Machine Learning for Quality Assurance of Lutz Catchment Runoff Data

Final Report

Gillian McGinnis

ADVISORS:
Sriram Iyengar, PhD (*The University of Arizona College of Medicine, Phoenix*),
Steven Paton, MSc (*Smithsonian Tropical Research Institute*)

December 10, 2025

Contents

List of Figures

List of Tables

List of Acronyms

AP	average precision
AUC PR	area under the precision-recall curve
BCI	Barro Colorado Island
FN	false negative(s)
FP	false positive(s)
iid	independently & identically distributed
LSTM	Long Short-Term Memory
OOF	out-of-fold
ROC AUC	receiver operating characteristic area under the curve
STRI	Smithsonian Tropical Research Institute
TN	true negative(s)
TP	true positive(s)
VFP	Visual FoxPro
XGBoost	Extreme Gradient Boosting

Abstract

The long-term goal of this project is to create models that can automate quality assurance of temporal runoff data from the Lutz catchment of Barro Colorado Island on the Republic of Panama. This will remove the need for manual data corrections, which not only use the valuable time and energy of researchers, but also have the potential to be imprecise or inconsistent. Rainfall and soil moisture measurement data were integrated to produce a more well-informed model. The optimized Extreme Gradient Boosting (XGBoost) model for classifying the presence of obstructions in runoff data achieved a recall of 0.838 and precision of 0.206 after threshold optimization, indicating high ability to flag actual blockage events with the tradeoff of flagging many non-problematic data points. Future work will continue to improve on the obstruction model, as well as create detection models for other failure modes and conduct quality assurance on the runoff values accordingly.

1 Introduction

1.1 Background

Runoff data has been continually collected from the Lutz weir (Figure 1) since 1972, making it one of the longest, continually monitored micro-catchment datasets for the neotropics.[1] The weir is located in the Lutz catchment, a 9.73 ha area located in 100-year-old, secondary low land tropical rainforest on Barro Colorado Island (BCI), a 15 km^2 island located in Lake Gatun in the Republic of Panama (Figure 2).[1] BCI is operated by the Smithsonian Tropical Research Institute (STRI), a branch of the Smithsonian Institution.[2]

At first, hand-written records (which have since been digitized) recorded runoff in intervals ranging from three minutes to three hours, but since mid-1989, runoff values have been electronically measured every five minutes.[3] The time-series dataset contains millions of data points, with more added as the sensor continues to record new measurements. In addition, rainfall measurements have been recorded at the Clearing station 170 m north of the weir. Soil moisture measurements have been taken weekly from 10 sampling sites within the catchment at two depths each (Figure 3).[3]

Quality issues in the runoff data come about due to environmental factors (such as partial blockages of the outflow channel), sensor failures, and calibration issues, to name a few. Currently, quality assurance is conducted manually using a custom written program in Visual FoxPro (VFP), a data management program from the 2000s. This program permits the visual inspection and correction of the data, a critical feature for manual quality assurance of micro-catchment data. However, not only does this approach run the risk of imprecise or inconsistent corrections, but the deprecated status of VFP leaves it vulnerable to future inaccessibility as it may eventually no longer be supported. Because quality assurance requires some subjective decisions by the individual analyzing the data using the VFP system, the results are potentially biased to some, unknown degree.[1] Past attempts at automated quality assurance have failed due to the failure modes' resilience against traditional stochastic methods.

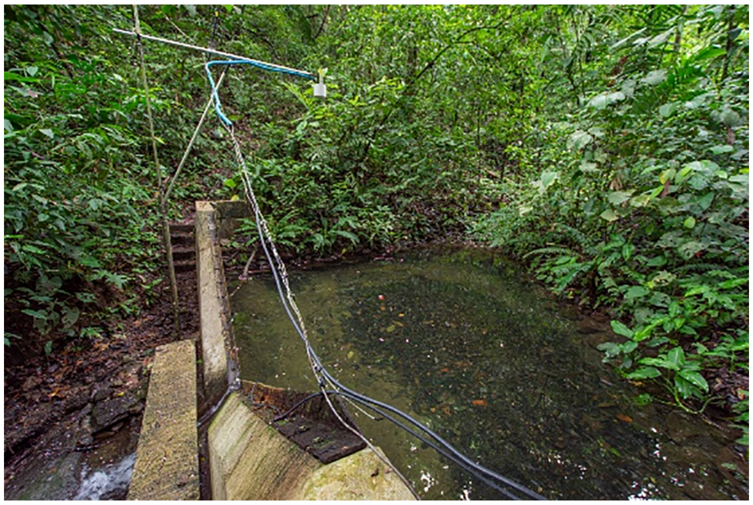

Figure 1: A photograph of the Lutz weir, including the electronic sensor measuring runoff.[4]

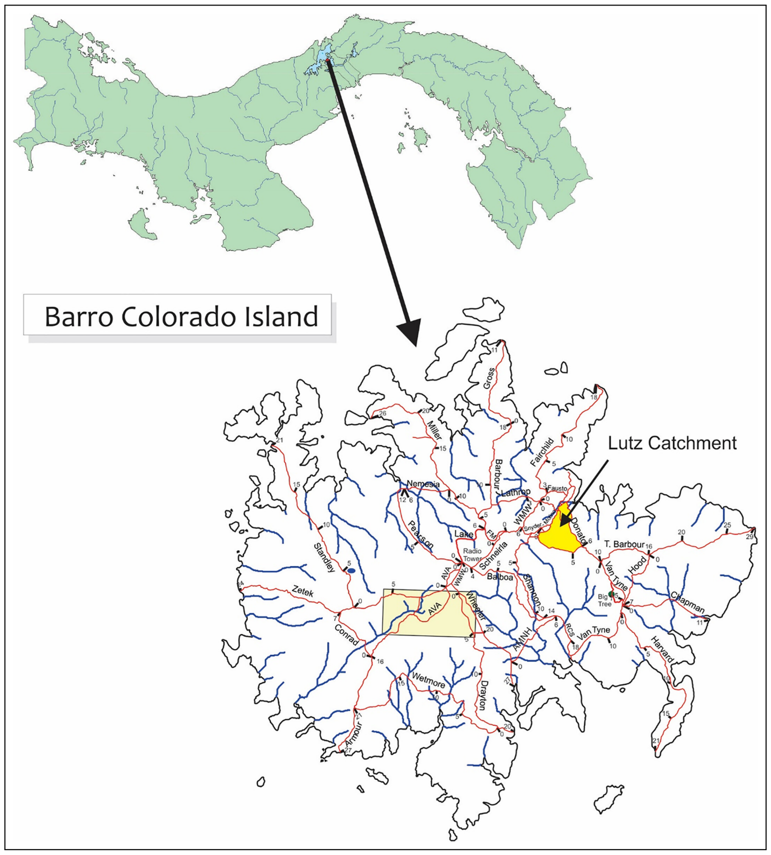

Figure 2: The location of the Lutz catchment on Barro Colorado Island (BCI) in the Republic of Panama.[4]

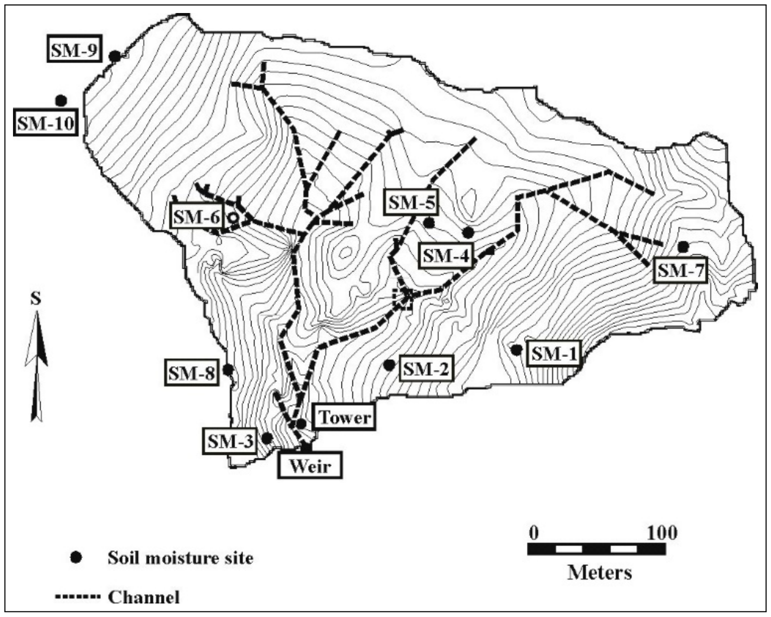

Figure 3: A fine-scale topographic map of the Lutz catchment showing the locations of the weir, tower, and 10 soil moisture sampling sites.[3]

1.2 Failure Modes

Periods of time in the data are flagged as containing “failure modes”, as outlined in Table 1. Some failures occur in extremely short intervals of time (even as few as one reading), while others can spoil data spanning multiple weeks. One long-term goal of the project is to create systems to determine where there are readings containing a particular failure mode. Examples of each failure mode can be found in Appendix A.

Failure mode	Description	Complicating factors	Example
Obstruction	Debris blocks the weir’s ‘V’, resulting in gradual increases in water level, and later abrupt decreases following blockage removal.	Rainfall events before, during, or shortly after a blockage can interrupt the base flow and decay curves.	Figure A1
Spike	Short and abrupt changes in level typically caused by equipment issues.	Extremely short-term issue that could be skipped over by random sampling.	Figure A2
Calibration	Standard checks that require baseline correction or slight data pivot.	Recovery after blockage clearing can render calibration points ineffective.	Figure A3
Sub-zero	The stream runs dry, or the pond is being drained for cleaning.	Rain during a pond draining can render new data unrecoverable.	Figure A4
Signal noise	Equipment failure results in impossible variability of reported values.	Impossible to manually fix & resists standard de-noising.	Figure A5

Table 1: Overview of weir failure modes.[3]

1.3 Model Selection

Machine learning has been used in hydrology for applications such as predictive runoff modelling[5] and modelling agricultural drought.[6] In the realm of quality assurance, rainfall runoff data collected by citizen scientists in Ethiopia has been visually (via graphical inspections) and quantitatively (via statistical methods) analyzed for quality compared to nearby professional references.[7]

Initially, a multi-class classification model was considered for flagging points containing failure modes. However, not all failure modes require all available variables in order to be successfully flagged, as shown in Table 2. Thus, individual models can be constructed based on failure mode and input variables in order to minimize the complexity of each model.

	Variable			
Failure mode	Raw runoff	Rainfall	Calibration	Soil moisture
Obstruction	✓	✓	✓	✓
Spike	✓			
Calibration	✓		✓	
Sub-zero	✓	✓		
Signal noise	✓			

Table 2: Variables that relate to or otherwise complicate each failure mode.

Obstructions are the most difficult failure mode to detect and correct (in-part because they require the greatest number of variable considerations) and thus were selected as the first failure mode to construct a model for. External variables of rainfall and soil moisture content are necessary for a well-informed model. Additionally, the calibration variable can have mixed success, since an obstruction or weir cleaning can render it useless. In the context of modelling expected droughts, Houmma et al.[6] highlights how incorporation of local and microclimate variables are becoming increasingly necessary due to climate change and "exceptional situations" which otherwise make extreme peaks of variables & interdependencies no longer able to be statistically modellable.

Despite time-series data not being independently & identically distributed (*iid*), statistical time-series problems can still use binary classification.[8] In hydrology, for example, Hudnurkar and Rayavarapu[9] applied binary classification to rainfall time series, categorizing "rain" or "not rain".

Unsupervised anomaly detection (such as isolation forest) would be useful in flagging any unobserved or new failure modes, however given the existing number of explicitly identified failure modes, it is unnecessary for the primary goal. Support vector machines were also ruled out because they are better for small-to-medium preprocessed regular data, and do not work well for time-series sequences.[10]

In the field of remote sensing and geospatial analysis, recurrent neural networks—especially Long Short-Term Memory (LSTM)—are successful for analyzing multi-temporal data, however scalability is limited and they can be prone to overfitting.[11] Non-continuous nature of other variables may also prevent meaningful pattern detection, meaning that it may struggle with the differing granularity of input variables for detecting obstructions. LSTM models are also sensitive to noisy data and may result in less accurate predictions.[12] Furthermore, LSTM are difficult to interpret, and would thus have resulted in a black box issue as to why a particular failure mode flag was triggered since feature importance is unclear.[13]

Boosting uses many weak learning models to sequentially improve its performance such that the final model is a strong learner.[14] Gradient tree boosting uses simple & shallow decision trees as weak learners, finding good terminal nodes using standard regression tree learning on the residuals, then resolving for the weights of each leaf by minimizing the loss function.[14] It is optimized for large datasets and is made even faster with parallel processing.[15] Extreme Gradient Boosting (XGBoost) adds a regularizer, utilizes a second-order loss approximation, can sample features internally, and scales better.[14,15] XGBoost has been used for time-series anomaly classification in quality assurance in the manufacturing sector.[16] Importantly, it does not require perfectly clean or consistent data—thus, it can handle missing values and noise.[15] However, it requires careful hyperparameter tuning and explicit feature inputs since it does not "learn" from previous temporal trends. Ultimately, XGBoost was selected as the first model to attempt for flagging instances of the obstruction failure mode.

1.4 Data Splitting

In environmental machine learning, improper missing data management techniques can worsen data leakage.[17] Unlike other applications of train/test splits, time-series data in this context should not be randomly split, and must instead remain continuous whenever possible. This is because time-series data is not *iid*, so random splits would cause data leakage. One consideration when making a train/test split is that real-world variables causing inconsistencies in the data or different concentrations of error instances make a simple split difficult define without careful pre-processing.

Time series analyses can use "window" splits for cross-validation, which involve training on a window of time, validating on an immediately proceeding window, and repeating as necessary.[18] This approach is beneficial for tuning hyperparameters, as it can indicate model stability over time and reflects real-world implementation.

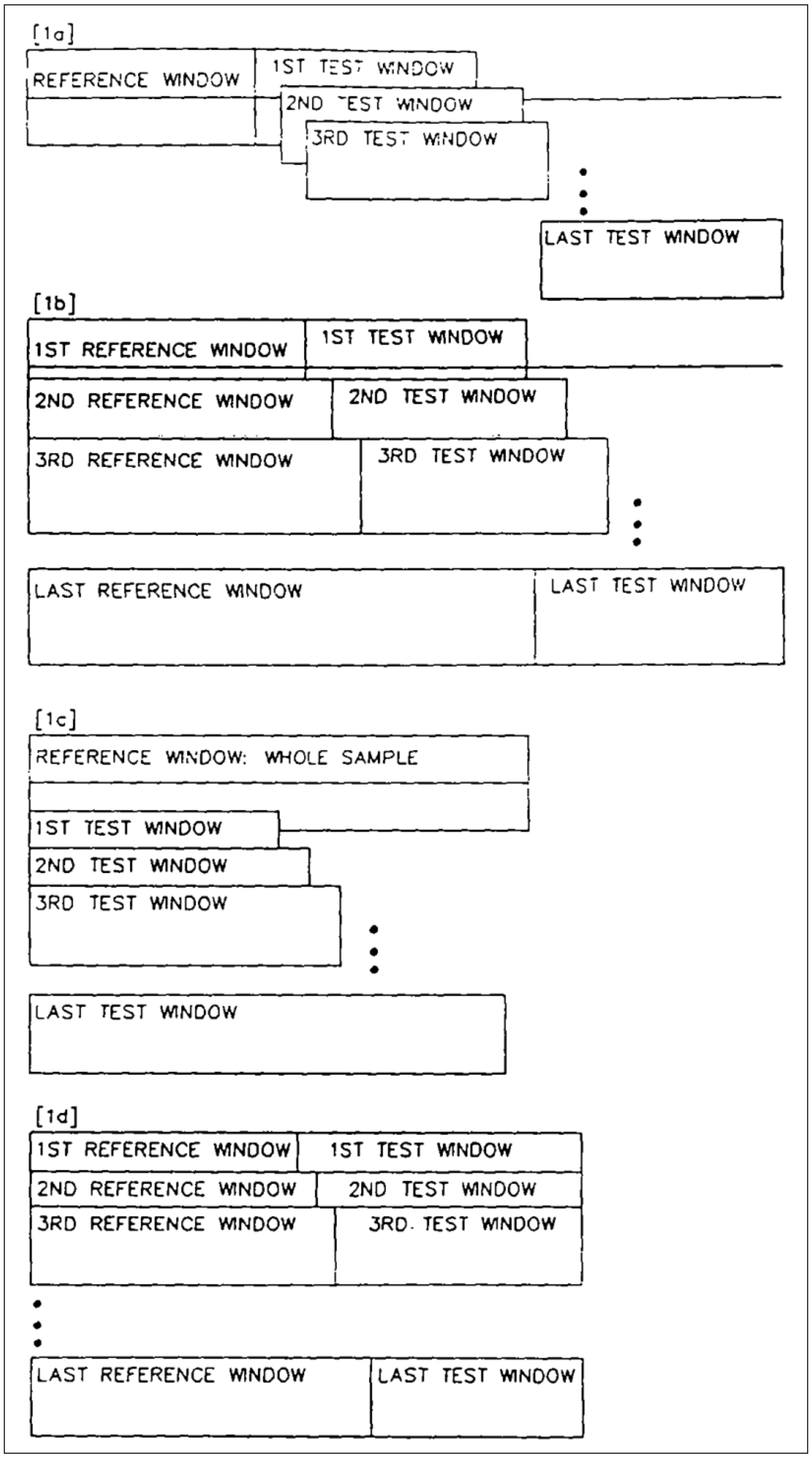

Figure 4: A comparison of different time-series segmentation techniques.[18]

2 Methods

Analyses were conducted in Python and compiled in Jupyter Notebooks. Annotated code, data files, and reports can be found in the project's GitHub repository.[i]

2.1 Data Preparation

2.1.1 Importing

All data sets are readily available in the form of `.csv` files, and are regularly updated on the Smithsonian Research Data Repository with `CC BY 4.0` licenses.[19] In their raw form, the data were stored in five individual data sets: runoff measurements (including raw and manually-adjusted values), rainfall measurements, weir calibration values, shallow-depth soil moisture measurements, and deep-depth soil moisture measurements.

2.1.2 Wrangling

In order to apply models, data first had to be uniformly coded, cleaned, and united into a cohesive set. Because electronic monitoring did not begin until mid-1989, values that relied on this digitization of manually-collected charts were first removed from the runoff datasets. Occasional sensor failures also resulted in modern records relying on these manual fallbacks, so those values were also removed. The largest of these failures was a window beginning in January 2013 when the sensor failed and there was no backup until a different one was installed in August 2014. Some entries were lacking in a data source flag, so those which were between chart-reliant values were also removed.

Soil data was not collected from late February through late December 2020 due to the COVID-19 pandemic. Because the obstruction failure mode detection depends on these values (due to differences in how they appear in the dry season versus the wet seasons), this window of time was also removed from the data prior to model creation.

Data points which were both marked as "missing" and had reported raw runoff values of -999.0 were also removed.[ii] In the manually-adjusted results, the levels of such points had been set to 0.0, but because there were so few instances of them in the filtered data set ($n = 12$), they were removed for obstruction analysis.

Soil moisture abnormalities There were some discrepancies in records between the shallow and deep soil moisture data sets. Although the "deep" set is for depths of 30-40 cm, some entries from 0-10 cm were included. Most of these values matched those in the "shallow" (1-10 cm) data based on timestamp and sample site number, however there were a few instances where the reported values differed. It was ultimately decided that those mismatched values ($n = 11$) could safely be eliminated from the "deep" set. Although this issue did not remove a significant number of entries when compared to the resulting totals ($n = 11,489$), it was an error that had not been found previously, and thus made a substantive contribution to the STRI dataset. Additionally, a few "shallow" samples with non-standard location numbers were removed ($n = 3$).[iii] Duplicated entries were also removed, as were those flagged as being "bad" or "doubtful" values.

[i] Project repository: https://github.com/gmcginnis/info-698-capstone.

[ii] This also differs from instances of gap fills.

[iii] Although it was possible that a minor miscoding caused this issue, it was not possible at the time of discovery to verify the solution because the primary data contact was unavailable for the majority of the project duration due to the federal government shutdown.

Failure mode flags The failure modes were originally encoded in singular string entries. Some data points have multiple failure modes, and some entries contained duplicated flags. For analysis, a series of Boolean variables were created to represent these. These consisted of: obstruction, spike, gap fill (often related to the "sub-zero" failure mode), weir cleaning (also related to sub-zero), spike, and calibration.

2.1.3 Feature engineering

Because XGBoost models use one entry of input features at a time, data had to be extended such that each entry also contained information of data before it. These representations came in the form of rolling statistics (e.g., mean rainfall over the past 3 hours) and lag features (a shift to a past data point, e.g., runoff 10 minutes prior). In the context of obstruction detection, rolling statistics included the sum and standard deviation of runoff values in windows spanning 10 min, 15 min, 20 min, 25 min, 30 min, 1 hr, 3 hr, 6 hr, 12 hr, and 24 hr prior to the data point, and rainfall values in windows spanning 10 min, 30 min, 1 hr, 3 hr, and 6 hr prior. Every other runoff value from 5 min through 3 hr prior were added as lag features, as were the lags for each aforementioned rain rolling sum feature from 15 min, 30 min, 1 hr, 2 hr, and 3 hr prior.

Temporal distances from events were also calculated and saved as features. This included minutes since a rainfall occurred, minutes since a calibration point was taken, and days since a soil moisture sample was taken. Change from the immediately preceding runoff measurement and rain measurement were also added as features, as was a feature which calculated the cumulative value for any rain "events" (e.g., the cumulative sum for a continuous rain lasting 30 min). A decay feature was also added for this cumulative rain feature, which duplicated the cumulative rain and continued the values forward weighted by an exponential decay value. The decay rate was set to -0.1, but would benefit from expert insight for a more precise value to better represent actual flow decay behavior.

Features to represent seasonality and time-of-day considerations were also added. These took the form of day of the year, month of the year, hour of the day, and minute of the hour, each transformed with sine and cosine to allow the model to be based on the cyclical patterns of time.[iv]

2.1.4 Train/test split

An initial 80/20 train/test split was conducted on the chronologically-ordered data. The sets were also separated into feature variables (X) and the target variable (y, in this case the obstruction flag).

A visual overview of this initial 80/20 split and methodology for splits in subsequent steps is provided in Figure 5.

2.1.5 Feature selection

To reduce high-dimensionality, avoid overfitting, and improve processing time & internal memory, highly-correlated features were removed. This was conducted by computing the pairwise Pearson correlation between all features in the X training set, and removing one of those in the pairs in which the calculated correlation was greater than 0.97.

[iv]For example, the original numeric value for the day of the year of December 31 (365) is distant from that of January 01 (001), but in reality they are very near.

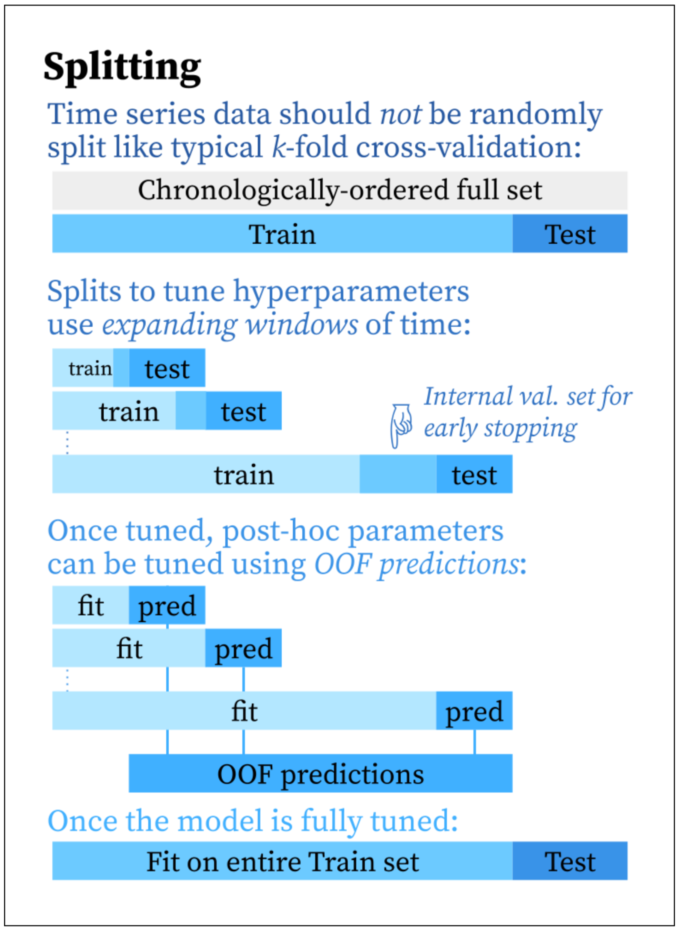
Splitting
Time series data should *not* be randomly split like typical *k*-fold cross-validation:
Chronologically-ordered full set
Train
Test
Splits to tune hyperparameters use *expanding windows* of time:
train
test
train
test
train
test
Internal val. set for early stopping
Once tuned, post-hoc parameters can be tuned using *OOF predictions*:
fit
pred
fit
pred
fit
pred
OOF predictions
Once the model is fully tuned:
Fit on entire Train set
Test

Figure 5: Visual representation of splitting techniques used in the analysis.

2.2 Modelling

2.2.1 Hyperparameter tuning

To tune XGBoost hyperparameters, an expanding window cross-validation (Figure 4.1b) approach was used. This is similar to how a model would act once deployed, as it would only gain more data over time. Because the data is mostly high-resolution, a larger window width is permissible.[20] Five splits on the training data were conducted, with the internal "training" set expanding with each fold (Figure 5).

In each of 50 iterations, a unique and randomly-selected combination of hyperparameters was selected from predefined distributions. The set was then evaluated across the five expanding-window splits, wherein the model was trained on the subset and used an internal validation set for early stopping to determine the optimal number of boosting rounds necessary before performance degrades or has diminishing returns. The performance metric for the individual rounds was the area under the precision-recall curve (AUC PR).[v] The specific set of hyperparameters and boosting rounds found to yield the highest AUC PR score across all cross-validation folds was then selected as the options for the final model.

[v]In this context, AUC PR is the better metric compared to receiver operating characteristic area under the curve (ROC AUC) since it is better-suited for imbalanced classes and places more emphasis on positive cases, and is superior to F_1 since it considers the full range of classification thresholds.

2.2.2 Out-of-fold predictions

For further post-hoc tuning, out-of-fold (OOF) predictions were collected using the same expanding-window splits as in hyperparameter tuning, fitting to each fold's "training" set and storing the predicted probability values on the window's "test" set (Figure 5).

2.2.3 Tuning post-hoc parameters

Smoothing can improve predictions by preventing standalone points that differ from their neighbors.[vi] Median windowing is often used for image analysis,[21] but is effective at removing outliers or interruptions in the data while maintaining edges.[22]

By default, a threshold of 0.5 is selected in binary classification for categorizing a point as "True" or "False". However, in practice a model that is more or less sensitive to "True" may make final results more useful. These measures were found by finding the combination of median window size and threshold that maximizes the F_1 score,[23] which balances consideration for both precision and accuracy and can be a superior metric for class-imbalanced data.[24] Applying smoothing via median windows of sizes from 1 (no smoothing) through 35 were tested with 100 thresholds ranging from 0.01 through 0.99.

Removing marginal gains Prioritizing minor changes in F_1 can result in over-fitting or creating an overly-complex model process.[vii] Thus, the window/threshold combination with the smallest median window and an F_1 99% similar to the "best" F_1 was selected.

2.3 Model Testing

2.3.1 Fitting tuned model

Prior to analyzing performance, the tuned XGBoost model with optimal parameters was fitted to the entire training set (i.e., the first 80% of the full dataset) (Figure 5).

2.3.2 Test set prediction

Finally, the fitted model was used to predict the values on the test set (i.e., the held-out 20% of the full dataset). The outputs were stored as predicted probabilities in order to test the different threshold performances.

2.3.3 Performance analysis

To quantify performance improvement with and without the tuned threshold, results were calculated using the default, initial-best (with median smoothing), and final thresholds.

In binary classification, the predicted values are compared to the actual values of y. True positives (TP) are instances where the model-predicted value and the actual value are both "True". True negatives (TN) are when the prediction and actual value are both "False". False positives (FP) are instances where the model incorrectly predicts "True", while the actual value is "False".[viii] False negatives (FN) are when the model predicts incorrectly "False", as the actual value is "True".

[vi]For example, having a sequence of "True" interrupted by one "False", or vice-versa, both of which are unlikely in the context of obstructions.

[vii]For example, applying a large smoothing window for an F_1 gain of only 0.0003 in the OOF set may have diminishing returns as it may lose valuable information upon smoothing.

[viii]This is analogous to a false alarm.

Four metrics utilizing these values were used to evaluate model performance:
Accuracy[ix] quantifies the proportion of correct predictions:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \tag{1}$$

Precision measures the proportion of “True” predictions that were correct:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

Recall measures the fraction of actual “True” values that the model correctly identified:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

F_1 is a measure that combines precision & recall:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4}$$

[ix]This metric can be misleading on data with imbalanced classes. For example, if 90% of a data set’s true values are “False”, a model that always predicts “False” is still 90% accurate.

3 Results

A time series of runoff and soil moisture samples of the complete data set used in the analysis is shown in Figure 6. The original dataset had 22 features.[x] In total, 117 additional features were added via feature engineering. Applying the correlation threshold filtering of 0.97 resulted in the removal of 31 features from the feature dataset.[xi] This resulted in a total of 108 input features (X). The initial 80/20 split resulted in a training set spanning from 19 July 1989 11:55 through 08 March 2018 21:50 ($n = 2,778,145$) and a test set from 08 March 2018 21:55 through 01 August 2025 13:00 ($n = 694,537$).[xii]

The tuned XGBoost hyperparameters are provided in Table 3.

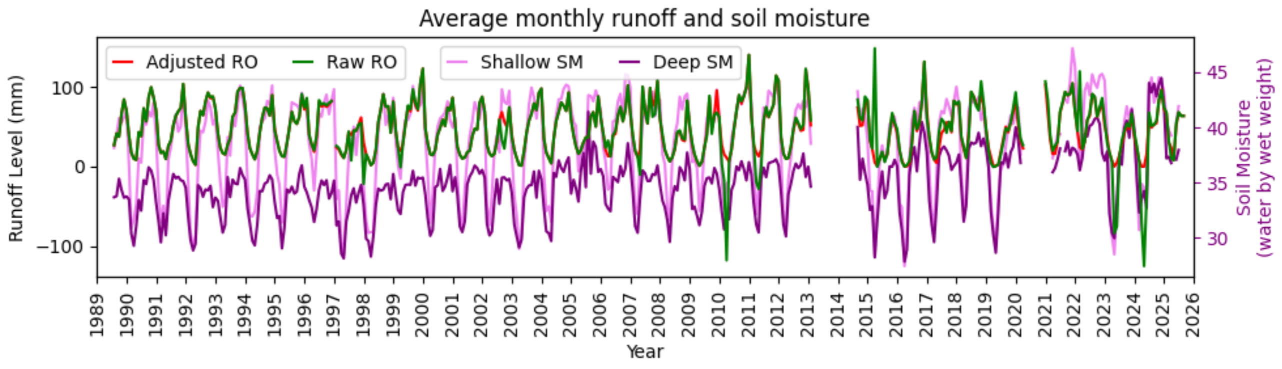

Figure 6: Time series showing monthly averages of raw (green) & adjusted (red) runoff and soil moisture values (pink & purple for shallow & deep, respectively).

Parameter	Purpose	Tuned value
n estimators	Number of trees	122
Learning rate	Impact of new trees	0.102
Max depth	Maximum depth of individual trees	3
Subsample	Random subset of training rows when building each tree	0.646
Column subsample by tree	Random subset of features when building each tree	0.709
Scale positive weight	Handles class imbalance	11
Gamma	Minimum loss reduction to split	0.128
Alpha	L1 regularization value	0.936

Table 3: Tuned hyperparameters for the XGBoost classification model of obstruction detection.

[x]These included raw runoff, rain, and 20 soil moisture measurements (as there were 10 total sites at two depths each).

[xi]These included: a forward-fill of the cumulative rain events; the cumulative rain event with extended decay; the runoff rolling sums of 10 min, 15 min, 30 min, & 1 hr; the differences in runoff value & rain from the previous entry; the runoff values from 10 min, 20 min, 30 min, & 40 min prior; the 15 min & 30 min lagged values of 3 hr & 6 hr rolling sums of rain; days since shallow soil samples 2 thru 10 & days since deep samples 1 thru 10 (thus, leaving "days since shallow sample 1" in the set).

[xii]Both sets are not perfectly continuous frequencies of 5 min due to gaps from occasional sensor failure, blips, and missing values. However, as previously mentioned, XGBoost can handle incomplete feature sets.

After applying the best hyperparameters and fitting to get OOF predictions, thresholds at different window sizes were tested and scored. The results of threshold and F_1 score for all smoothing windows are shown in Figure 7. The best post-hoc parameters initially selected were a smoothing windowing size of 29 and a classification threshold of 0.307 (F_1 = 0.463). After controlling for marginal gains (i.e., 99% similarity in F_1), the resulting values were no windowing and with a threshold of 0.317.

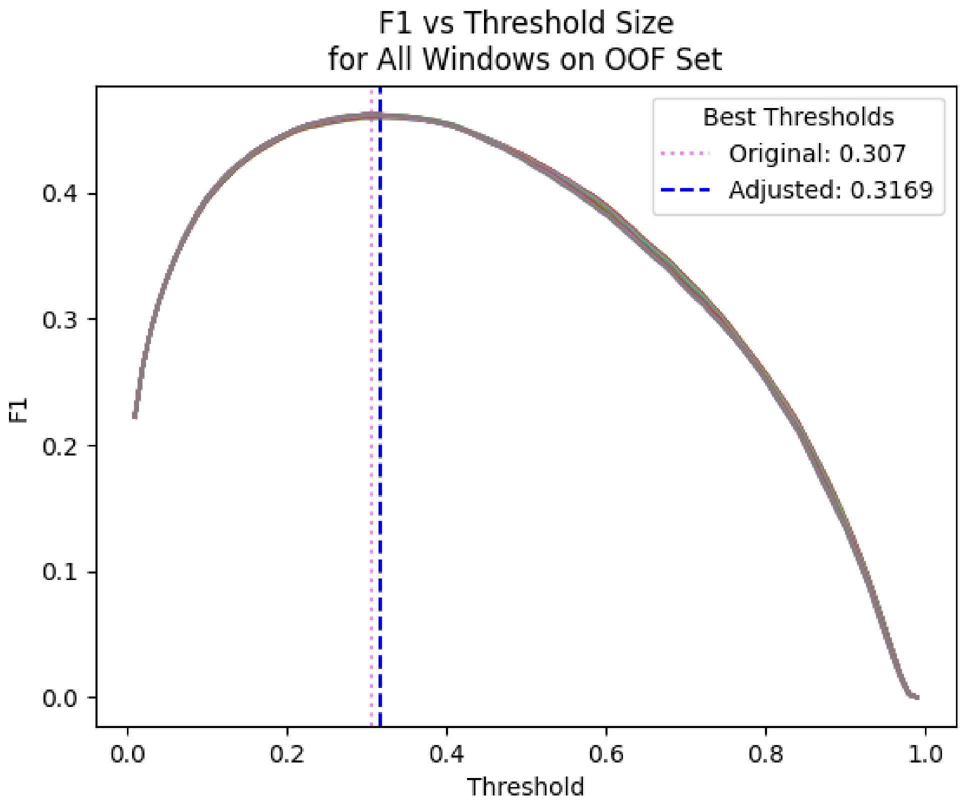

Figure 7: F_1 versus classification threshold for all windows on the out-of-fold set. All 30 windows reported extremely similar F_1 score curves as the classification threshold was adjusted, making it difficult to visually distinguish them individually.

The resulting precision-recall curve of the model after being fitted on the full training set and predicting probabilities of the held-out test set is shown in Figure 8. The metrics of model performance with different post-hoc parameters are reported in Table 4.

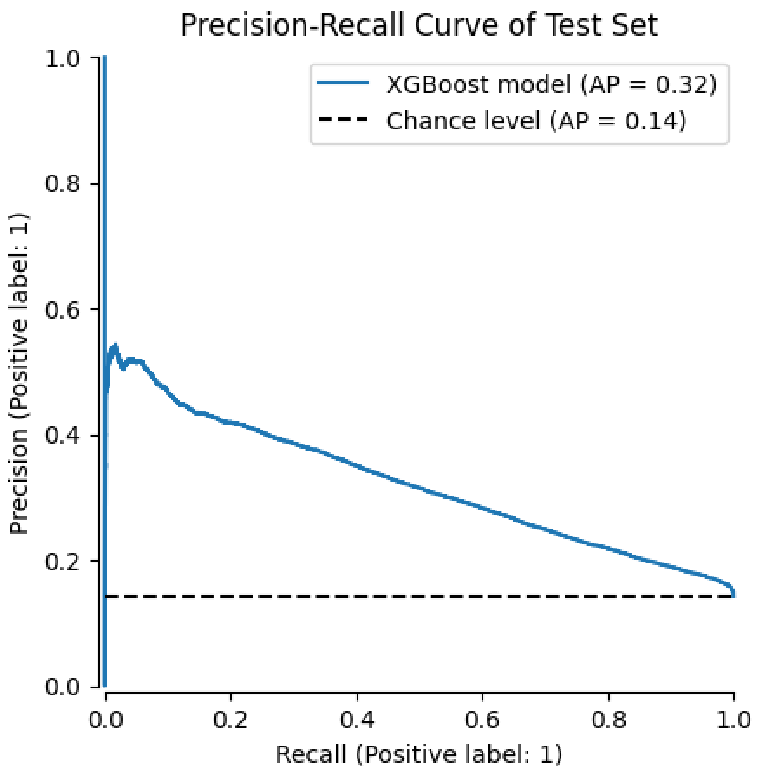

Figure 8: Precision-recall curve of the test set with average precision (AP) values.

Smoothing window	Threshold	Accuracy	Precision	Recall	F_1
-	0.500	0.667	0.253	0.686	0.370
29	0.500	0.669	0.255	0.686	0.371
29	0.307	0.509	0.204	0.845	0.329
-	0.317	0.518	0.206	0.838	0.331

Table 4: Performance metrics of the final model at different smoothing and thresholds.

A time series showing monthly averages of model accuracy over the complete test set is shown in Figure 9. An example of model performance in detecting blockages is shown in Figure 10.

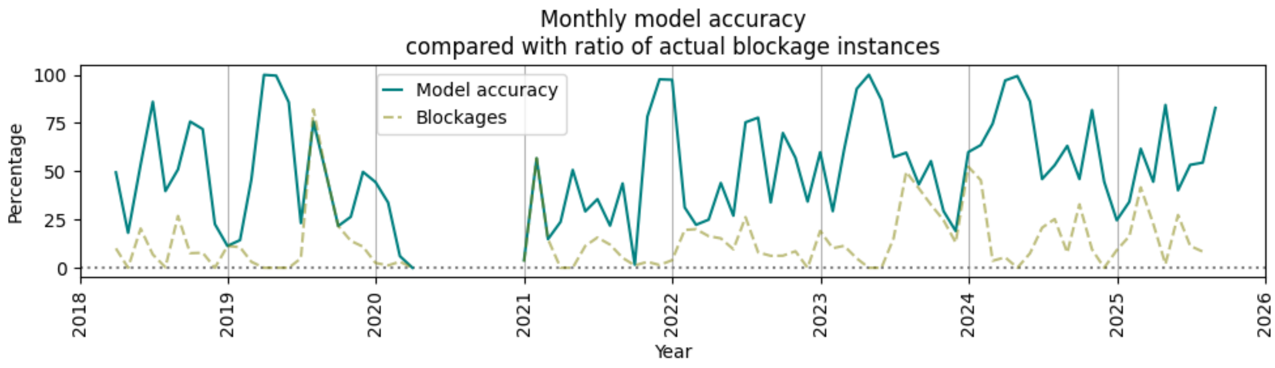

Figure 9: Monthly model accuracy (teal) and percentage of data containing blockages (light green).

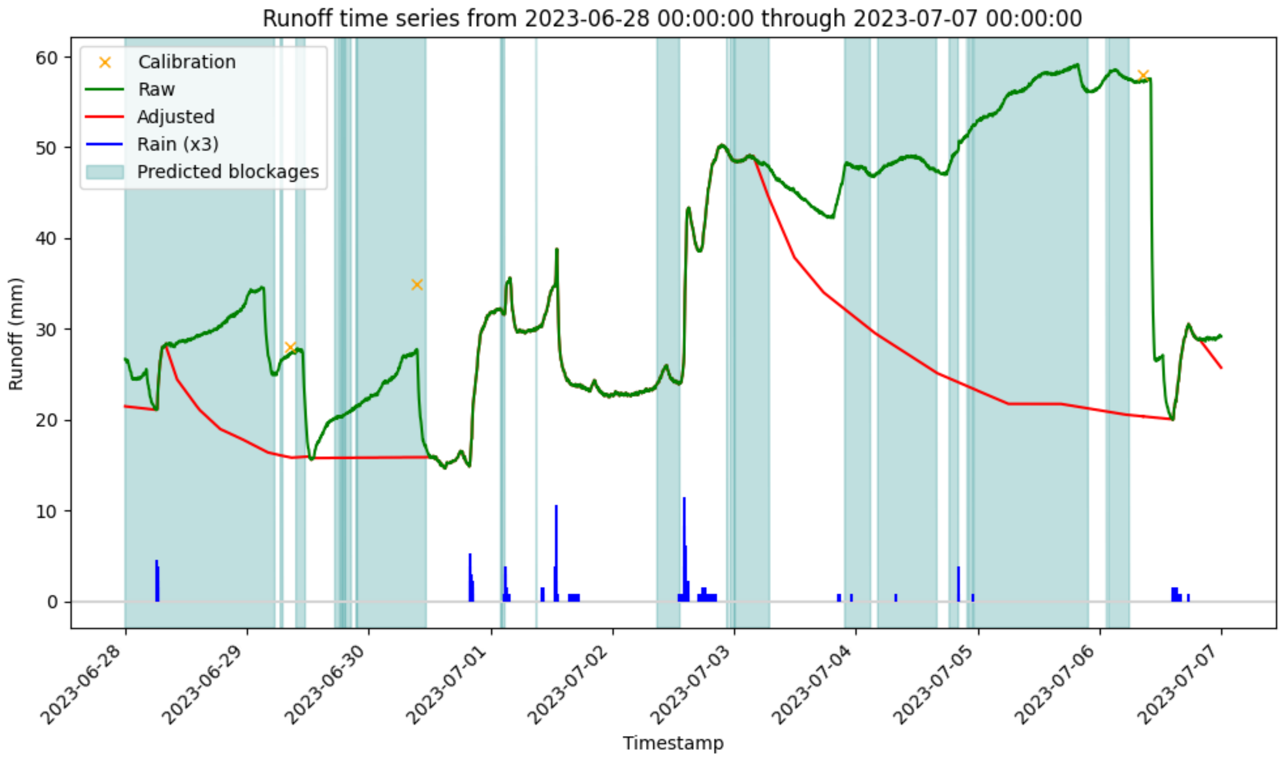

Figure 10: Example of final model performance on a small subset of the test data, where red lines are actual instances of an obstruction.

4 Discussion and Future Work

A reflection on some of the challenges faced over the course of the project can be found in Appendix B.

4.1 Interpretation of the Results

Based on Figure 7, it can be concluded that threshold tuning significantly improved model performance, but window smoothing appeared ineffective. The test results in Table 4 show that threshold tuning greatly improved recall but at the detriment of other metrics: high recall indicates that the tuned-threshold model can successfully identify most actual instances of obstructions, however the low precision indicates that many data points were erroneously flagged as being obstructions. The threshold selected from the absolute best smoothing window (i.e., prior to marginal gains filtering) had a slightly higher recall than the final selected threshold.

From Figure 9, model accuracy seems to improve when there are fewer actual instances of obstructions. The jagged accuracy line indicates that the current model will perform inconsistently on time periods with differing concentrations of obstruction events. The performance example in Figure 10 shows that the model successfully highlighted much of the obstruction events, but struggled to maintain a smooth window to indicate the full range of the failure mode.

It is possible that column subsampling (Table 3) could be negatively impacting model performance. An analysis by Chen and Guestrin[15] found that column subsampling in a classification experiment gave slightly worse performance compared to using all features, potentially due to the lack of "important" features in the dataset, and that the model may benefit from greedily selecting from all features instead. More advanced feature selection prior to modelling may remove the need for column subsampling, thus potentially improving the model by simplifying it and reducing dimensionality further. Despite XGBoost being able to yield successful results despite noisy data,[15] it is also possible that the presence of other failure modes in the raw runoff values are impeding model performance.

4.2 Future Work

Fitting large-parameter models usually requires very large training datasets to obtain satisfactory results. However, model performances usually do not scale linearly with the size of the training set; rather, after initial rapid improvements with increasing train set size, they often reach a point of diminishing returns.[25] This point will be a combination of the model accuracy required, the nature of the model and the data being used, and the cost of refitting the model with new data. Thus, it is proposed that the existing model be retrained with smaller subsets of the original training data, allowing performance of the resulting fits to be compared to a model based on the full training set in order to quantify how quickly the model improves with increasing training set size.[25] This will inform on how frequently the model might need to be updated (if at all) with additional data as it comes in in order to appreciably improve the results.[25]

Improvements to feature engineering for the XGBoost model based on expert input would likely benefit the model performance. The model only relied on look-behind features to avoid data leakage and reflect performance of a "live" model. This differs from the manual approach, which often use look-ahead metrics.[xiii] It is possible that performance would improve if instead the classification model was able to utilize such "look-ahead" features (as Zhang et al.[26] proposes in the context of failure predictions) although implementation of this would require careful consideration since it is atypical from usual time-series analysis, and would prevent the model from being able to predict on live data as it arrives in the database. Additionally, although small tests were conducted in an attempt to integrate a feature of the previous value's predicted obstruction state, this proved computationally infeasible at the

[xiii]For example, a steep drop-off in runoff values followed by a calibration point can indicate the end of a blockage.

time since high-power GPU access was integrated later in the project process. With this new resource, it is possible that integration of this feature may also improve results.

Future work will also improve on the model for obstruction detection, and create other models for the other failure modes. It is likely that different classification algorithms may have to be considered for other failure mode detection models, so further research will be necessary.

Additionally, other models are expected to be constructed in order to correct the actual raw runoff values during windows of failure modes to the appropriate adjusted values. If failure mode flagging models continue to have poor performance in regards to precision, it may be necessary to create a system to apply such correction models only to manually-confirmed windows of time, since modifying accurate data harms data quality & integrity. However, failure mode flagging models with good recall may still be able to be successfully utilized as a "first-pass" alert system for initial quality assurance checks.

Appendices

A Failure Mode Examples

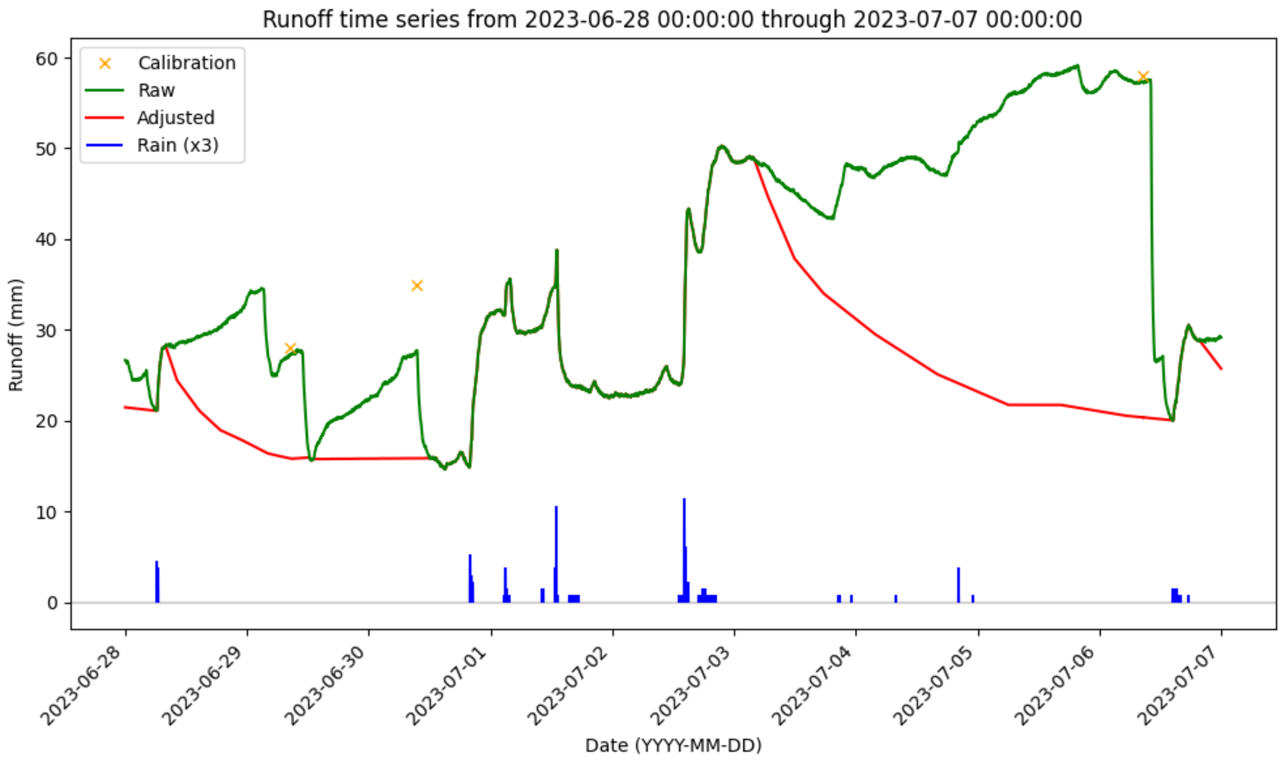

Figure A1: Example of consecutive distinct obstruction events.

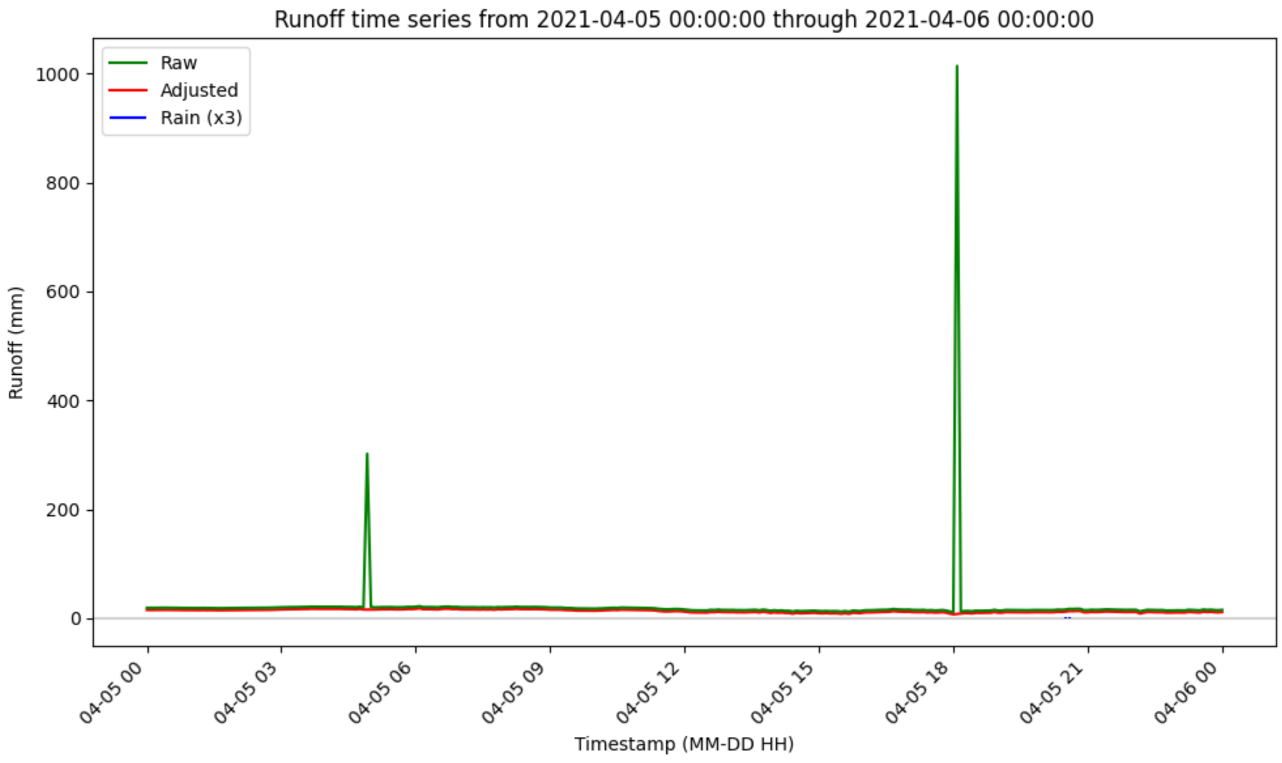

Figure A2: Example of two distinct spike events.

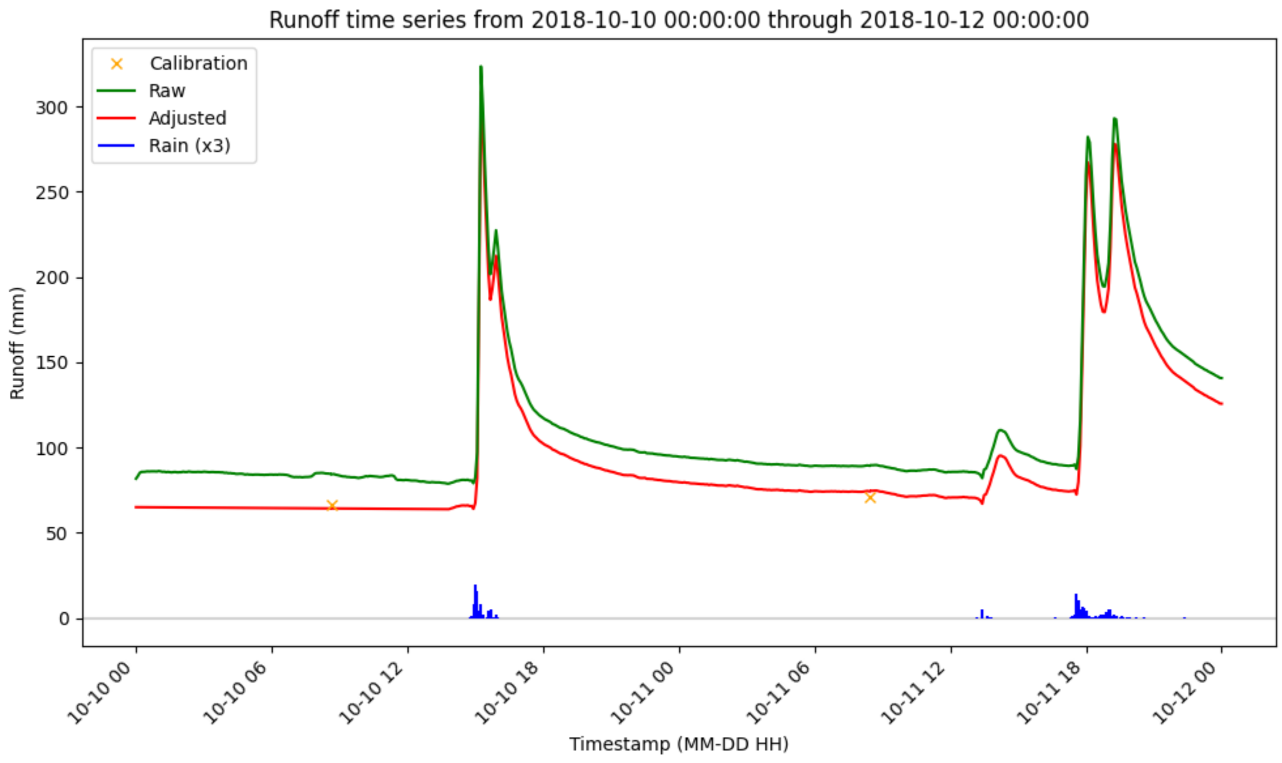

Figure A3: Example of an long-spanning calibration adjustment.

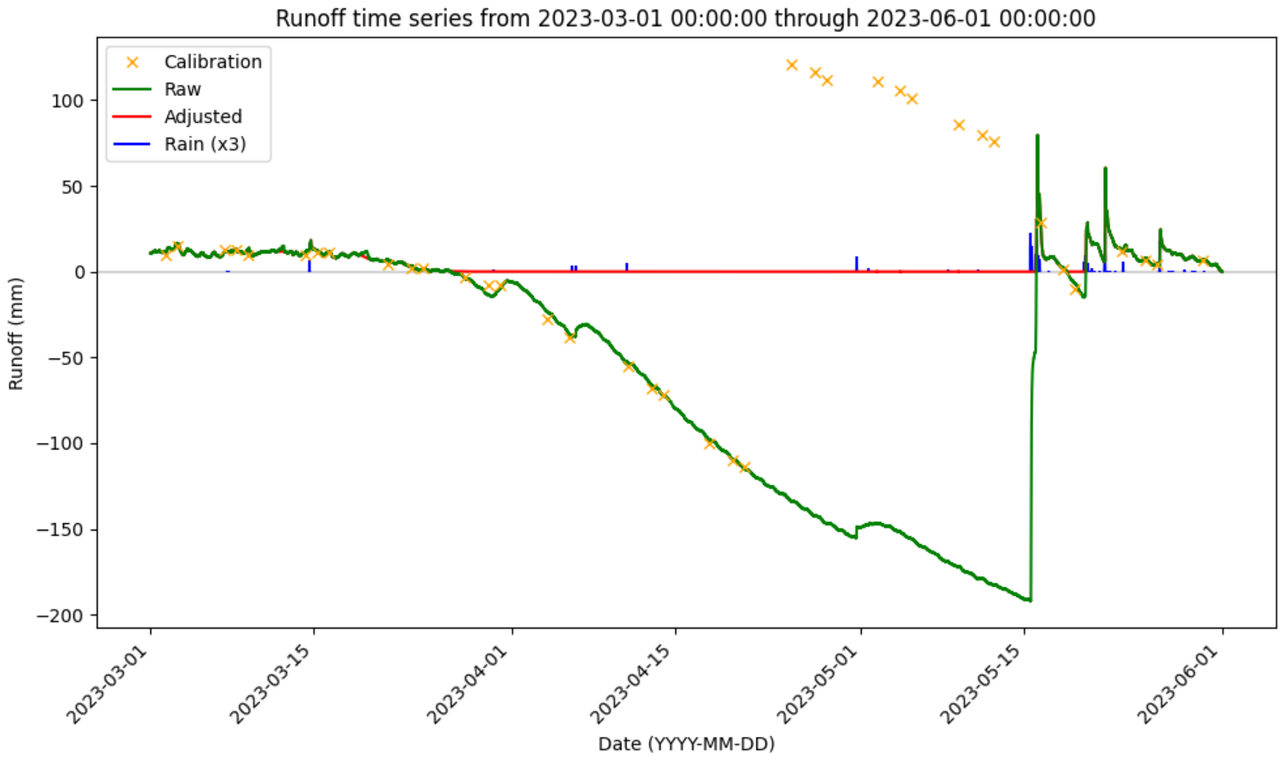

Figure A4: Example of sub-zero runoff measurements.

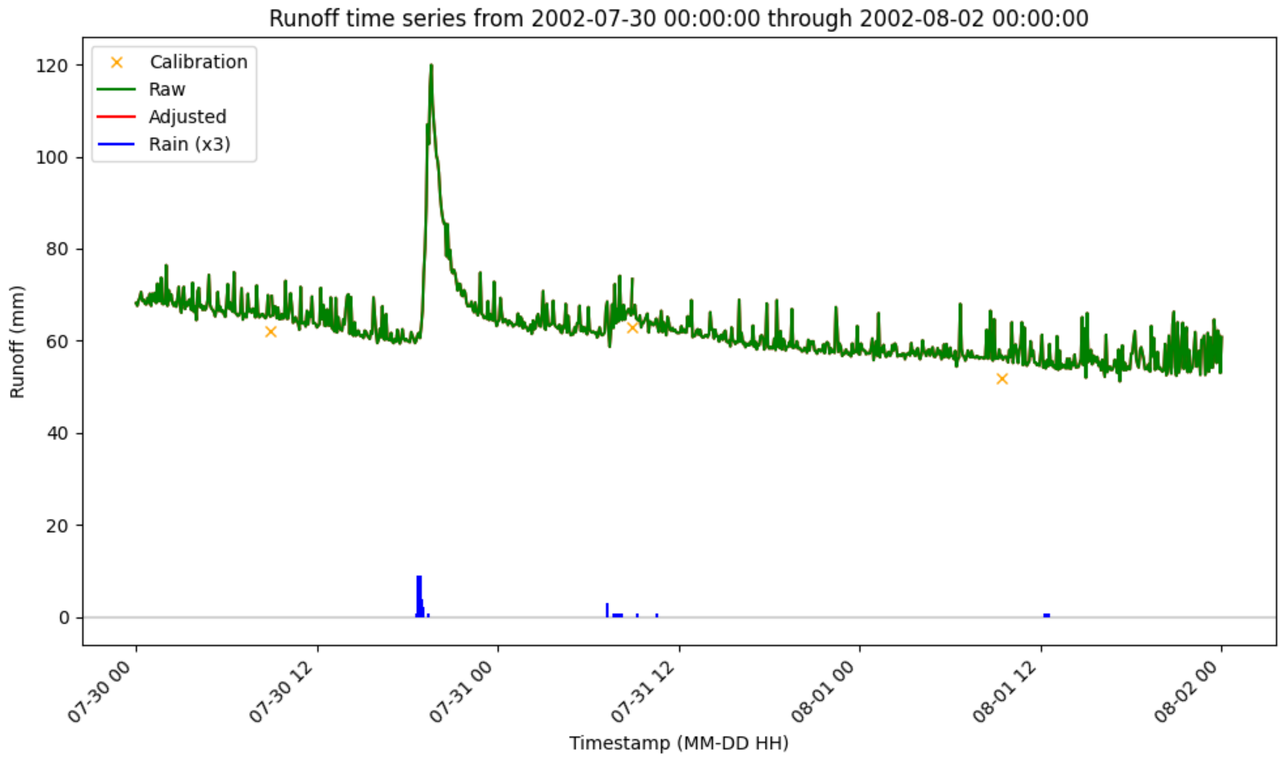

Figure A5: Example of noisy runoff measurements, which have yet to have accurate adjusted values assigned.

B Challenges

A number of unforeseen challenges were faced during the analysis, due in-part to the novelty and size of the project itself. It was known that past stochastic approaches had failed; thus, extensive background research was necessary to determine the appropriate machine learning model types. This also provided the opportunity to research and better understand various time-series analysis techniques and appropriate splitting & validation techniques.

Data cleaning and wrangling also took significantly longer than expected. Although all data was relatively uniform in its available state, filtering for the specific entries of interest and removing problematic entries without unnecessary data deletion was a challenge. Fortunately, this will reduce the preparation required for future analysis of other failure modes.

Communication with the expert data contact at STRI was not possible the majority of October and November due to the federal government shutdown. This did, however, provide the opportunity for more thoughtful data selection decisions since relying solely on external clarifications was not immediately available.

Computational power also proved a challenge, as there were more than 4.17 million data points (each with their own associated datetime-stamp) collectively in the imported sets. Even after removing irrelevant data points, the complexity was expanded further with feature engineering, as this resulted in most data points having more than one hundred individual values associated with it. Ultimately, a cloud-based high-RAM GPU proved the most efficient approach for the final model runs, although this required other code adjustments to further improve memory and prioritize parallel processing.

Although only one failure mode was able to be addressed in the duration of the project, it was specifically chosen because it was the most difficult, and because such machine learning analysis had yet to be applied in this way to hydrological time-series data on such a large scale.

References

(1) Paton, S. Runoff proposal suggestions, personal communication, 2025.

(2) Smithsonian Tropical Research Institute Barro Colorado (Clearing, Lutz, Conrad weir), 2025, https://striresearch.si.edu/physical-monitoring/barro-colorado/ (accessed 09/13/2025).

(3) Paton, S. BCI Hydrology Introduction, personal communication, 2025.

(4) Larsen, M. C.; Stallard, R. F.; Paton, S. Lutz Creek watershed, Barro Colorado Island, Republic of Panama. *Hydrological Processes* **2021**, *35*, e14157, DOI: 10.1002/hyp.14157.

(5) Mohammadi, B. A review on the applications of machine learning for runoff modeling. *Sustainable Water Resources Management* **2021**, *7*, 98, DOI: 10.1007/s40899-021-00584-y.

(6) Houmma, I. H.; Mansouri, L. E.; Gadal, S.; Garba, M.; Hadria, R. Modelling agricultural drought: a review of latest advances in big data technologies. *Geomatics, Natural Hazards and Risk* **2022**, *13*, 2737–2776, DOI: 10.1080/19475705.2022.2131471.

(7) Kebede Mengistie, G.; Demissie Wondimagegnehu, K.; Walker, D. W.; Tamiru Haile, A. Value of quality controlled citizen science data for rainfall-runoff characterization in a rapidly urbanizing catchment. *Journal of Hydrology* **2024**, *629*, DOI: 10.1016/j.jhydrol.2024.130639.

(8) Ryabko, D.; Mary, J. Reducing statistical time-series problems to binary classification. **2012**, DOI: 10.48550/ARXIV.1210.6001.

(9) Hudnurkar, S.; Rayavarapu, N. Binary classification of rainfall time-series using machine learning algorithms. *International Journal of Electrical and Computer Engineering (IJECE)* **2022**, *12*, 1945, DOI: 10.11591/ijece.v12i2.pp1945-1954.

(10) Li, D.; Lin, J.; Bissyandé, T.; Klein, J.; Traon, Y. L. In *21st International Conference on Extending Database Technology*, Vienna, Austria, 2018, DOI: 10.5441/002/EDBT.2018.19.

(11) Dritsas, E.; Trigka, M. Remote Sensing and Geospatial Analysis in the Big Data Era: A Survey. *Remote Sensing* **2025**, *17*, 550, DOI: 10.3390/rs17030550.

(12) Baek, M.; Kim, S. B. Failure Detection and Primary Cause Identification of Multivariate Time Series Data in Semiconductor Equipment. *IEEE Access* **2023**, *11*, 54363–54372, DOI: 10.1109/ACCESS.2023.3281407.

(13) Jakubowski, J.; Stanisz, P.; Bobek, S.; Nalepa, G. J. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE: Porto, Portugal, 2021, pp 1–10, DOI: 10.1109/DSAA53316.2021.9564228.

(14) Murphy, K. P. In *Probabilistic Machine Learning: An Introduction*; MIT Press: 2022; Chapter 18, pp 616–618, http://probml.github.io/book1.

(15) Chen, T.; Guestrin, C. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM: San Francisco California USA, 2016, pp 785–794, DOI: 10.1145/2939672.2939785.

(16) Mellah, S.; Trardi, Y.; Graton, G.; Ananou, B.; Adel, E. M. E.; Ouladsine, M. Semiconductor Multivariate Time-Series Anomaly Classification based on Machine Learning Ensemble Techniques. *IFAC-PapersOnLine* **2022**, *55*, 476–481, DOI: 10.1016/j.ifacol.2022.07.174.

(17) Zhu, J.-J.; Yang, M.; Ren, Z. J. Machine Learning in Environmental Research: Common Pitfalls and Best Practices. *Environmental Science & Technology* **2023**, *57*, PMID: 37384597, 17671–17689, DOI: 10.1021/acs.est.3c00026.

(18) Chu, C.-S. J. Time series segmentation: A sliding window approach. *Information Sciences* **1995**, *85*, 147–173, DOI: 10.1016/0020-0255(95)00021-G.

(19) Smithsonian Institution Smithsonian Research Data Repository, https://smithsonian.dataone.org/ (accessed 09/19/2025).

(20) Ranjan, K. G.; Tripathy, D. S.; Prusty, B. R.; Jena, D. An improved sliding window prediction-based outlier detection and correction for volatile time-series. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields* **2021**, *34*, e2816, DOI: 10.1002/jnm.2816.

(21) Rezaee, A. Partition Fuzzy Median Filter for Image Restoration. *Fuzzy Information and Engineering* **2021**, *13*, 199–210, DOI: 10.1080/16168658.2021.1921377.

(22) Kaminsky, R.; Liaskovska, S. In *SmartIndustry 2025: 2nd International Conference on Smart Automation & Robotics for Future Industry*, CEUR: Lviv Ukraine, 2025, https://ceur-ws.org/Vol-3970/PAPER18.pdf.

(23) Giannakopoulos, P.; Pikrakis, A.; Cotronis, Y. Improving Post-Processing of Audio Event Detectors Using Reinforcement Learning. **2022**, *10*, DOI: 10.48550/arxiv.2208.09201.

(24) Lipton, Z. C.; Elkan, C.; Naryanaswamy, B. In *Machine Learning and Knowledge Discovery in Databases*, ed. by Calders, T.; Esposito, F.; Hüllermeier, E.; Meo, R., Springer Berlin Heidelberg: Berlin, Heidelberg, 2014, pp 225–239, DOI: 10.1007/978-3-662-44851-9_15.

(25) Paton, S. Phase II module, personal communication, 2025.

(26) Zhang, X.-Y.; Feng, D.; Tan, Z.-P.; Xie, Y.-W.; Zhao, S.-F.; Wei, Y.-Y. LWCM: A Lookahead-Window Constrained Model for Disk Failure Prediction in Large Data Centers. *Journal of Computer Science and Technology* **2025**, *40*, 748–765, DOI: 10.1007/s11390-025-3850-4.