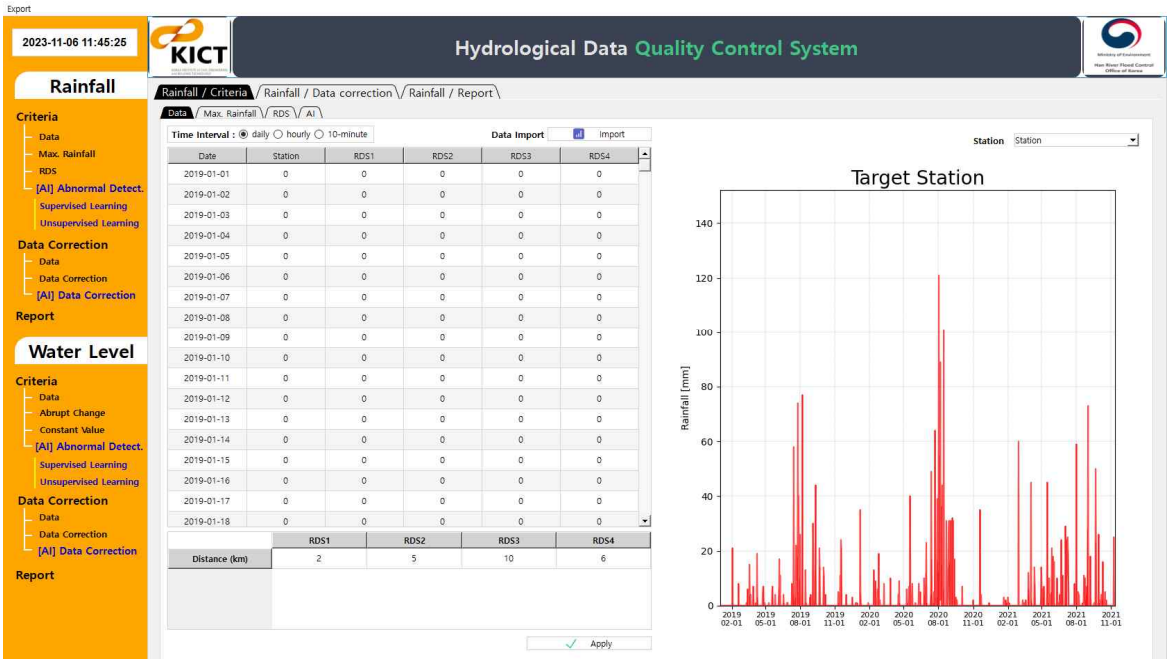

Fig 3. (Module 1 for Rainfall Data) Criteria Establishment – Input Data Features

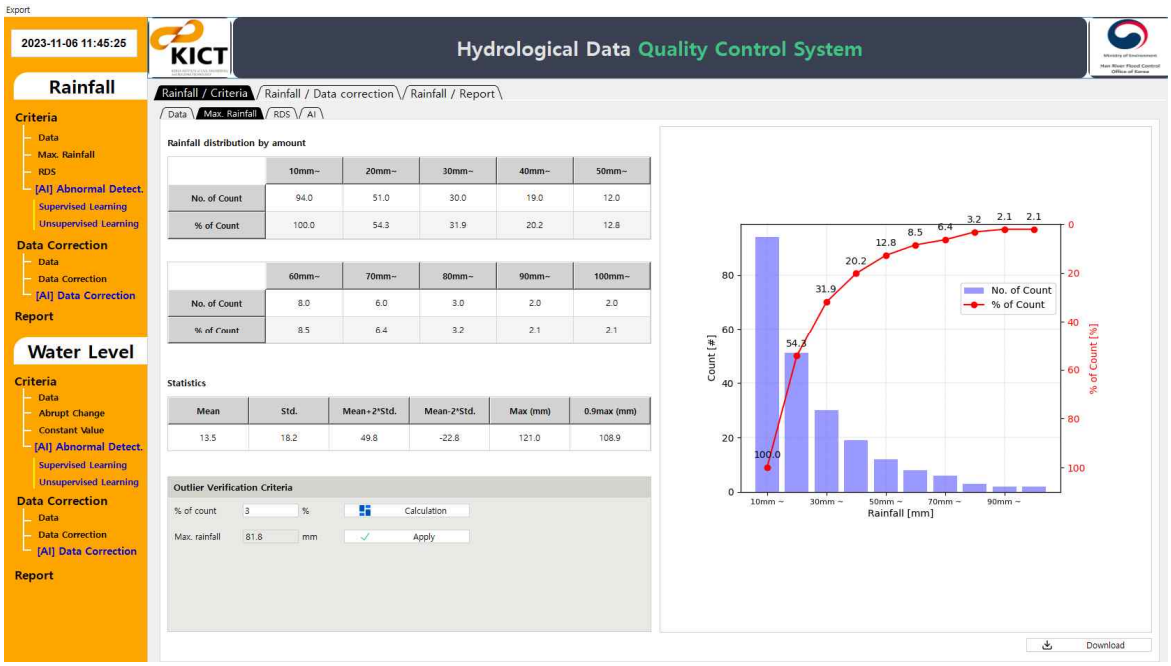

Fig 4. (Module 1 for Rainfall Data) Criteria Establishment – Maximum Rainfall

5) The distances (in km) of the nearby stations should also be defined by the user to facilitate the computation of Reciprocal Distance Squared (RDS) rainfall in later stage.

(Module 1 for Rainfall Data) Criteria Establishment – Maximum Rainfall

1) The outlier verification criteria for maximum rainfall is defined by specifying the percentage of count of the inserted data (station under studied). For example, if we take 3 % of the total data distribution in the right tail as the threshold, then 3 should be inserted and the corresponding maximum rainfall threshold will be computed automatically by clicking the button (Calculation).

2) The graph of total data distribution for maximum rainfall can be saved by clicking the button (Download).

3) This maximum rainfall threshold will be adopted as the criteria for identifying the outliers of maximum rainfall by clicking the button (Apply).

(Module 1 for Rainfall Data) Criteria Establishment – Reciprocal Distance Squared (RDS) Rainfall

1) The RDS rainfall is computed automatically when the “RDS” tab is clicked and the rainfall time series graphs for all the stations are displayed.

2) The outlier verification criteria for RDS can be defined by clicking the “RDS Criteria” tab and the corresponding thresholds for upper and lower differences will be computed accordingly once the verification criteria is specified by clicking the button (Calculation).

3) For exaple, if we take 5% at left and right tail of the total data distribution (differences between station rainfall and RDS rainfall) as the outlier verification criteria, then 5 should be defined as the upper and lower limit.

4) The graph of total data distribution for differences between station rainfall and RDS rainfall can be saved by clicking the button (Download).

5) The computed thresholds will be adopted as the criteria for identifying the outliers of RDS by clicking the button (Apply).

Fig 5. (Module 1 for Rainfall Data) Criteria Establishment – Reciprocal Distance Squared (RDS) Rainfall

(Module 1 for Rainfall Data) Criteria Establishment – Using Artificial Intelligence (AI)

1) The rainfall outlier identification using A.I techniques can be facilitated by clicking the "AI" tab.

2) The rainfall outlier identification using supervised machine learning – XGBoost and unsupervised machine learning – Isolation Forest will be prompted at the left and right hand side of the interface.

3) For XGBoost, the user is required to specify the length for training and testing to initial the verification by clicking the button (Apply) after specifying the desired values.

4) For example, 7 should be inserted for training if 70% of the inserted data will be used for training purpose and 3 for testing if

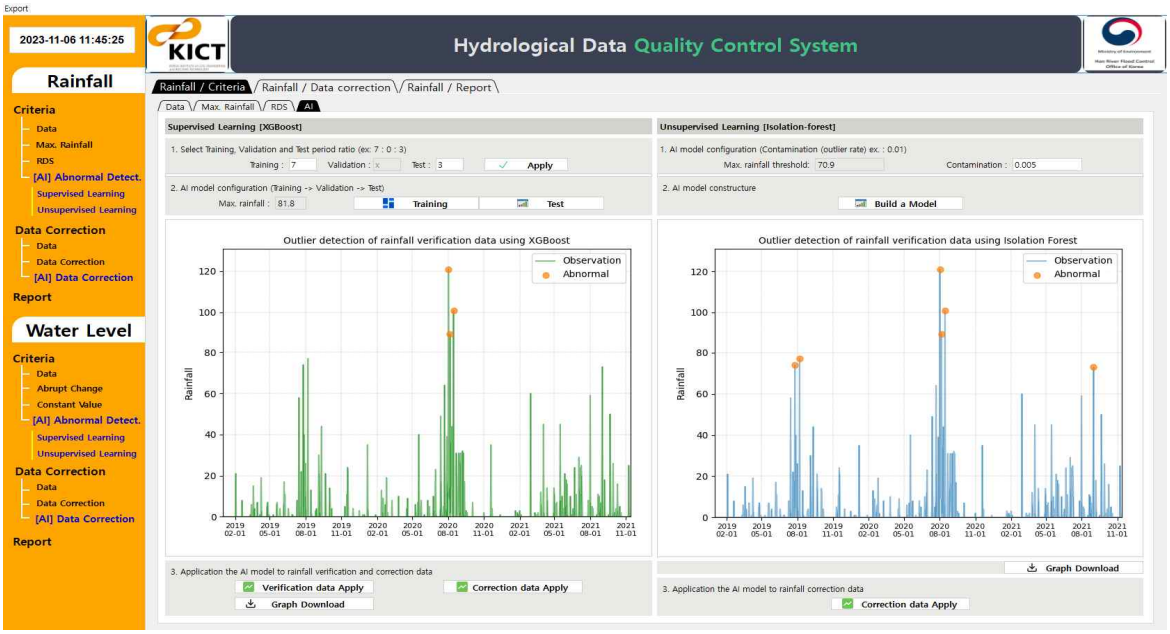

Fig 6. (Module 1 for Rainfall Data) Criteria Establishment – Using Artificial Intelligence (AI)

the remaining 30% are intended for testing purpose.

5) XGBoost is a supervised learning algorithm and thus it requires labeled data to "inform" the model how to differentiate the outlier from the normal data. The threshold values computed in previous stages for maximum rainfall and RDS can be adopted as the labeled data for XGBoost supervised learning.

6) The XGBoost model for outlier identification is trained and tested by clicking the button (Training) and (Test).

7) For Isolation Forest, the user is only required to specify the contamination level for outlier identification, where contamination level of 0.01 should be specified if 1% of the total data distribution at both tails are considered as outliers.

8) The Isolation Forest model for outlier identification is established by clicking the button (Build a Model).

9) The outliers identified from XGBoost and Isolation Forest are adopted for verification by clicking the button (Verification data Apply).

10) The graph which is showing the rainfall time series as well as the identified outliers can be saved by clicking the button (Download).

(Module 2 for Rainfall Data) Data Correction

1) The rainfall outlier verification criteria that have been established in previous stage are used to verify or check the new rainfall time series that is desired to be corrected.

2) The new rainfall data input file can be inserted by clicking the button (Import) under the Rainfall/Data Correction tab.

3) The format of the new rainfall data input file should be similar to the one used to establish the outlier verification criteria.

4) The imported new rainfall data quality are assessed according to the outlier verification criteria established previously by clicking the button (Verification) under the "Data Correction" tab.

5) The outliers identified are then corrected based on the computed RDS value by clicking the button (Auto Correction) and (Final Correction). The data indicated in orange color represent maximum rainfall outliers while the data in green color represent the RDS outliers.

6) The summary of data corrected for maximum rainfall and RDS can be found at 'Correction Result".

7) 9 categories of data quality class have been adopted to describe the quality status of the rainfall data

i) Class 0100 – Good time series data

ii) Class 0200 – Data with modifying abnormal value

iii) Class 0211 – Max. rainfall outlier (an excessive amount of rainfall data)

iv) Class 0221 – Lower RDS outlier

v) Class 0222 – Upper RDS outlier

vi) Class 0300 – Data with modifying missing value

vii) Class 0400 – Unidentified data (which is necessary to be reviewed and processed by operator)

viii) Class 0500 – Manual corrected data by operator

ix) Class 0991 – Missing rainfall data

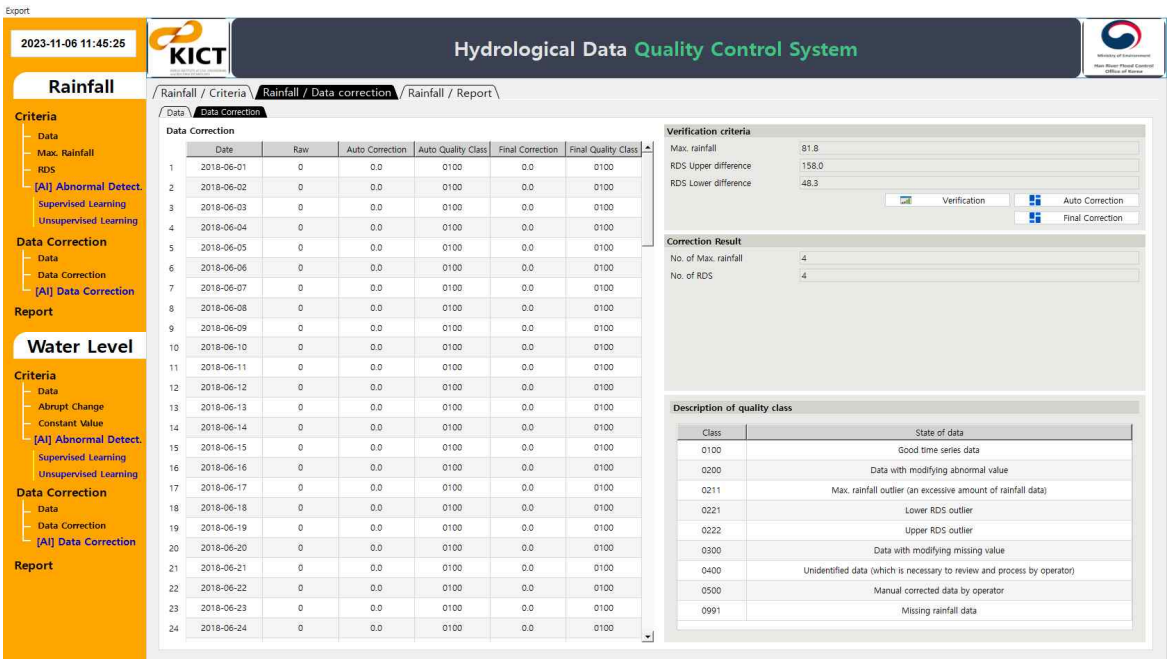

Fig 7. (Module 2 for Rainfall Data) Data Correction

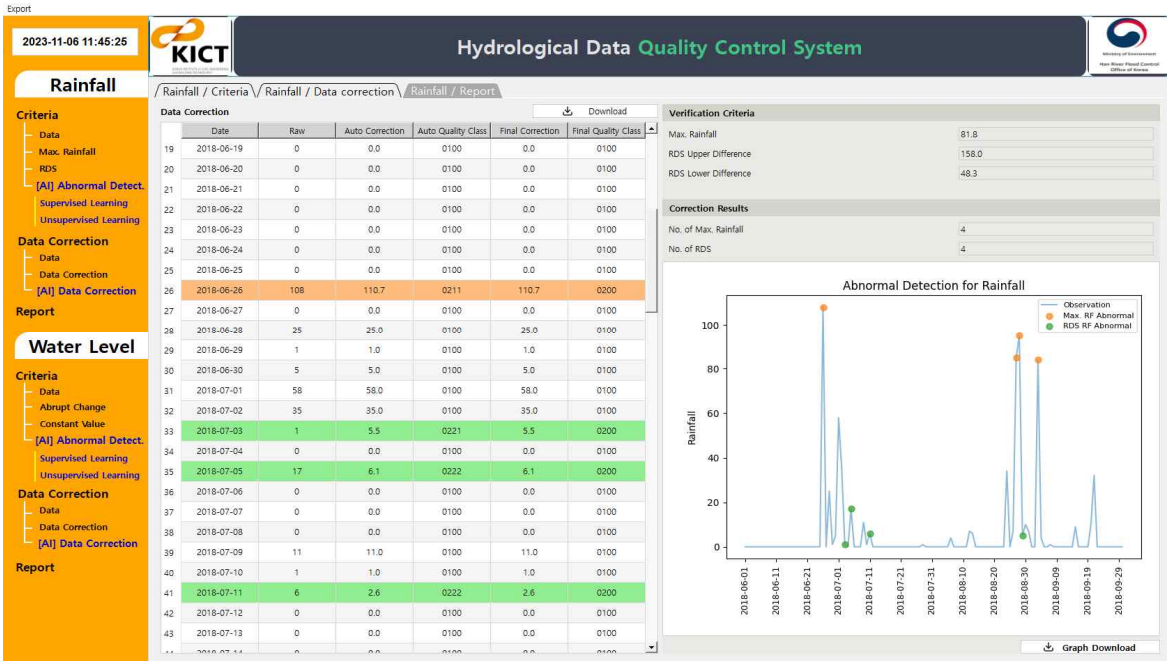

Fig 8. (Module 3 for Rainfall Data) Data Report

8) The data correction using AI techniques is still not yet available while this manual is being drafted and it will be featured into the system in near future.

(Module 3 for Rainfall Data) Report

1) The summary and report of the identified outliers and corrected rainfall data are available under the “Rainfall/Report” tab.

2) The summary in tabulated form can be saved by clicking the (Download) button.

3) The summary in graphical form can be saved by clicking the (Graph Download) button.

(Module 1 for Water Level Data) Criteria Establishment – Input Data Features

1) The system is designed to allow the user to insert a water level time series by clicking button (Import), which should be stored in file of *csv*. format.

2) The user should also specify the time interval of the inserted input data whether they are daily, hourly, or 10-minute.

3) A graph that is showing the water level time series will be shown at the right hand side of the interface once the data is successfully imported.

(Module 1 for Water Level Data) Criteria Establishment – Water Level Abrupt Change Ratio

1) The water level abrupt change ratio between the water level at present time step and water levels at pervious 1 and 2 time steps is calculated automatically by clicking the “Abrupt Change” tab.

2) The outlier verification criteria for water level abrupt change can be defined by clicking the “Abrupt Change Criteria” tab and

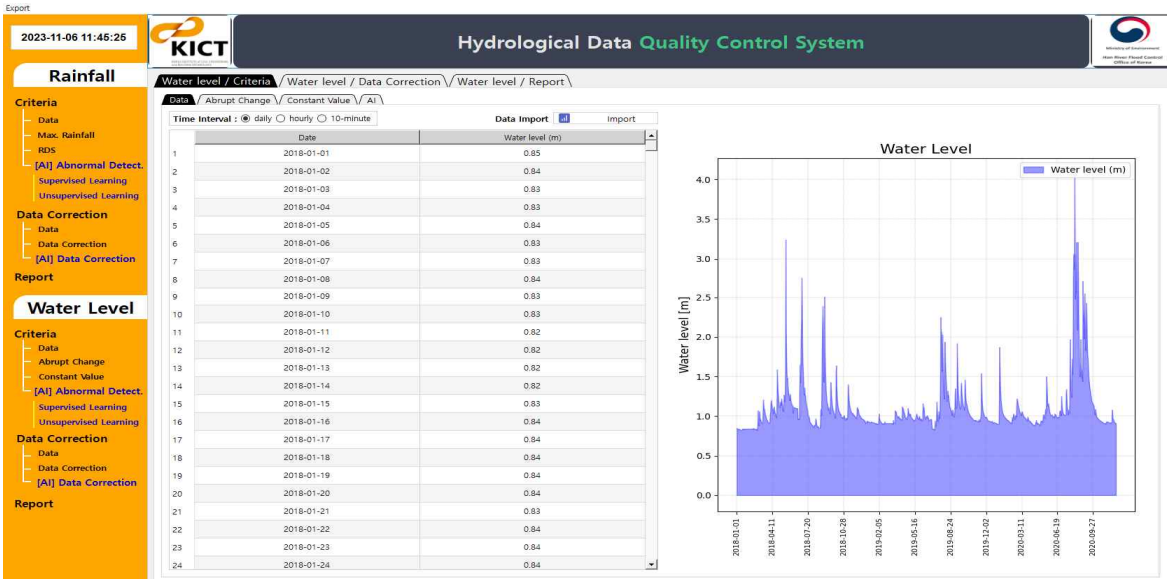

Fig 9. (Module 1 for Water Level Data) Criteria Establishment - Input Data Features

(Module 1 for Water Level Data) Criteria Establishment – Water Level Abrupt Change Ratio

1) The water level abrupt change ratio between the water level at present time step and water levels at pervious 1 and 2 time steps is calculated automatically by clicking the “Abrupt Change” tab.

2) The outlier verification criteria for water level abrupt change can be defined by clicking the “Abrupt Change Criteria” tab and the corresponding thresholds for upper and lower changes will be computed accordingly once the verification criteria is specified by clicking the button (Calculation).

3) For example, if we take 1% at left and right tail of the total data distribution (ratio computed between the water level at present time step and water levels at pervious 1 and 2 time steps) as the outlier verification criteria, then 1 should be defined as the upper and lower limit.

4) The graph of total data distribution for water level change ratio can be saved by clicking the button (Download).

5) This ratio threshold will be adopted as the criteria for identifying the outliers of water level abrupt change by clicking the button (Apply).

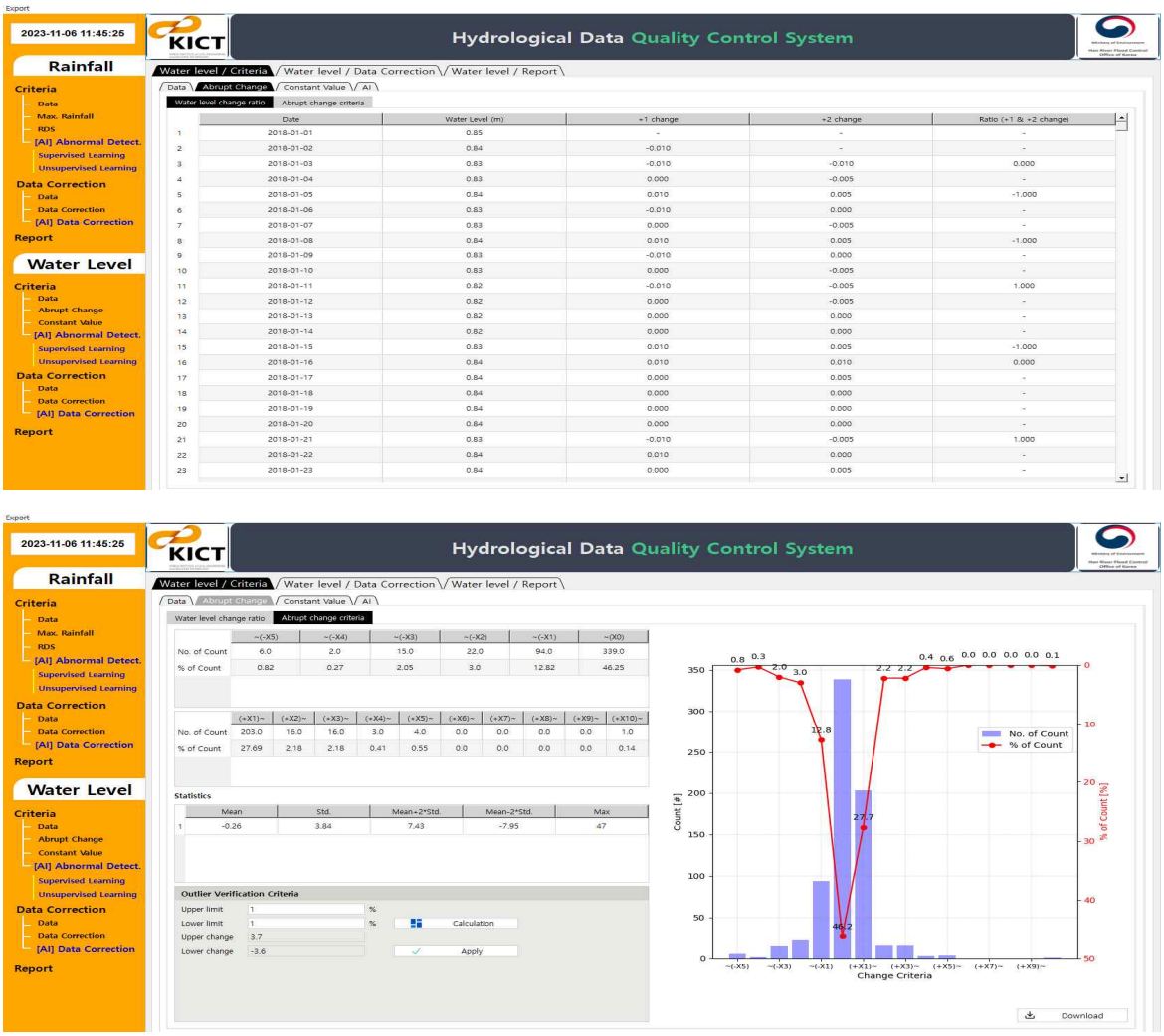

Fig 10. (Module 1 for Water Level Data) Criteria Establishment – Water Level Abrupt Change Ratio

(Model 1 for Water Level Data) Criteria Establishment – Constant Water Level

1) The statistic of constant water level at specified time interval is calculated automatically when the “Constant Value” tab is clicked.

2) The corresponding thresholds for maximum period of constant water level will be computed accordingly once the outlier verification criteria is specified by clicking the button (Calculation).

3) For example, if we take 1% at right tail of the total data distribution (duration with constant water level) as the outlier verification criteria, then 1 should be defined as the percentage of count.

4) The graph of total data distribution for duration of constant water level can be saved by clicking the button (Download).

5) This maximum period threshold will be adopted as the criteria for identifying the outliers of constant water level durations by clicking the button (Apply).

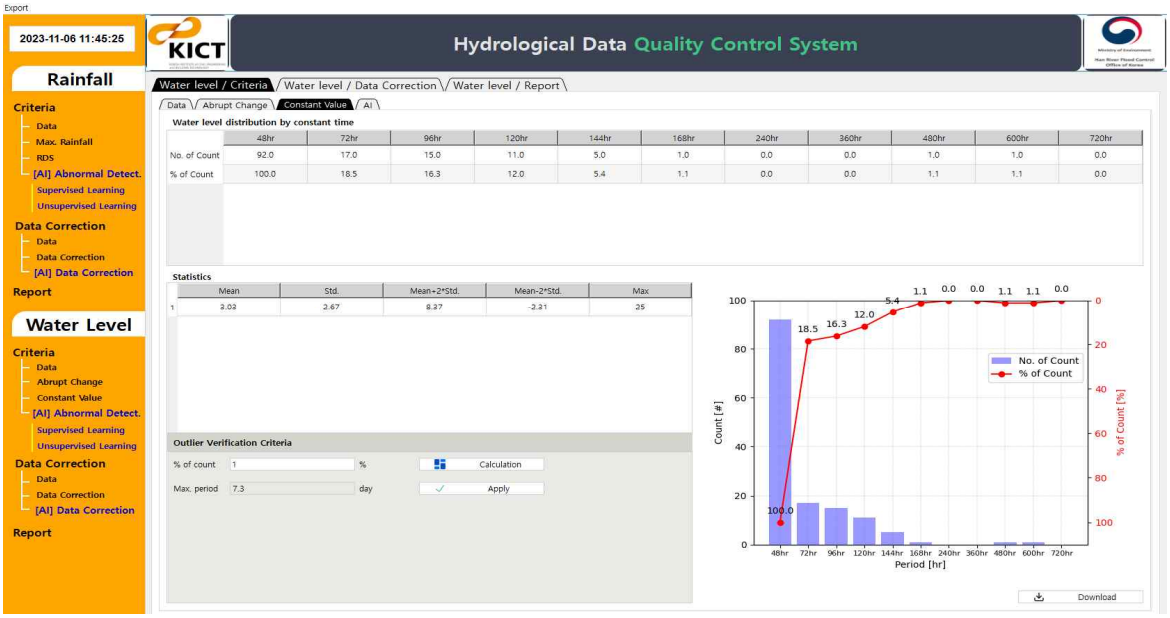

Fig 11. (Module 1 for Water Level Data) Criteria Establishment – Constant Water Level

(Module 1 for Water Level Data) Criteria Establishment – Using Artificial Intelligence (AI)

1) The water level outlier identification using A.I techniques can be facilitated by clicking the “AI” tab.

2) The water level outlier identification using supervised machine learning – XGBoost and unsupervised machine learning – Isolation Forest will be prompted at the left and right hand side of the interface.

3) For XGBoost, the user is required to specify the length for training and testing to initial the verification by clicking the button (Apply) after specifying the desired values.

4) For example, 7 should be inserted for training if 70% of the inserted data will be used for training purpose and 3 for testing if the remaining 30% are intended for testing purpose.

5) XGBoost is a supervised learning algorithm and thus it requires labeled data to “inform” the model how to differentiate the outlier from the normal data. The threshold values computed in previous stages for upper/lower change ratio and constant value can be adopted as the labeled data for XGBoost supervised learning.

6) The XGBoost model for outlier identification is trained and tested by clicking the button (Training) and (Test).

7) For Isolation Forest, the user is only required to specify the contamination level for outlier identification, where contamination level of 0.01 should be specified if 1% of the total data distribution at both tails are considered as outliers.

8) The Isolation Forest model for outlier identification is established by clicking the button (Build a Model).

9) The outliers identified from XGBoost and Isolation Forest are adopted for verification by clicking the button (Verification data Apply).

10) The outlier data points identified for upper, lower change ratio, and constant value are displayed as orange, green, and purple color point respectively.

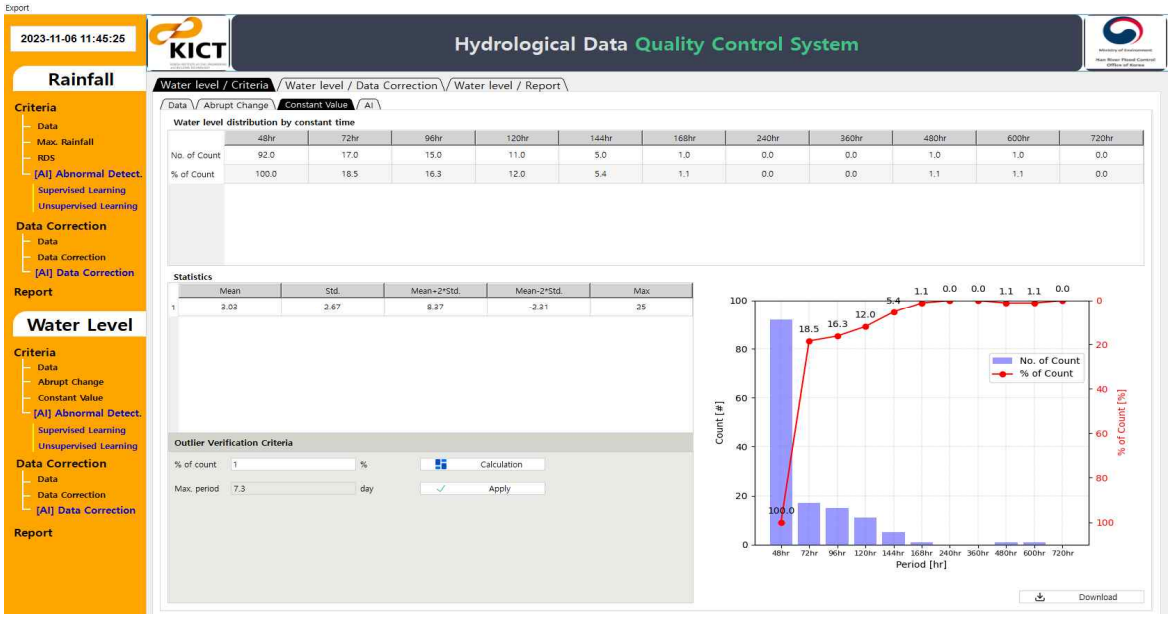

Fig 12. (Module 1 for Water Level Data) Criteria Establishment – Using Artificial Intelligence (AI)

11) The graph which is showing the water level time series as well as the identified outliers can be saved by clicking the button (Download).

(Module 2 for Water Level Data) Data Correction – Inserting Data from Nearby Stations

1) The water level outlier verification criteria that have been established in previous stage are used to verify or check the new water level time series that is desired to be corrected.

2) The new water level data input file can be inserted by clicking the button (Import) under the Water Level/Data Correction tab.

3) The format of the new water level data input file consists of the water level time series that are desired to be corrected in the 1st column, then is followed by the 3 columns of water level time series data collected from either upstream or downstream nearby stations.

4) The water level time series data from upstream or downstream nearby stations are used to compute the univariate regression formula between the water level recorded at the target station and those nearby stations.

(Module 2 for Water Level Data) Data Correction

1) The imported new water level data quality are assessed according to the outlier verification criteria established previously by clicking the button (Verification) under the “Data Correction” tab.

2) The univariate regression formula can be computed by clicking the button (Setup) and a window is prompted out, allowing the user to select the desired periods and pairing nearby station for establishing the regression formula.

3) Apart from that, the user also is available to choose whether the regression is in polynomial or exponential form. A regression formula that gives highest value of coefficient of determination, R2 should be adopted as the best approach for correcting the water level outlier.

4) The outliers identified are then corrected based on the best regression formula by clicking the button (Auto Correction) and

(Final Correction). The data indicated in orange and green color represent upper and lower abrupt water level change ratio outliers while the data in purple color represent the constant value outliers.

5) The summary of data corrected for abrupt water level change and constant value can be found at ‘Correction Result”.

6) 10 categories of data quality class have been adopted to describe the quality status of the water level data

i) Class 0100 – Good time series data

ii) Class 0200 – Data with modifying abnormal value

iii) Class 0231 – Water level data which is changed to “zero” value

iv) Class 0232 – Upper abrupt water level change outlier

v) Class 0233 – Lower abrupt water level change outlier

vi) Class 0241– Water level data which is not change during the specified period

vii) Class 0300 – Data with modifying missing value

viii) Class 0400 – Unidentified data (which is necessary to be reviewed and processed by operator)

x) Class 0500 – Manual corrected data by operator

xi) Class 0992 – Missing water level data

7) The data correction using AI techniques is still not yet available while this manual is being drafted and it will be featured into the system in near future.

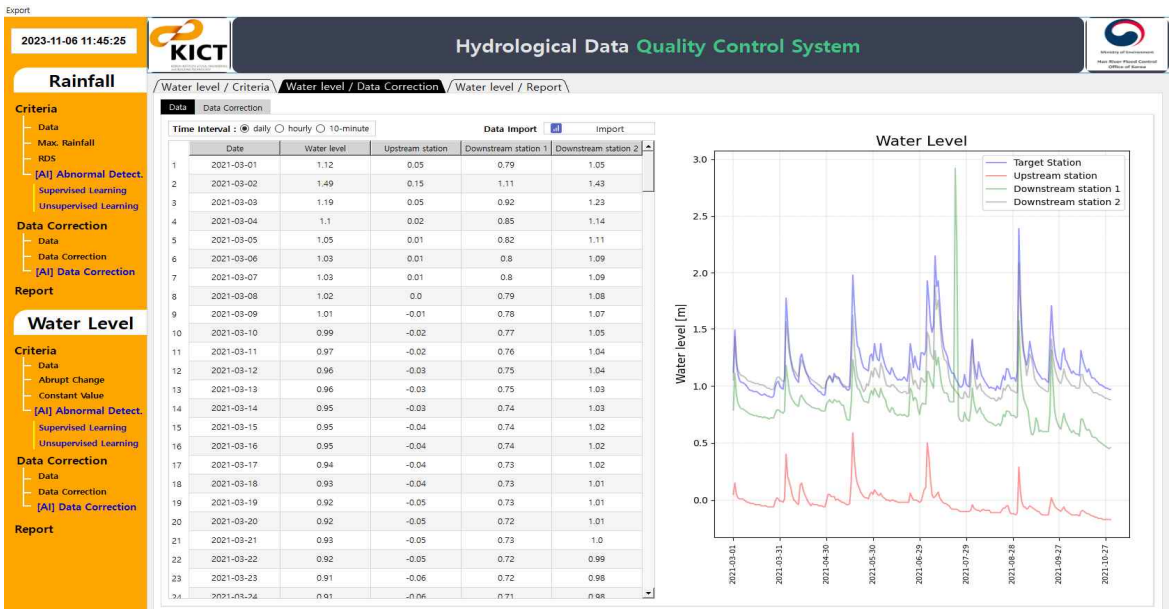

Fig 13. (Module 2 for Water Level Data) Data Correction – Inserting Data from Nearby Stations

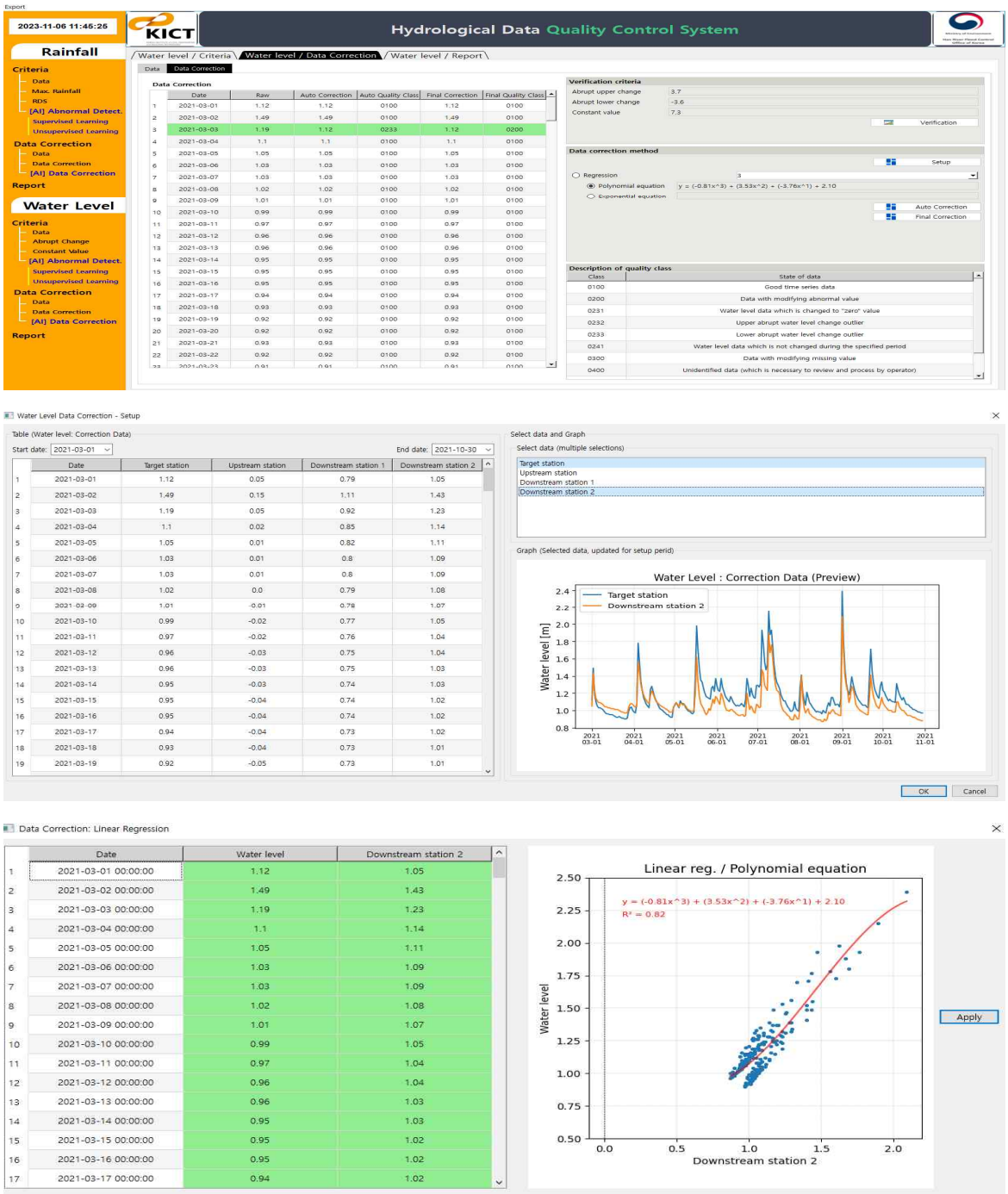

Fig 14. (Module 2 for Water Level Data) Data Correction

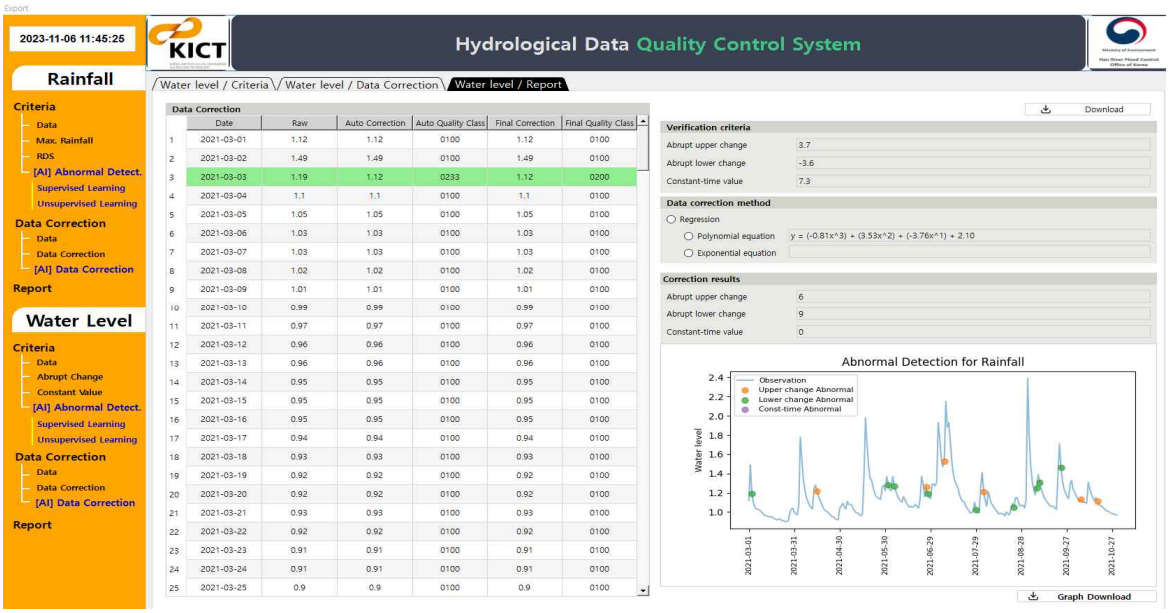

Fig 15. (Module 3 for Water Level Data) Report

(Module 3 for Water Level Data) Report

1) The summary and report of the identified outliers and corrected water level data are available under the “Water Level/Report” tab.

2) The summary in tabulated form can be saved by clicking the (Download) button.

3) The summary in graphical form can be saved by clicking the (Graph Download) button.

2. FLOOD FORECASTING USING AI

Floods are among the most devastating natural disasters, posing significant threats to human lives, property, and infrastructure. Accurate and timely flood forecasting is essential to mitigate these risks, providing authorities with critical information to implement preventative measures, allocate resources, and issue warnings. Traditional flood forecasting methods, such as hydrological models and numerical simulations, rely heavily on physical principles and detailed parameterization of complex watershed systems. However, these methods often require extensive calibration and are constrained by the availability and quality of input data. On the other hand, progress in artificial intelligence and machine learning has enabled data-driven methods like Long Short-Term Memory (LSTM) models, which have demonstrated strong potential in flood forecasting. Guo et al. (2021) highlighted that machine learning models can serve as surrogate models to replace physically based flood simulations, under the assumption that these models are capable of learning the behavior of the target system even without explicitly representing the underlying physical processes, provided that sufficient and representative data are available. However, the authors also acknowledged limitations in their study due to the lack of large-scale flood datasets. This scarcity was attributed to the long computational time required for physically based simulations and the challenges associated with the widespread deployment of sensors for collecting observational data, both of which constrained the predictive performance of their CNN model.

A recent study by Guglielmo et al. (2025) introduced a novel approach for integrating physical principles into data-driven models, demonstrating enhanced predictive performance even when extrapolating beyond the training data boundaries or in data-scarce scenarios. This advancement suggests that applying data-driven models to real-world datasets from complex systems is becoming increasingly feasible and practical. LSTM, a specialized type of recurrent neural network (RNN), has been widely used in sequence-based tasks due to its ability to capture long-term dependencies in data. This characteristic is particularly advantageous in hydrological forecasting, where the dynamics of rainfall-runoff processes exhibit temporal dependencies over extended periods. By learning patterns directly from historical data, LSTM models bypass the need for explicit physical equations, enabling them to model complex, nonlinear relationships between inputs (e.g., rainfall, temperature, and river flow) and outputs (e.g., water levels or dis-charge rates).

2.1 River Water Level/Streamflow Prediction Using Deep Learning Algorithm – Long-Short Term Memory (LSTM)

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN), a sub-set of artificial neural networks (ANN), introduced by Hochreiter et al. (1997). LSTM networks are specifically designed to overcome the challenge of long-term dependency in sequential data, where traditional recurrent neural networks (RNNs) struggle to effectively propagate information from earlier time steps as the interval between data points increases. An LSTM architecture comprises three primary components: an input layer, a memory cell, and an output layer. The memory cell is characterized by three gating mechanisms—the forget gate, input gate, and output gate—that regulate the cell state. The forget gate utilizes a sigmoid activation function to determine the degree to which information

from the previous time step should be preserved. The input gate adjusts the cell's state by updating weights associated with the current time step, informed by variables from the preceding time step. Lastly, the output gate computes the output for the current state by integrating information from both the previous output and the current input variables. The typical architecture of a LSTM model is illustrated in Figure 16.

The input data required for water level prediction using LSTM consists of independent variables and a dependent variable. The input features such as rainfall (X_1) are used as the independent variable ($X_1(t)$), while the output/target feature such as water level after one day or hour is used as the dependent variable ($Y_1(t + 1)$). This approach followed a supervised learning algorithm. A Python script (version 3.10.12) was developed to convert the raw input data into a format compatible with the supervised learning algorithm implemented in the LSTM model utilized in this study. The input data were further transformed into a three-dimensional array (*x, y, z*) to comply with the LSTM cell's expected input structure, a process accomplished using Python's reshape function. In the context of machine learning, sequence length denotes the length of the time window used in training the time-series data.

For example, with a sequence length of 3 and a daily time step, the model input consists of data at the current time step *X(t)*, one day prior X(t − 1), and two days prior *X(t − 2)*, with the model subsequently predicting the output for the following day *Y(t + 1)*. Table 2 details the model architecture. The forecast horizon can be modified by adjusting the lead time parameter. Consequently, the choice of sequence length and lead time for modeling and forecasting is informed by cross-correlation analysis results between rainfall and water level time series. Leveraging cross-correlation analysis to guide the choice of sequence length can help strike a balance between model performance and training efficiency, potentially reducing computation time without compromising prediction accuracy.

The collected dataset was partitioned into two distinct phases: training and testing. This separation was crucial for objectively evaluating the performance of the LSTM

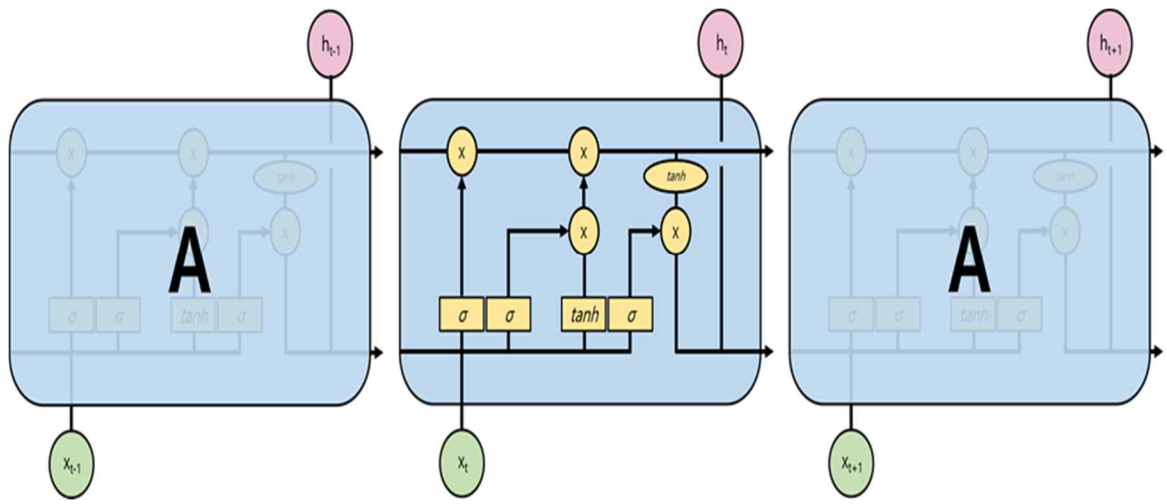

Fig 16. Typical schematic of LSTM (Hochreiter et al., 1997)

models, as it ensured that model assessment was conducted using data that were not involved in the training process. The trained model was then assessed for potential under-fitting or overfitting by conducting simulations on both the training and testing datasets. It is common to allocate 70% of the data for training the model, while the remaining 30% are reserved for testing purposes.

Table 2. Configuration of input and output formats for LSTM model implementation

Input Format (X)	Output Format (Y)
Input features recorded at time *t*	Output features predicted at time *t+n*
X_t	*Y(t+n)*

2.1.1 Cross-correlation between input and output features

The relationship between two time series representing different variables over a specified period can be analyzed using the statistical analysis of cross-correlation. This method quantifies the degree of correlation between the two time series at various lag intervals, yielding values within the range of −1 to 1. A correlation coefficient close to +1 indicates a strong positive relationship, whereas a value near −1 signifies a strong negative or inverse correlation. Cross-correlation analysis was applied to evaluate the temporal relationship between the input and output, as defined by Equations (10)–(12), which are expressed as:

$$CC_{xy}(k) = \frac{C_{xy}(k)}{C_{xy}(0)} \quad \text{(Eq.10)}$$

$$C_{xy}(k) = \frac{1}{N-k-1}\sum_{i=1}^{N-k}(x_i - \overline{x})\,(y_{i+k} - \overline{y}) \quad \text{(Eq.11)}$$

$$C_{xy}(0) = \frac{1}{N-k-1}\sum_{i=1}^{N}(x_i - \overline{x})\,(y_i - \overline{y}) \quad \text{(Eq.12)}$$

In this context, *CCxy (k)* denotes the cross-correlation coefficient at a lag of *k*, where *x* and *y* represent the two time series under analysis, where *x* usually represents the inputs while *y* indicates the outputs. The terms $\bar{x}, \bar{y}$ refer to the arithmetic means of the respective time series, *N* is the total number of observations, and *Cxy* denotes the cross-variance between the two series. A graphical representation of the cross-correlation coefficients computed across various lag times is referred to as a cross-correlogram. This analysis is important during the selection of sequence length for training the LSTM model as adopting a sequence length which is shorter than the lagged period with the highest correlation coefficients may result in improper learning of the flood forecasting model The cross-correlation coefficients can be computed using the *ccf* unction from the *tsa.stattools* module within the Python *statsmodels* library.

2.1.2 Optimization Algorithms for LSTM Modeling

LSTM model relies on hyper-parameters to drive its data-driven modeling process. The performance of such models is highly sensitive to the chosen hyper-parameters, making their optimization essential. To address this, three algorithms—Grid Search, Random Search, and Bayesian Search—have been widely used to identify the most effective hyper-parameter combinations for simulation using a LSTM model.

In Grid Search, a finite set of discrete values is specified for each desired hyper-parameter, and the optimal combination is identified by exhaustively cross-validating all possible configurations. The grid search

method is the easiest to implement and under-stand, but sadly not efficient when the number of parameters is large and not strongly restricted under H_0. Let Ω be the space of nuisance parameters $v = (v_1, v_2, \ldots v_m)$ over which we maximize the p-value. A simple way to setup a grid search consists in defining a vector of lower bounds $a = (a_1, a_2, \ldots, a_m)$ and a vector of upper bounds $b = (b_1, b_2, \ldots, b_m)$ for each component of v. Grid search involves taking n equally spaced points in each interval of the form $[a_i, b_i]$ including a_i and b_i. This creates a total of nm possible grid points to check. Finally, once each pair of points is calculated, the maximum of these values is chosen. The problem with this type of method is that the number of evaluations increases exponentially as n and m increase. Since we cannot really reduce m, decreasing n is the only possible way of assuring that the method stops in a reasonable time, but this decreases the validity of the solution.

In contrast, Random Search defines a discrete or continuous distribution for each hyper-parameter and randomly samples potential combinations from the joint distribution. As a result, Random Search is generally more efficient than the exhaustive Grid Search. Random Search is the simplest stochastic method for global optimization, as it generates a sequence of independent and identically distributed points in the feasible region S while keeping track of the best point that is found. The sequence of points converges to a global optimum with probability one, the probability that a point in S_i is reached within the first N iterations, is equal to:

$$Pr_i = 1 - (1 - \varphi(S_i))^N \qquad \text{(Eq.13)}$$

Where φ denotes the distribution on S. Random Search relies solely on the random sampling of a sequence of points in the feasible region of the problem. Therefore, it is applicable to a wide class of problems and often preferable for problems whose mathematical structure is difficult to analyze.

Bayesian Search, like Random Search, samples hyper-parameters from the search space. However, it continually updates the search space throughout the process based on the results of prior evaluations. In other words, it intelligently explores the space of possible hyper-parameter combinations by selecting the next configuration to evaluate based on previous observations. It applied Bayesian theorem to identify the optimum hyper-parameters in a more efficient way than just randomly searching all of the possibilities. In fact, Bayesian Search has been adopted to find various lost sea vessels such as the USS Scorpion. Apart from that, it also was used to help in recovering flight recorders in the case of Air France Flight 447 and to attempt to locate the remains of Malaysia Airlines MS370.

Suppose a combination of hyper-parameters has a probability p as the optimum solution, and the probability of successfully identifying the optimum solution is q. If a combination is searched and no optimum solution is found, then, according to the Bayesian theorem, the revised probability of that combination (p') is given as:

$$p' = \frac{p(1-q)}{(1-p)+p(1-q)} = \frac{p(1-q)}{1-pq} < p \qquad \text{(Eq.14)}$$

The readers are advised to refer to Caudle (2010) for the detailed procedures of optimal solution searching algorithms using Bayesian updates.

These optimization algorithms were implemented in Python and integrated into the LSTM model architecture as described in the previous section. Five hyper-parameters—neuron units, learning rate, dropout rate, batch size, and number of epochs—were tuned using the specified optimization methods. The process of locating the optimal hyper-parameters using the aforementioned optimization algorithms is illustrated in Figure 17. Assuming two

hyper-parameters, h_1 and h_2, a total of 24 combinations were required to be optimized. All possible combinations of these hyper-parameters are represented as black crosses in Figure 17. In Grid Search, all 24 configurations within the search space were exhaustively cross-validated according to the specified criteria, and the most optimal combination, X_o, closest to the ideal solution (red circle), was identified from these validated configurations. In contrast, Random Search identifies the optimal combination from a specified number of configurations sampled randomly within the search space. For instance, if instructed to optimize 24 combinations, Random Search randomly samples one combination in each search, and the optimal solution is selected from the 24 sampled configurations. Bayesian Search, on the other hand, intelligently explores the space of potential hyper-parameter choices by using the results of previous searches to decide which combination to evaluate next. As illustrated, the black crosses in Bayesian Search are concentrated around the area where the ideal solution is located. Once a combination close to the ideal solution is identified, Bayesian Search continues to refine its search in that region, iteratively improving the results until the predefined number of searches is completed.

2.1.3 Evaluation of LSTM Model

The Root Mean Square Error (RMSE), Nash–Sutcliffe Efficiency (NSE), and Mean Absolute Error (MAE) can be used as performance metrics to assess the accuracy of the model's predictions. RMSE measures the error between predicted and observed values, with its range extending from 0 to infinity. A smaller RMSE value, closer to 0, indicates smaller errors and predictions that are more closely aligned with observations.

Conversely, NSE is a standard metric extensively utilized to quantify the predictive accuracy of hydrological modeling approaches. An NSE value closer to 1 signifies smaller discrepancies between observations and predictions. Conversely, a negative NSE value indicates that the model's predictions are inefficient, performing worse than simply using the mean of the observed data as the predictor.

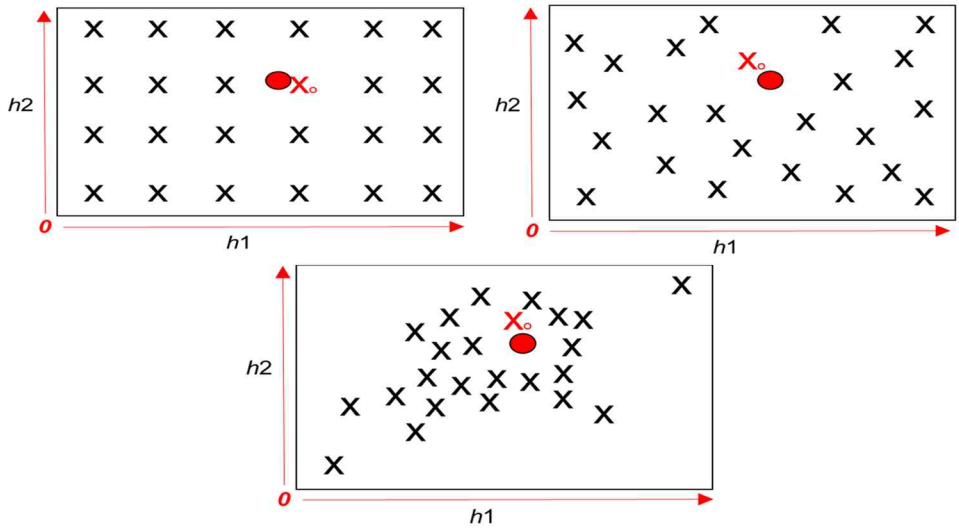

Fig 17. Process of locating optimum hyper-parameters using optimization algorithms: (upper left) Grid Search, (upper right) Random Search, (bottom) Bayesian Search (Kim et al., 2025)

MAE simply measures the average absolute difference between the predicted and observed values.

The mathematical formulations of each performance metric are presented in Equations (15), (16), and (17), respectively:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(O_i - P_i)^2}{n}} \qquad \text{(Eq.15)}$$

$$\mathrm{NSE} = 1 - \frac{\sum_{i=1}^{n}(O_i - P_i)^2}{\sum_{i=1}^{n}(O_i - O')^2} \qquad \text{(Eq.16)}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|O_i - P_i| \qquad \text{(Eq.17)}$$

Where O_i represents the predicted output and P_i indicates the observed output at time step i, O' is defined as the mean of the observed output, and n is the total number of observations in the dataset.

2.2 Demonstration on Flood Forecasting System Using AI Techniques

A PC desktop version of flood forecasting system using deep learning LSTM model based on Python language has been successfully developed and it is currently under final checking and tuning before distribution to the target TC member countries for applications in real-world flood forecasting. This flood forecasting system consists of 4 major modules, which are:

Module 1: Data Preprocessing

Module 2: Hyper-parameter Optimization Using Bayesian Search

Module 3: LSTM Model's Training and Testing

Module 4: Flood Forecasting Simulation

The user interfaces and functions for each module are illustrated and elaborated respectively in the following section.

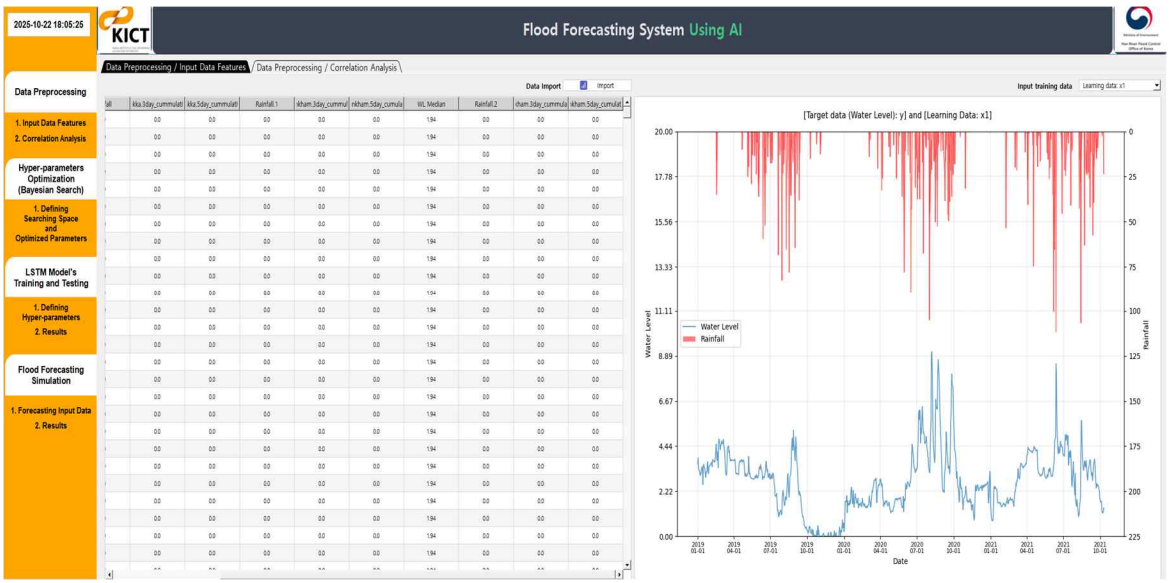

Fig 18. (Module 1) Data Preprocessing – Input Data Features

(Module 1) Data Preprocessing – Input Data Features

1) The system is designed to allow the user to insert up to 10 input features (rainfall or temperatures etc.) and one output feature (water level or streamflow) by clicking button (Import).

2) The input and output features' data should be stored in file of *csv*. format where the first column represents the output feature (target data, *Y*) while the remaining columns indicate the input features (training data, *X*).

3) A graph will be automatically generated by plotting the target data in the primary axis against the learning data in the secondary axis.

4) The user is allowed to choose which training data to be displayed in the graph by clicking the button (Input training data) and select the desired one from the list (if the inserted input feature is more than one).

(Module 1) Data Preprocessing – Correlation Analysis

1) The system is also featured with a function of cross-correlation analysis for studying the correlation relationship between the selected input features and target variable in previous/lagged time steps.

2) This analysis is important during the selection of sequence length for training the LSTM model as adopting a sequence length which is shorter than the lagged period with the highest correlation coefficients may result in improper learning of the flood forecasting model.

3) The Table shown in the left summarizes the cross-correlation coefficients obtained between the target data at present time step (lag=0) and the training data in each lagged period (lag=0,1,...)

4) The user is also allowed to choose which correlation coefficients of training data to be displayed in the graph by clicking the button (learning data) and select the desired one from the list (if the inserted input feature is more than one).

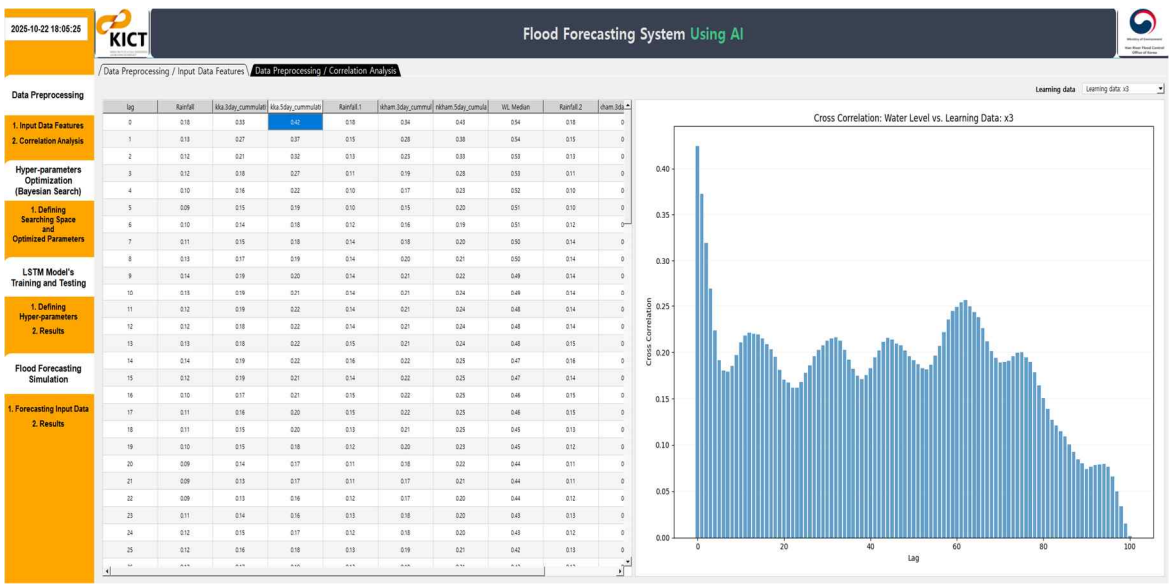

Fig 19. (Module 1) Data Preprocessing – Correlation Analysis

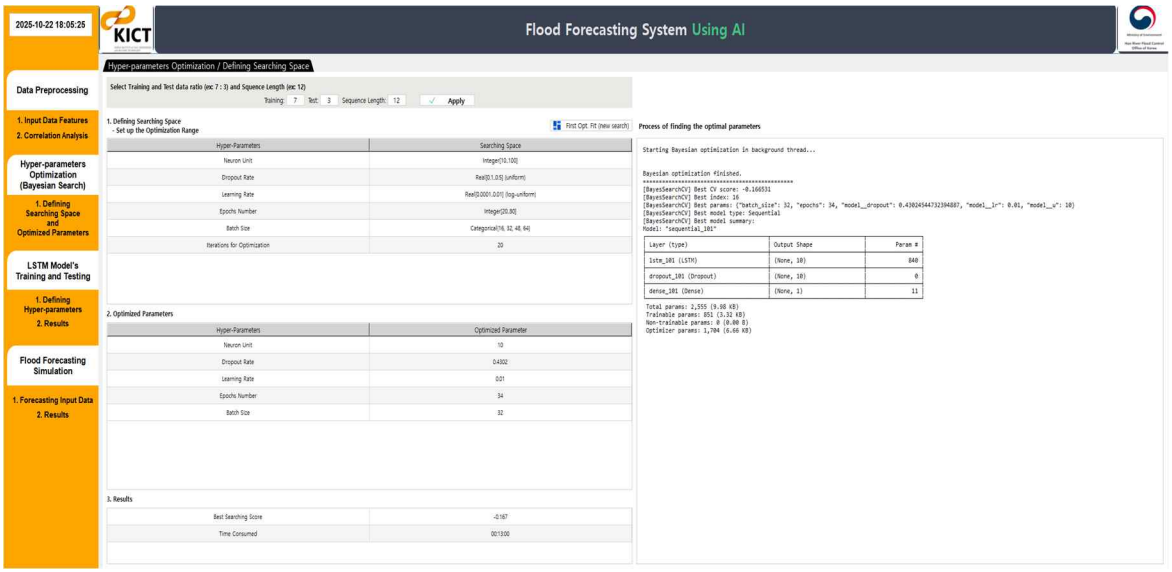

Fig 20. (Module 2) Hyper-parameters Optimization Using Bayesian Search – Defining Search Space and Optimized Parameters

(Module 2) Hyper-parameters Optimization Using Bayesian Search – Defining Searching Space and Optimized Parameters

1) The system is featured with an optimization module based on Bayesian Search technique for automatically locating the optimized hyper-parameters: *neuron units, dropout rate, learning rate, epoch size, and batch size*.

2) The user is required to specify the length for training and testing as well as the sequence length to initial the searching process by clicking the button (Apply) after specifying the desired values.

3) For example, 7 should be inserted for training if 70% of the inserted data will be used for training purpose and 3 for testing if the remaining 30% are intended for testing purpose. 10 should be key in for sequence length when learning data at 10 previous time steps will be used to predict the target data at present time step in each iteration.

4) The Table of "*Defining Searching Space*" summarizes the range of values specified for each hyper-parameter during the optimization searching process.

5) The best searching score and time consumed for optimization are displayed over the interface as well for user's reference.

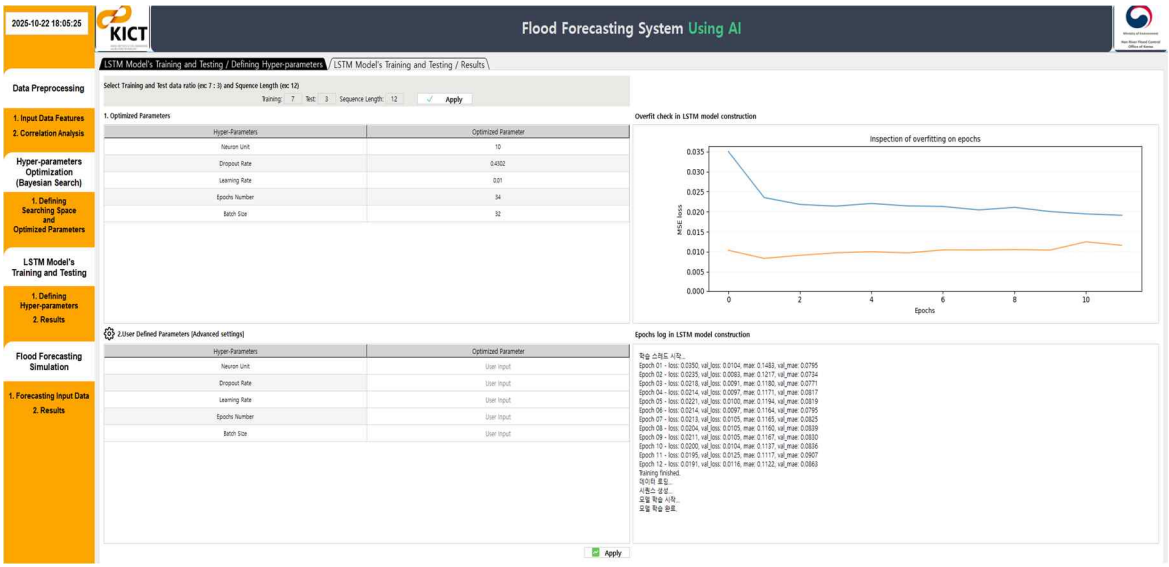

Fig 21. (Module 3) LSTM Model's Training & Testing – Defining Hyper-parameters

(Module 3) LSTM Model's Training & Testing – Defining Hyper-parameters

1) The optimized hyper-parameters obtained in previous module will be adopted automatically in this module to train and test the LSTM forecasting model.

2) The user can also specify their own desired hyper-parameters values for customized training and testing its own forecasting model.

3) The graph of loss curves during training and testing periods of trained model are portrayed in the right of the interface.

4) Convergence in the loss curves between training and testing may signify a proper learning while divergence may indicates over or under-fitting.

(Module 3) LSTM Model's Training & Testing – Training & Testing Results

1) The performances of the trained and tested forecasting models can be evaluated through the performance metrics of:

i) Root Mean Square Error (RMSE)

ii) Mean Absolute Error (MAE)

iii) Nash-Sutcliffe model Efficiency coefficient (NSE)

iv) Coefficient of determination (R^2)

2) The user is also allowed to choose which learning data to be displayed in the secondary axis (in red line) of the graph by clicking the button (Prediction Results in Training/Testing Period) and select the desired one from the list (if the inserted input feature is more than one).

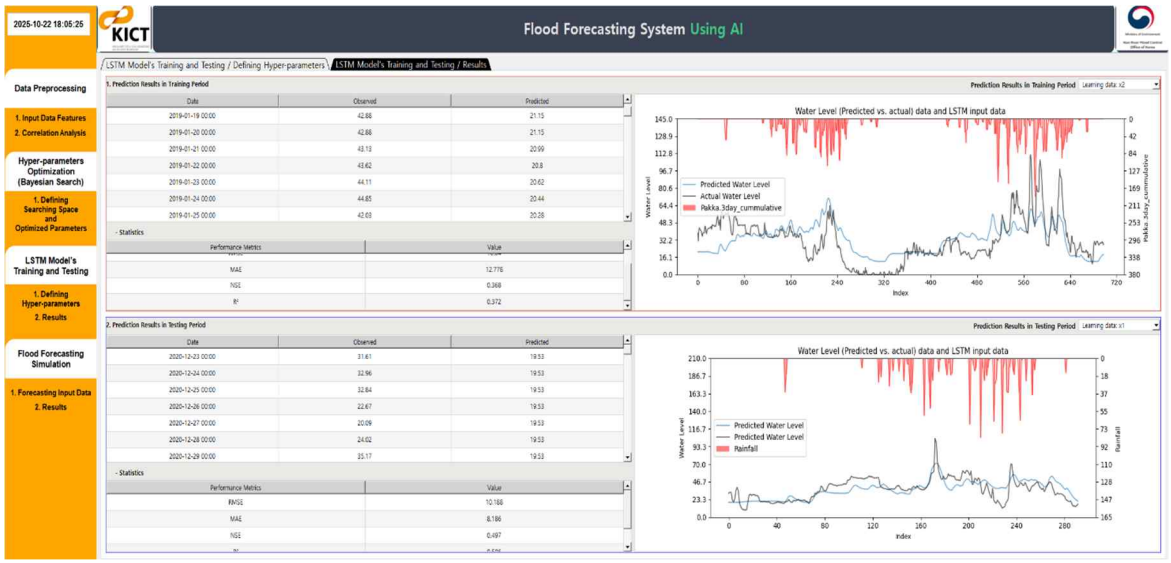

Fig 22. (Module 3) LSTM Model's Training & Testing – Training & Testing Results

(Module 4) Flood Forecasting Simulation – Forecasting Input Data

1) The trained and tested model then can be used to forecast future water level or streamflow data by feeding the flood forecasting model with the future input features' data by clicking the button (Import) for "Forecasting Data Input".

2) The forecasting input data should also be stored in file of *csv*. format where they should follow the similar format used to train the forecasting model.

3) It is essential to ensure that the length of the input data should be at least or longer than the sequence length specified during the model's training.

4) The target data to be predicted in future should be filled up with any numeric number while preparing the forecasting input data in file of *csv*. format.

5) The trained and tested forecasting model can be saved for future prediction or forecasting purpose by clicking the button (Forecasting AI Model Download) in the form of *Keras*. file.

Fig 23. (Module 4) Flood Forecasting Simulation – Forecasting Input Data

3. REFERENCES

Caudle, K. Searching Algorithm Using Bayesian Updates. J. Comput. Math. Sci. Teach. 2010, 29, 19–29.

Chen, T.Q.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. Pages 785 – 794. https://doi.org/10.1145/2939672.2939785

Guo, Z.; Leitão, J.-P.; Simões, N.-E.; Moosavi, V. Data-driven flood emulation: Speeding up urban prediction by deep convolutional neural network. J. Flood Risk Manag. 2021, 14, e12684. https://doi.org/10.1111/jfr3.12684.

Guglielmo, G.; Montessori, A.; Tucny, J.-M.; La Rocca, M.; Prestininzi, P. A priori physical information to aid generalization capabilities of neural networks for hydraulic modeling. Front. Complex. Syst. 2025, 2, 1508091. https://doi.org/10.3389/fcpxs.2024.1508091

Halicki, M.; Niedzielski, T. A new approach for hydrograph data interpolation and outlier removal for vector autoregressive modelling: a case study from the Odra/Oder River. Stochastic Environmental Research and Risk Assessment. 2024. 38:2781–2796. https://doi.org/10.1007/s00477-024-02711-5

Hochreiter, S.; Urgen Schmidhuber, J. Long Short-Term Memory. Neural Comput. 1997, 9, 1735–1780.

Kim, C.-S.; Kok, K.-H.;Kim, C.-R. Quasi-Optimized LSTM Approach for River Water Level Forecasting. *Water* **2025**, 17, 2087. https://doi.org/10.3390/w17142087

Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation forest. In: 2008 Eighth IEEE international conference on data mining. IEEE 2008, pp 413-422.

Digital Printed in Macao, China. Feb. 2026

ISBN 978-99981-833-4-6

Secretariat of ESCAP/WMO Typhoon Committee
Avenida 5 de Outubro, Coloane
Macao, China
Tel.: (+853) 88010531
Fax: (+853) 88010530
E-mail: info@typhooncommittee.org

ISBN 978-99981-833-4-6

ESCAP/WMO TC/TC-No.0024