# Real-Time Outlier Detection in Time Series Data of Water Sensors*

L. van de Wiel[1,2][0000−0002−5884−4367], D. M. van Es[2][0000−0002−7067−6394], and A. J. Feelders[1][0000−0003−4525−1949]

[1] Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands
[2] Ynformed, Stadsplateau 4, 3521 AZ Utrecht, The Netherlands

**Abstract.** Dutch water authorities are responsible for, among others, the management of water levels in waterways. To perform their task properly, it is important that data is of high quality. We compare several univariate and multivariate methods for real time outlier detection in time series data of water sensors from Dutch water authority "Aa en Maas". Their performance is assessed by measuring how well they detect simulated spike, jump and drift outliers. This approach allowed us to uncover the outlier parameter values (i.e. drift or jump magnitude) at which certain detection thresholds are reached. The experiments show that the outliers are best detected by multivariate (as opposed to univariate) models, and that a multi-layer perceptron quantile regression (QR-MLP) model is best able to capture these multivariate relations. In addition to simulated outliers, the QR-MLP model is able to detect real outliers as well. Moreover, specific rules for each outlier category are not needed. In sum, QR-MLP models are well-suited to detect outliers without supervision.

**Keywords:** Outlier Detection · Time Series · Quantile Regression · Synthetic Evaluation · Machine Learning.

## 1 Introduction

Data validation is an important issue for water authorities in the Netherlands. These regional government bodies are responsible for, among others, sewage treatment, dyke management and the management of water levels in waterways. It has been shown that validation pipelines along with implementation advice result in more reliable policy advice, improved operational management and enhanced assessment of current management practices[18]. We examine water data from *Waterschap Aa en Maas*, one of the 21 water authorities in the Netherlands.

To improve data quality, we try to separate outliers from 'real' data points. Our focus is on real-time outlier detection in time series of water sensor measurements. The sensor data consists of time series with fixed intervals between measurements. Different sensors can output time series that are correlated with

* Made possible by Ynformed and Waterschap Aa en Maas

each other. Here, we can use time series from one or multiple sensors to predict other sensor values. If a big difference between the predicted and observed value occurs, the value may be classified as an outlier[1]. It is important that outliers are detected in real-time, as it enables taking immediate action to resolve possible issues, such as misbehaving sensors or a change in the sensor environment.

Our research focuses on finding which methods can be applied to detect outliers in an unlabelled, unvalidated data set of multivariate time series in a real-time setting. The data is unvalidated; it is raw sensor data that has not gone through any processing steps to improve quality. The data is generally also unlabelled, which means that domain experts have not indicated whether outliers occur. An exception to this is in a few time series that we used for analysis.

We compare different regression-based methods, that predict sensor values given (1) only the sensors history ('univariate'), or (2) given only measurements of other sensors ('multivariate'). Outliers are then determined when the observed data deviates too much from the predicted value. The univariate approach is simpler and can be implemented more easily in practice. Yet, this method runs the risk of carrying past outliers (such as drift) into the future. This would then correctly predict outlying sensor behaviour, thereby failing to label it as outlying. We expect the multivariate approach to solve this problem, as it is not informed about the potentially outlying target history.

## 2 Data Overview

The data from Aa en Maas comprises water height data in weirs, with a measurement frequency of 15 minutes. At these weirs, we have access to water height on the upper part and the lower part of the weir, and also to water flow rate and weir shutter height. We used water heights on the *upper* part of the weirs for the analysis. These time series (which are the exceptions described in Section 1) were designated by domain experts as not containing any outliers. We gathered all data between 05-06-2015 and 01-07-2019.

An example of water height data on the upper part of a specific set of weirs is shown in Figure 1[3]. We see that this data is not without errors. For example, the "108HOL_upper" time series (bottom line) has a strange swing around October 2018. Some other minor spikes can also be encountered in this same series. Furthermore, missing values can occur, as seen near the end of the "108IJZ_upper" time series. This data set has relatively few missing values (approximately 2500); other sensor sets have more.

The data of Figure 1 is of relatively high quality. However, when looking at other sensor sets, time series seem more noisy. In addition, other sensors had more missing data. To evaluate outlier detection capabilities under varying data quality conditions, we selected multiple sensors that had varying data qualities. We chose four weirs for the analysis and model evaluation. These weirs are 102BFS, 103HOE, 104OYE and 201D. For each target time series, we used four other time series as features in multivariate modelling, see Section 3.1.

[3] This data is not used in our main experiments.

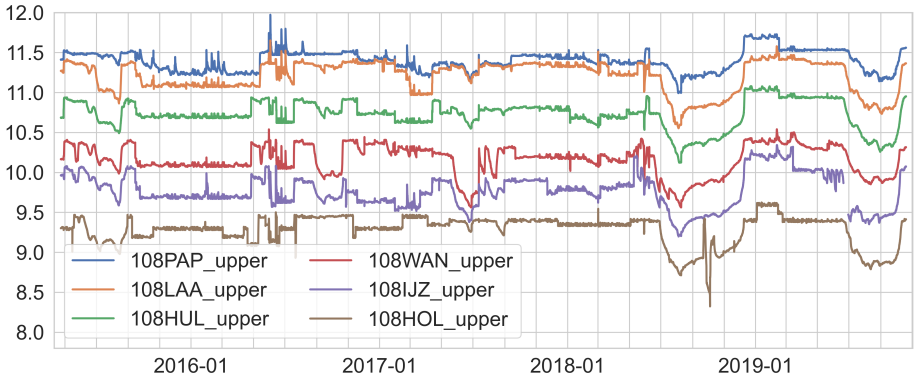



**Fig. 1.** An example of available water height data. These six sensors are all on the same body of water and are relatively close to each other (about 6km as biggest distance).

## 3 Experiment setup

### 3.1 Outlier detection pipeline

**Sensor selection** We first discarded all sensors that had more than 10% missing values. Then, for each target sensor, from the hundreds of sensors we selected four sensors that correlated most with the target and used those as predictors. This ensures decent model performance while reducing the danger of overfitting.

**Imputation** The data set contained missing values, which can be the result of sensor network issues or sensor malfunctioning, for example. We imputed rather than discarded these values to ensure evenly spaced time series. To determine the best suited imputation method for our problem, we benchmarked various methods. For this, we simulated gaps (of similar duration distributions compared to the actual missing gaps) in the time series and measured how well a MICE procedure[17] with different estimators (extra trees, linear regression, Bayesian ridge, KNN, random forests and MTSDI[8]) was able to reproduce the missing values. This showed that the linear regression estimator worked best.

**Feature engineering** For most multivariate experiments, we used rolling lag, min, max and mean features with time steps of 15 and 30 minutes and 1, 2, 4, 8 and 16 hours. In the univariate setting, only the mean values over a prolonged period of time turned out to be useful. For most univariate models we used the mean values of window sizes [64 hours, 128 hours, ..., 1048 hours]. The used features per model type are described in Table 1.

**Feature scaling** To stabilise and enhance model training, we scaled all features to unit variance and zero mean.

**Table 1.** Features per model type. Models are described in Section 4. The five multivariate models are linear regression (LR), MLP, Perceptron (P), QRF and RNN. The five univariate models are LR, MLP, P, AR and isolation forests (IF).

| Algorithm | LR | MLP | P | QRF | RNN | LR | MLP | P | AR | IF |
|---|---|---|---|---|---|---|---|---|---|---|
| Uni-(U)/Multivariate(M) | M | M | M | M | M | U | U | U | U | U |
| Use feature engineering | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ |
| Use raw lag values | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| Use target sensor itself | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Use correlated sensors | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |

**Modelling and predicting (ab)normal behaviour** The models used can be divided into two categories. The first one is regression-based models, where a prediction for a target variable is made (possibly accompanied by quantile values). We can compare this against the actual value and then calculate residuals.

The other category is direct classification. This approach looks at data and then directly determines whether it is an outlier or not.

**Outlier classification** Most of our regression-based models use quantile regression. To perform outlier detection when using these models, we applied the Western Electric rules[19]. We applied Rule 1 and a variation of Rule 2. Rule 1 indicates a single point that falls outside of the $3\sigma$-limit as outlying. Rule 2 does this if two out of three successive points fall beyond the $2\sigma$-limit.

The original Rule 2 led to a high number of false positives. Our improved approach was to look at predictions averaged over the span of a day, and check whether this exceeds the averaged values of the 2nd quantile. Minor short-lived errors now get smoothed out and we get a more accurate way of describing a gradual change. This is described in Algorithm 1. The first three lines downsample the target time series and the upper and lower 2nd quantile (which were outputted by the quantile regression model) from a frequency of 15 minutes to daily data. The next line performs the detection: if the downsampled time series is above the upper limit, or below the lower limit, an outlier is classified. Eventually, this data is upsampled to a frequency of 15 minutes and returned.

**Algorithm 1** Drift detection by downsampling.

1: **function** DRIFT_DETECTION($y_{in}, q2_{upper}, q2_{lower}$)
2: $y_{daily} \leftarrow$ DOWNSAMPLE_TO_DAY($y_{in}$)
3: $q2_{upper} \leftarrow$ DOWNSAMPLE_TO_DAY($q2_{upper}$)
4: $q2_{lower} \leftarrow$ DOWNSAMPLE_TO_DAY($q2_{lower}$)
5: $outliers\_daily \leftarrow (y_{daily} > q2_{upper}) \cup (y_{daily} < q2_{lower})$
6: **return** UPSAMPLE_TO_15MIN($outliers\_daily$) ▷ Outliers per 15 minutes.

### 3.2 Synthetic evaluation

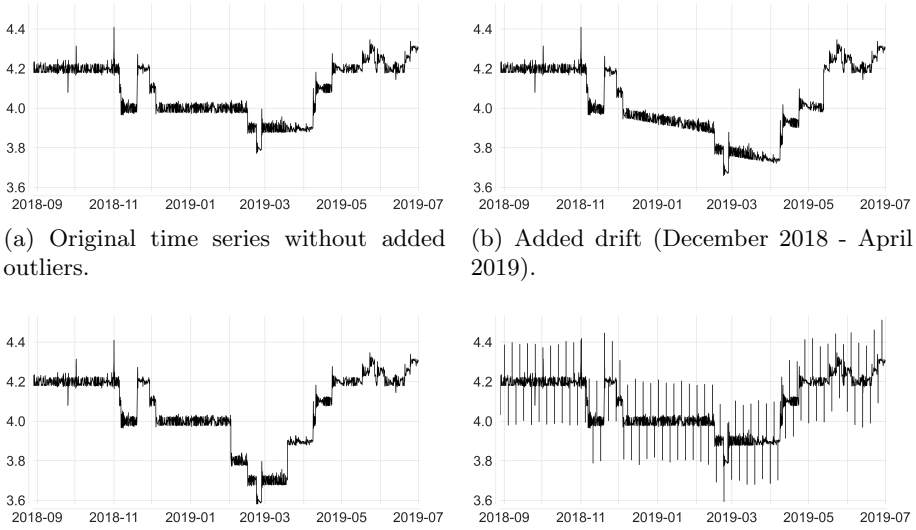
Correctly recognising outliers is crucial. Therefore, we decided to use the $F_\beta$-score with $\beta$=2 to evaluate performance.

Experts of the water authority established that no outliers are present in the test data of the four time series. This is beneficial for the synthetic evaluation, as already present outliers might interfere with the ones we introduce. The synthetic evaluation method entails that we altered the data to simulate outliers that might happen in reality. Such a method has been applied before in the literature[14].

We studied common outlier definitions to get an idea for outlier categories in water time series data [10][18]. We focused on three synthetic outlier categories because they were regarded to be important by the domain experts:

- **Jumps:** A period of data which is increased or decreased by a constant value. After the period has ended, the data values return back to the original range.
- **Extreme values:** Isolated data points which are increased or decreased by a constant value.
- **Linear drift:** The occurrence of a series which has a gradual linear trend upwards or downwards.

To perform synthetic evaluation for jumps and linear drift, we created multiple test series with different outliers in it. We used one specific drift or jump and then moved this outlier throughout the data, with each movement yielding a new series. We alternated between outliers oriented upwards and downwards. For each test case, we created 100 of these series. We used multiple outlier generation values (in meters), which were 0.02, 0.05, 0.1, 0.2 and 0.3. Jumps had the duration of approximately 1.5 months, whereas drifts lasted for approximately 6 months. Examples are shown in Figure 2.




(a) Original time series without added outliers.

(b) Added drift (December 2018 - April 2019).

(c) Added jump (February 2019 - middle of March 2019).

(d) Added extremes.

**Fig. 2.** Outlier examples in sensor 104OYE for outlier value 0.2. For jump and drift, this is 1 of the 100 created series.

## 4 Modelling and hyper-parameter tuning

In this section we give a short description of the models and algorithms used in this study, and how we tuned their hyper-parameter settings. Input features were used as described in Table 1. The data set is divided into training (60% of the data), validation (20%) and testing (20%) sets. The training data is used to fit different models which are described in this section. The validation data is used to perform hyper-parameter tuning. The test data is used to assess model performance. Model evaluation is performed through a synthetic outlier approach (see Section 3.2). According to the 60% - 20% - 20% split, this means that the training data is in the range of 05-06-2015 - 13-11-2017, the validation data ends at 06-09-2018 and the testing data ends at 01-07-2019.

An important distinction is between univariate models that only use a sensor's own history to predict future values, and multivariate models that use the values of other sensors. An advantage of the univariate approach is that it is always applicable as no other time series are needed. Furthermore, large sudden changes in values might be easy to track. If we use a multivariate approach, we ignore a sensor's own history and base the detection on other sensor time series. We do this to prevent consistently predicting the same value as currently present (working like a persistence model), which will fail to detect drift. An advantage of the multivariate approach is that we can detect (gradual) changes which happen in only one sensor. If a sensor is slowly drifting, for example, a multivariate approach could detect this based on data from other sensors, whereas a univariate approach may be unable to detect this successfully.

### 4.1 Regression-based models

**Autoregressive (AR) models** AR models[2] are often used in practice for time-series modelling, but are applied in outlier detection as well[10]. We implemented AR models for univariate modelling.

We experimented with different lags used for prediction. Depending on the target sensor, a minimum number of 3-5 lags was needed before the model stabilised. Further lags had little influence, so 5 lags were picked. If a predicted value deviated too much from the observed value, it was classified as an outlier. Based on experiments, we selected a threshold of 4 cm. Lower values gave us too many false alarms, with higher values we missed too many genuine outliers.

Water levels are not changing a lot in successive measurements. So, the models learn coefficients which favour predicting a similar value as the current one.

**Linear regression** We used linear regression with Lasso penalty[16]. We used the validation set to determine the ideal value for Lasso penalty $\lambda$. Validation loss for $\lambda$-values in [3.0, 1.0, 0.3, ..., 0.001, 0.0003] was reported. It is useful to use a $\lambda$-value which scores well, but also is relatively large. Based on the 1SE rule and the modelling ability of promising sensors, we decided to use a $\lambda$-value of 0.03 throughout the experiments.

An issue when using linear regression for outlier detection is in defining the outlier detection threshold. We opted for a quantile approach so we could use the

same classification rules as in Section 3.1. To calculate the quantiles, we added or subtracted the standard deviation of the target time series multiplied by a scalar value to the prediction. This is shown in Equation 1.

$$q_i = \hat{y} \pm \frac{i}{2}\hat{\sigma}(y_{train}) \quad | \quad i \in \{1, 2, 3\} \tag{1}$$

A disadvantage is that this will lead to a fixed quantile width for the whole model. Varying quantile width is desirable, as uncertainty about the predictions can differ throughout the data.

**Quantile regression forests (QRF)** Parameter values of the QRF algorithm [13] were based on experiments. We used 1000 different trees in total. For each tree, we used the same parameter settings: A node needs to have at least 40 samples in it for it to be considered for a split, a resulting leaf node must have at least 20 samples and the maximum number of considered features per split is one third of the total number of features.

### 4.2 Neural network-based approaches

To perform neural network architecture tuning systematically, we used the Hyperband algorithm[11]. In the multivariate experiments we averaged the predictions of 10 different neural networks. This ensemble approach is chosen as random weight initialisation has a sizeable effect on the model. In the univariate experiments the ensemble size is lowered to 5, to keep running times acceptable. Many extra predictions are needed because the input of the testing data changes for each outlier time series, which was not the case in the multivariate modelling.

**Quantile regression: Multi layer perceptron** The quantile regression multi layer perceptron (QR-MLP) model is a neural network with hidden layers that only uses dense layers. Multiple output nodes are used to calculate values for different quantiles. We use the pinball loss function[9][15] where all the quantiles are taken into account. In the algorithm runs, we used early stopping with a patience value of 5 and a mini-batch size of 128.

The Hyperband algorithm used 5 executions per trial, 3 Hyperband iterations, a factor of 3 and max epochs of 30. In the end, 270 trials were run. It selected the number of layers (1, 2, 4 or 8), number of neurons per layer (16, 32, 64, 128 or 256), dropout (0.0, 0.1, 0.2, 0.3 or 0.4) and learning rate of the network (0.005, 0.001, 0.0005, 0.0001, 0.00005 or 0.00001).

It was not possible to find one general network architecture that works in all cases. There seems to be some correlation between the validation loss and the network complexity. For example, sensor 104OYE can be modelled relatively well and only uses one layer. On the other end of the spectrum we see 102BFS (which was selected to test the impact of its low correlation with other sensors), which has high validation loss and needs more complex models. We decided to use a different architecture for each sensor. Due to computational and time restraints, we were not able to optimise different numbers of neurons per layer. The resulting architectures are shown in Table 2.

**Table 2.** QR-MLP model architectures per sensor.

| Sensor | Dropout | Learning rate | Number of layers | Neurons per layer | Average validation loss of final model |
|---|---|---|---|---|---|
| 104OYE | 0.4 | 0.0005 | 1 | 128 | 0.1820 |
| 103HOE | 0.0 | 0.005 | 1 | 256 | 0.8790 |
| 201D | 0.4 | 0.005 | 2 | 128 | 0.9580 |
| 102BFS | 0.4 | 0.00005 | 8 | 128 | 1.5330 |

**Quantile regression: Perceptron model** A baseline neural network model in the form of a QR-perceptron model was created. This network has no hidden layers. It is somewhat similar to the linear regression model, but like in the QR-MLP model, we use the pinball loss function with multiple output nodes. We thus still have varying quantile width. The only hyper-parameter that needs to be tuned is the learning rate. An exhaustive grid search is now possible. Experiments showed that a relatively large learning rate of 0.005 works best for this kind of model. This value was used for all the QR-perceptron models.

**Quantile regression: RNNs** These networks used RNN layers instead of dense layers. We let the tuner decide if a GRU[3] or LSTM[7] kind of RNN layer should be used. For speed, we now use at most 4 layers, a batch size of 2048 and a window size of 32. A difference with the other multivariate approaches, is that since we have a RNN, all these 32 values are used in every step. Also, this disallows us from explicitly modelling features like the minimum and mean features.

Again, there did not seem to be a best overall network architecture. Moreover, it seems that neither LSTM- nor GRU-layers work best for every network. We use a different network architecture per sensor, as shown in Table 3.

**Table 3.** RNN model architectures per sensor

| Sensor | RNN type | Dropout | Learning rate | Number of layers | Neurons per layer | Average validation loss of final model |
|---|---|---|---|---|---|---|
| 104OYE | GRU | 0.1 | 0.0005 | 1 | 256 | 0.2060 |
| 103HOE | LSTM | 0.0 | 0.005 | 1 | 256 | 0.9401 |
| 201D | LSTM | 0.4 | 0.001 | 4 | 64 | 1.3292 |
| 102BFS | LSTM | 0.2 | 0.0001 | 4 | 64 | 1.2330 |

### 4.3 Direct classification model: Isolation Forests (IF)

The isolation forest model[12] is often applied in the literature[5]. When using multivariate feature sets, we can only look at outliers of a whole system (like a group of 5 sensors), instead of at outliers of a single sensor. Also, it is mandatory

to incorporate the history of the target sensor. Since we want to know if a specific sensor is behaving strangely, this method is only suited for our univariate setting.

We have performed hyper-parameter tuning to determine the ideal value of the contamination parameter. If we set the contamination value too low, we will detect few outliers. If it is set too high, the precision of our model will drop. Our experiments suggested a value of 0.07.

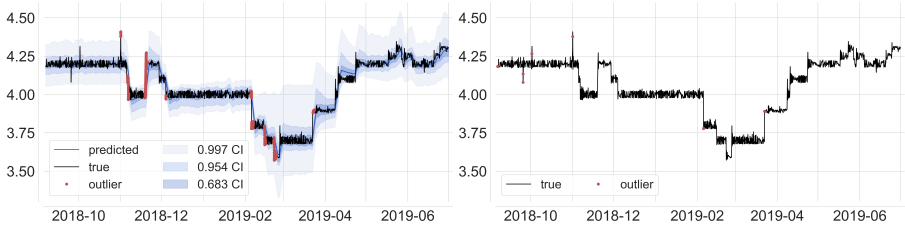
## 5 Results

In this section, we first describe illustrative examples for the univariate and multivariate models. Then, we compare these models. We end this section with a description of the practical impact of the best performing ones.

### 5.1 Illustrative examples: Univariate results

The QR-RNN and QRF models have not been applied to the univariate modelling experiments, since this became prohibitively slow. We applied the linear regression, QR-MLP, QR-Perceptron, AR and IF models here. We show the visual results of one specific time series of sensor 104OYE in Figure 3. We added a jump of 0.2m from February 2019 to the middle of March 2019.

**Results QR-MLP** For the univariate QR-MLP models, we used the same architecture for each sensor, as we do not have to take into account correlated time series. This was the same architecture that was used for 104OYE in multivariate QR-MLP modelling, as described in Table 2. Figure 3a shows that some parts of the added jump can be detected, but this is certainly not the case for the sequence of outliers as a whole.




(a) Using the univariate QR-MLP model. (b) Using the AR model.

**Fig. 3.** Plots of sensor 104OYE, with added jump.

**Results AR** AR model performance is shown in Figure 3b. We see here that the begin and end points of the added outlier sequence can be detected. The period in between can not be detected, though. Also, some other sudden changes in the time series have been classified as outliers.

### 5.2 Illustrative example: Multivariate results QR-MLP

We compared the multivariate linear regression, QR-MLP, QR-Perceptron, QRF and QR-RNN models. Figure 4 shows the results of the same 104OYE time series with added jump, now modelled multivariately by QR-MLP. This jump is detected well, but some false positives are also present. Some outliers may have been missed by the domain experts. This is most visible around November 2018. Results of drifts detection for all outlier values are shown in Figure 5.

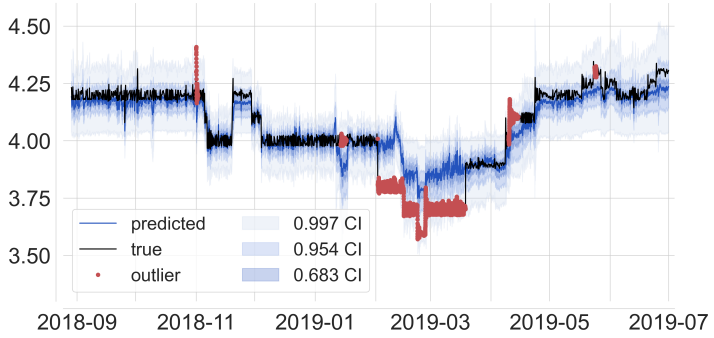



**Fig. 4.** Quantile plot of sensor 104OYE, with added jump using the QR-MLP model.

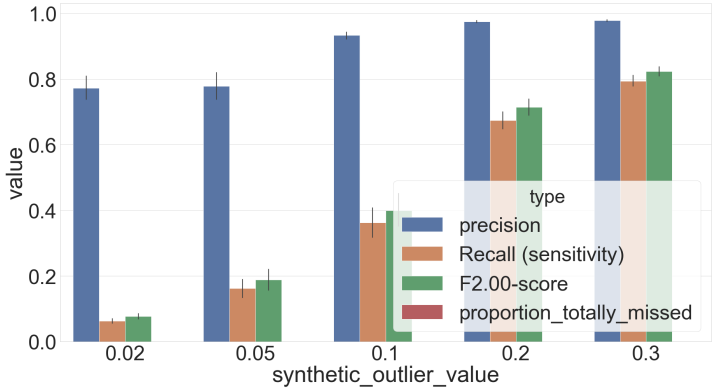
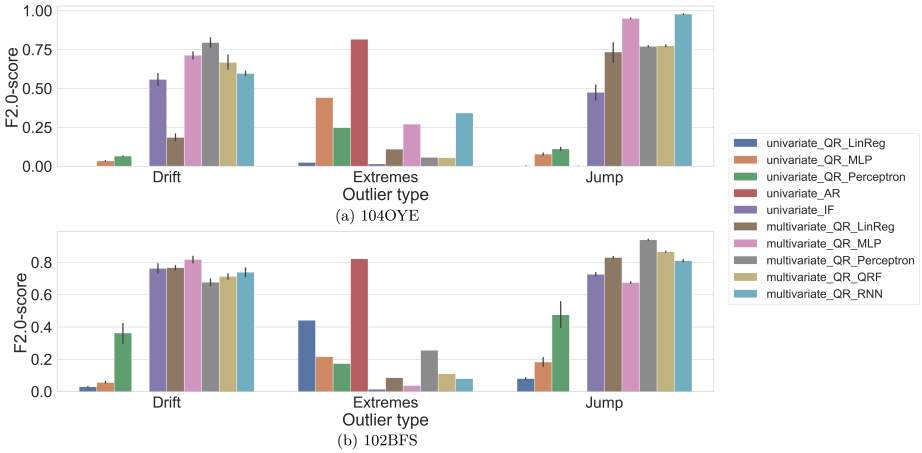



**Fig. 5.** Bar plots of QR-MLP model scores of sensor 104OYE for all outliers values for drifts. The legend shows the *proportion totally missed*, which indicates the proportion of the outlier sequences missed completely. This value is 0 for each outlier value.

### 5.3 Comparison of univariate and multivariate modelling techniques

The $F_2$ scores of all models are shown in Figure 6. This figure shows how well different multivariate and univariate models score on different outlier categories for different sensors. We see some clear differences. To compare the performance of these different models, we followed a two-step approach[4]. First, a Friedman Aligned Ranks test was performed to check whether there are significant differences between the distributions of the results of the models[6]. If this test yielded a significant difference, we used the Nemenyi test to compare all models pairwise.




(a) 104OYE

(b) 102BFS

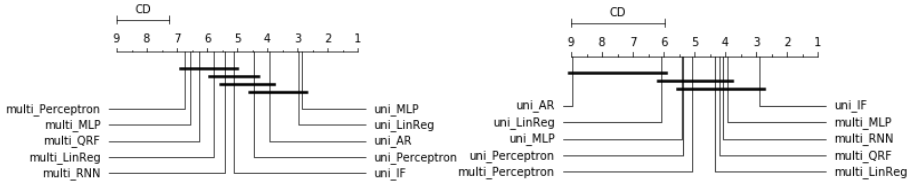
**Fig. 6.** Bar plots of model performance on all data sets, univariate and multivariate combined for outlier value 0.2.

We used the Friedman Aligned Ranks test in 8 scenarios. One of them consisted of all sensors and outlier categories and sizes. We also divided the data by outlier category and by sensor. All these distributions are significantly different from each other (see Table 4). This is expected, as we compare many models which have very different performances as is visible in Figure 6.

**Table 4.** Friedman Aligned Ranks results ($\alpha$=0.05) of all experiments.

| Sensors | All | All | All | All | 102BFS | 103HOE | 104OYE | 201D |
|---|---|---|---|---|---|---|---|---|
| Outliers | All | Drift | Jump | Extremes | All | All | All | All |
| $\chi^2$ | 118.936 | 100.000 | 115.084 | 60.752 | 38.274 | 36.949 | 39.365 | 51.748 |
| $p$-value | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Significant? | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |

We now perform the Nemenyi test to see which models have significantly different performance. The results are shown in Figure 7. On the horizontal axis, the average ranking of the algorithms is shown. The further a model is to the left on the x-axis, the better it is scoring on average. Algorithms that are connected by a bold line are not differing significantly from each other. In the overall comparison of Figure 7a, the five multivariate models outperform the five univariate ones. This is because they score better in drift and jump detection scenarios. If we want a single model to detect all outlier types, then the multivariate QR-MLP or QR-perceptron model seems the best choice. In the extreme outlier category (Figure 7b), however, AR seems to perform exceptionally well. This is due to the fact that AR almost works like a persistence model and can detect a large sudden change easily.




(a) All data. (b) All sensors, only extremes.

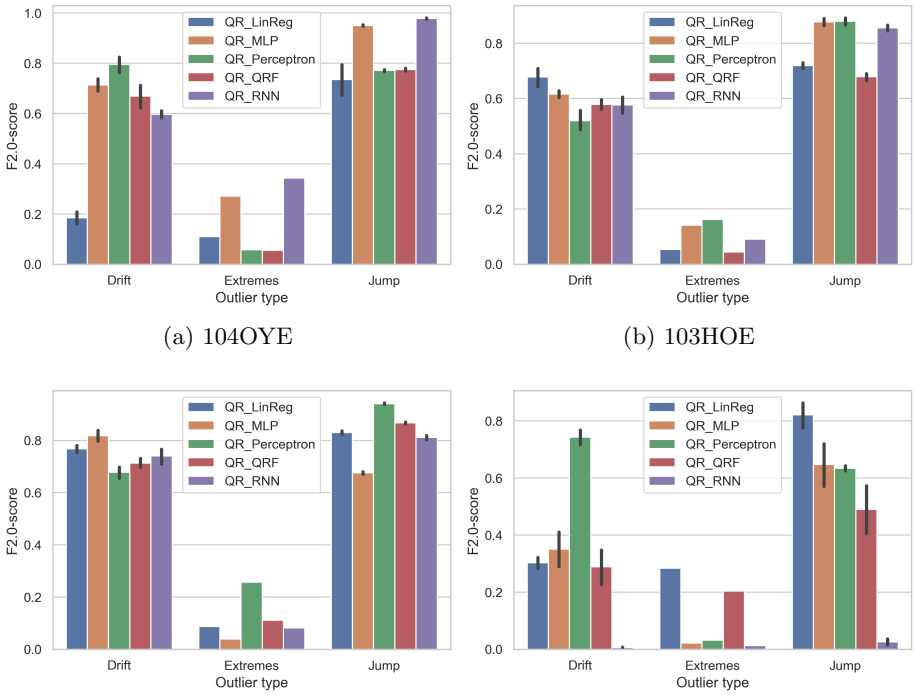
**Fig. 7.** Nemenyi test results for univariate and multivariate models combined.

### 5.4 Comparison of multivariate modelling techniques

As the multivariate models outperform the univariate ones in most cases, we zoom in further on the multivariate ones. Figure 8 gives an overview of the $F_2$-score results of multivariate models for outlier value 0.2m. Note that the $F_2$-score for extremes is low in all cases. This may be explained by the fact that fewer outliers are added here than in the other categories. In the drift category, 17520 outlier points are added. In the jump category this number is 4320 and when using extremes, only 100. The total number of true positives differs greatly per method, so the roughly constant number of false positives can severely impact precision and thus the $F_2$-score.

We see some big differences between the models, but we also note that model performance differs greatly per sensor. Results of the Friedman Aligned Ranks test are shown in Table 5.

In Figure 9, the Nemenyi test result for all data is shown. QR-MLP and QR-perceptron are significantly different from QR-RNN. Other comparisons showed similar results. Since QR-MLP and the QR-perceptron model perform decently most of the time, these could be go-to algorithms.




(a) 104OYE (b) 103HOE

(c) 102BFS (d) 201D

**Fig. 8.** Bar plots of model performance of multivariate models on all data sets for outlier value 0.2.

**Table 5.** Friedman Aligned Ranks test results ($\alpha$=0.05) in multivariate experiments.

| Sensors | All | All | All | All | 102BFS | 103HOE | 104OYE | 201D |
|---|---|---|---|---|---|---|---|---|
| Outliers | All | Drift | Jump | Extremes | All | All | All | All |
| $\chi^2$ | 14.947 | 10.556 | 8.399 | 4.384 | 3.287 | 4.273 | 11.549 | 22.646 |
| $p$-value | 0.005 | 0.032 | 0.078 | 0.357 | 0.511 | 0.370 | 0.021 | 0.000 |
| Significant? | Yes | Yes | No | No | No | No | Yes | Yes |

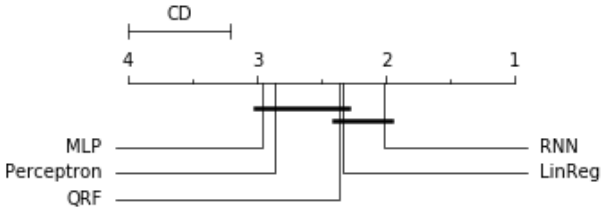



**Fig. 9.** Nemenyi test results for multivariate models.

### 5.5 Practical impact

An advantage of the multivariate QR-MLP model is that it generalises to many different kinds of outliers. In Figure 10, domain experts annotated the subsequence between middle September 2018 and middle October 2018 as outlying. This is detected nicely (indicated by the red dots).

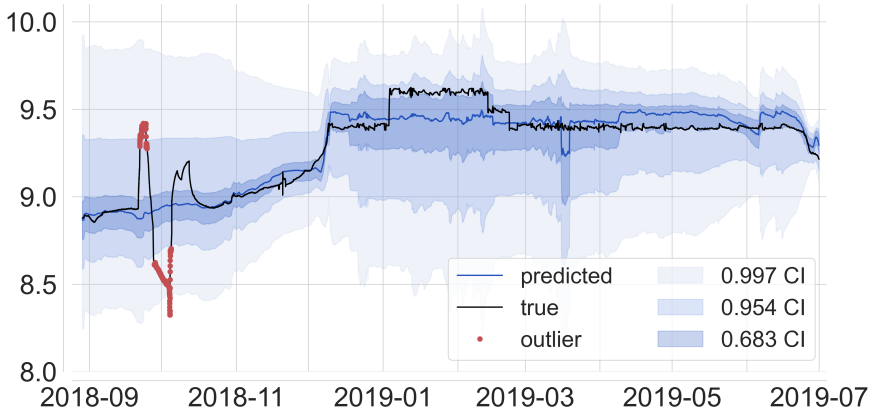



**Fig. 10.** Outlier detection plot of 108HOL modelled by the multivariate QR-MLP model. No synthetic outliers were added. Although the already present outlier around October 2018 can not be detected fully, it still can be detected well.

Domain experts stated that jump values and extreme values of 0.2m are reasonable in real life. We can derive from Figure 6 that the performance of the best performing models is satisfactory in many cases. The domain experts also stated that a drift is generally in between 0.05m-0.10m over the period of a year. This roughly corresponds to the two categories of outlier values 0.02m and 0.05m. Outliers for these values are harder to detect, as seen in Figure 5. However, this judgement is (too) harsh, as the algorithms do detect almost every drift sequence after some time. The experts stated that periodical checks for the occurrence of drift are normally performed yearly. Our models need one month on average to detect drift of value 0.05m, which leads to improvements over a manual periodical check.

## 6 Conclusion and future work

In this work, we applied multivariate and univariate real-time outlier detection models in unlabelled water height time series. Instead of only cleaning historical data, the trained models can be used to monitor sensor measurements and directly signal outlying values. The key contribution of this work is the systematic comparison of algorithms and the easily parametrisable synthetic validation scenarios which were constructed in cooperation with domain experts.

We showed that multivariate approaches work better than univariate approaches for jump and drift outlier types. For extreme values however, univariate approaches appear to outperform multivariate ones. Yet, we think that this result will not hold in practice, because it partly is an artefact of our synthetic evaluation procedure. Univariate outlier models basically function by signalling large instantaneous changes. This indeed highlights extreme values, but may fail to detect slightly more gradual ones. Also, natural (more gradual) jumps in the data will be missed. As we only added instantaneous extreme outliers, many true positives were present. Few other already present data points changed so quickly, so few false positives were present. Thus, this category of models performed well in our simulations. In real life however, extreme values occur less frequently, and natural jumps are more apparent. Therefore, these models are likely to result in inadequate performance when implemented in practice.

It should be noted that the multivariate modelling approach is not applicable for all sensors (like 102BFS). A multivariate approach is only suitable when sufficiently correlating series are available.

Within the category of multivariate models, we found that the QRF approach and the QR-RNN models performed poorly. The QRF model resulted in very jagged quantile boundaries, which resulted in the misclassification of many data points. The QR-RNN model often resulted in very wide quantiles, which worsened performance. Linear regression performed relatively decently, although the fixed quantile width remains an issue. In the end, we can conclude that the QR-MLP and the QR-perceptron models performed the best overall.

To get a better overview how well these different models work in practice, it is recommended that a pilot program is carried out to test the performance in a more practical setting.

Our extreme values scenario had some artefacts. Although we selected a realistic outlier value in cooperation with the domain experts, it is worthwhile to investigate more realistic scenarios. An example is a more gradual extreme value. This is fundamentally different from drift, as a gradual extreme value could occur in a few time steps, in contrast to a duration of multiple months. Research into a combination of different (extreme) outlier categories may also be useful.

It may be a fruitful idea to use different models to detect different outlier categories. For example, combining the results of an AR model and a multivariate QR-MLP model could work to detect extreme values, jumps, and drifts.

An interesting research subtopic regards determining outlier causes. An outlier can be caused by multiple factors. Different kinds of outliers might require different means of alleviation. It is of interest to determine these different causes with additional techniques.

Another noteworthy subtopic concerns propagating sensor errors. If a sensor malfunctions, this will not only affect its own predictions, but will affect all other sensor predictions that make use of the values of this malfunctioning sensor as a predictor variable as well. Further research is needed to make accurate claims about this phenomenon.

## References


1. Aggarwal, C.C.: Data mining: the textbook. Springer (2015)
2. Chatfield, C.: The analysis of time series: an introduction. Chapman and Hall/CRC (2003)
3. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research **7**(Jan), 1–30 (2006)
5. Ding, Z., Fei, M.: An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proceedings Volumes **46**(20), 12–17 (2013)
6. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences **180**(10), 2044–2064 (2010)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
8. Junger, W., de Leon, A.P.: mtsdi: Multivariate time series data imputation. https://cran.r-project.org/web/packages/mtsdi/index.html (2012), R package version 0.3.5.
9. Koenker, R., Hallock, K.F.: Quantile regression. Journal of economic perspectives **15**(4), 143–156 (2001)
10. Leigh, C., Alsibai, O., Hyndman, R.J., Kandanaarachchi, S., King, O.C., McGree, J.M., Neelamraju, C., Strauss, J., Talagala, P.D., Turner, R.D., et al.: A framework for automated anomaly detection in high frequency water-quality data from in situ sensors. Science of The Total Environment **664**, 885–898 (2019)
11. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. The Journal of Machine Learning Research **18**(1), 6765–6816 (2017)
12. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 413–422. IEEE (2008)
13. Meinshausen, N.: Quantile regression forests. Journal of Machine Learning Research **7**(Jun), 983–999 (2006)
14. Perelman, L., Arad, J., Housh, M., Ostfeld, A.: Event detection in water distribution systems from multivariate water quality time series. Environmental science & technology **46**(15), 8212–8219 (2012)
15. Rodrigues, F., Pereira, F.C.: Beyond expectation: Deep joint mean and quantile regression for spatio-temporal problems. arXiv preprint arXiv:1808.08798 (2018)
16. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) **58**(1), 267–288 (1996)
17. Van Buuren, S., Groothuis-Oudshoorn, K.: MICE: Multivariate imputation by chained equations in R. Journal of statistical software pp. 1–68 (2010)
18. Versteeg, R., de Graaff, B.: Valdidatieplan Waterkwantiteitsmetingen (in Dutch). STOWA 2009-20 (2009)
19. Western Electric Company: Statistical quality control handbook. Western Electric Company (1956)