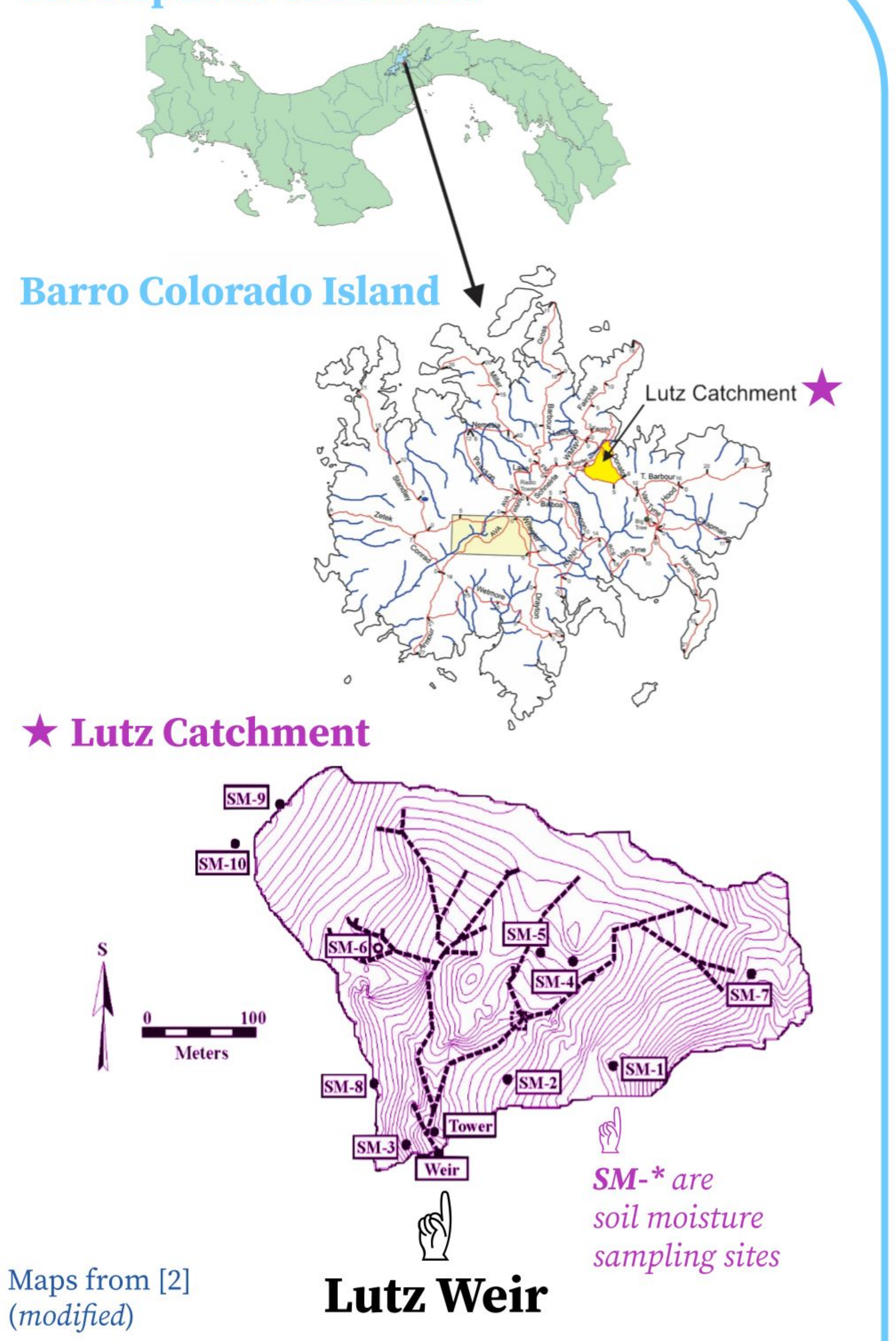
Machine Learning for Quality Assurance of Lutz Catchment Runoff Data

Gillian A. McGinnis, BA

University of Arizona College of Information Science

INFO 698 Capstone Project, Fall 2025



Maps from [2] (modified)

Background

Water **runoff data** has been continuously collected by the Smithsonian Tropical Research Institute (STRI) at the **Lutz Catchment weir** on Barro Colorado Island since 1972, & with electronic sensors **since 1989.** [1, 2]

Different "**failure modes**" impact data quality, resulting in the need for adjustments which are currently conducted manually.

★ Goals

Determine if models can be constructed to **effectively identify windows of failure** and eventually conduct **quality assurance** corrections on the raw runoff values without the need for manual intervention.

Challenges

➢ **New project & approach**
Past stochastic approaches had failed; extensive background research was necessary to determine appropriate machine learning model types

➢ **Gaps & missing data**
Sensor failures, missing values, and inconsistent labels, comments, & flags

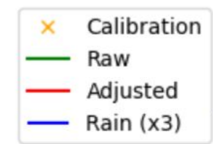
➢ **Differences in data frequency**
Runoff & rain:
every 5 minutes for 36 years from 1 site
Soil moisture:
once a week from 10 sites ×2 depths each

➢ **Computational power**
More than 4.15 million total entries

➢ **Access & communication**
Primary data contact was unavailable the majority of Oct & Nov due to the federal government shutdown

Failure Modes

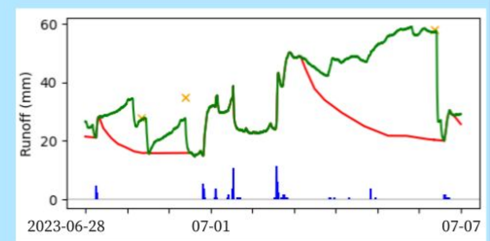
Five major failure modes cause data quality issues.



Rain tripled to help visibility

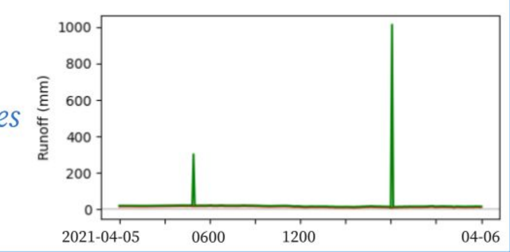
Blockage ★

Debris blocks the weir's 'V'
Fix: data pivot or decay curve
***Most difficult failure mode to identify & correct**



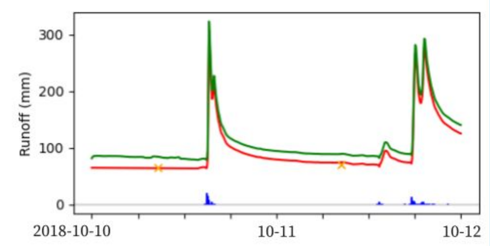
Spike

Short & abrupt changes in level
Fix: smoothing to neighbors' values
*Differs from sudden increase in flows following heavy rain



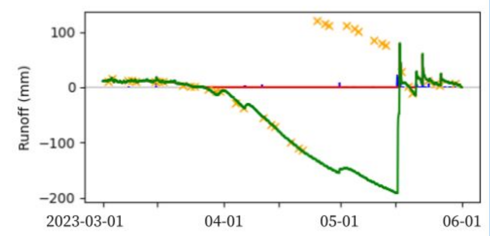
Calibration

Misalignment with standard (×)
Fix: baseline correction
*Points used for corrections, but **Blockage** can render ineffective



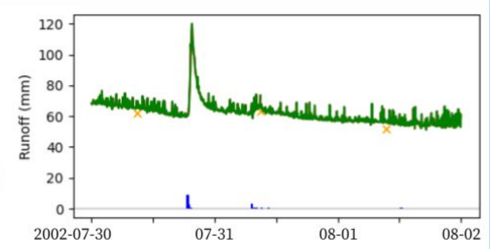
Sub-Zero

Stream runs dry or is drained
Fix: setting to zero
*Data after rain may be unrecoverable



Signal Noise

Equipment failure
Fix: N/A
*Impossible to manually correct, and resists standard de-noising



Methods for Blockage flagging

Language: Python (*Jupyter Notebooks*)
Dataset: n = 3,472,682 annotated points
Process:

1. Introduce **raw data** (*imported from CSV*)
2. **Prepare** data (*clean & wrangle*)
3. Engineer **features** (*time, lags, & rolling stats*)
4. Remove **high-correlation** features (*>0.97*)
5. Conduct **Train/Test** split (*80:20*)
6. Tune XGBoost **hyperparameters** (*w/ PR-AUC—good for imbalanced classes*)
7. Fit to get **out-of-fold** predictions (*OOF*)
8. Tune post-hoc **smoothing** parameters w/ F1 (*median-windowing & classification threshold*)
9. **Fit** tuned XGBoost model to entire training set
10. Predict on the **test set** (*the held-out 20%*)
11. **Analyze** results (*w/ & w/o post-hoc params.*)

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

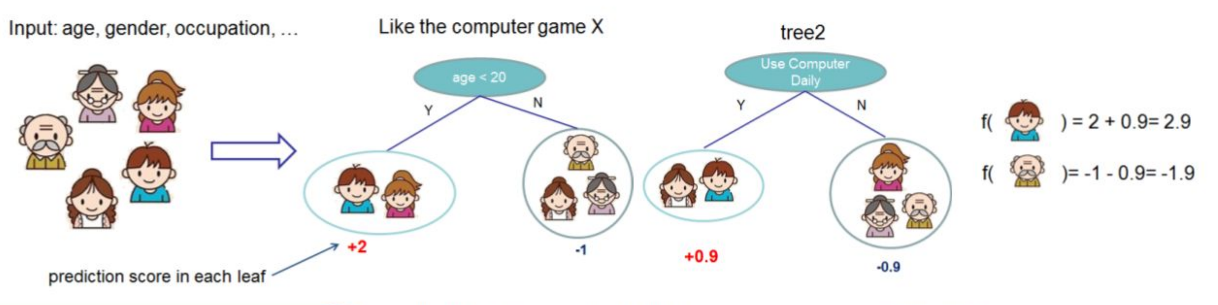
XGBoost: Extreme Gradient Boosting

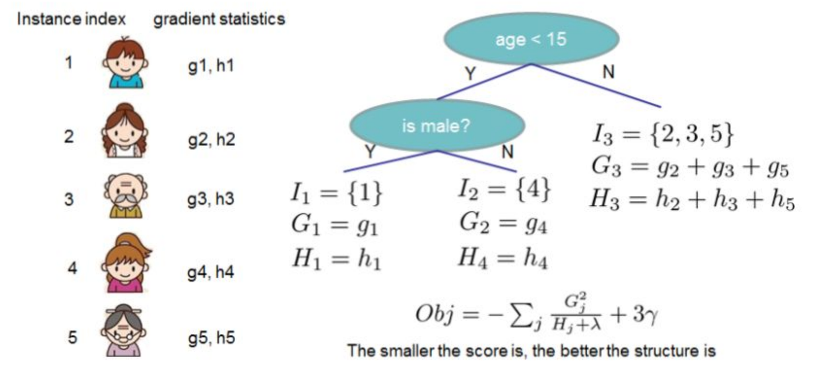
XGBoost is a form of **gradient tree boosting**, which iteratively adds **weak learners** (*shallow, simple decision trees*) using gradient descent to minimize error.

It can also **handle missing & incomplete data.**

The final model is an ensemble, which sums up the predictions from all trees.

*Simple **example** from XGBoost documentation [3]*





Advantages of XGBoost over standard gradient tree boosting:

\+ adds a regularizer
\+ utilizes second-order loss approx'n
\+ can sample features internally
\+ scales better

Training Results

n = 2,778,145
19 Jul 1989 11:55 *-thru-* 08 Mar 2018 21:50*

Number of features:

	22 original	*Runoff, rain, and ×20 soil*
+	117 engineered	*Lags and rolling stats*
–	31 high-correlated	*feats >0.97 correlated w/ another*
=	**108 input features**	

Hyperparameters:

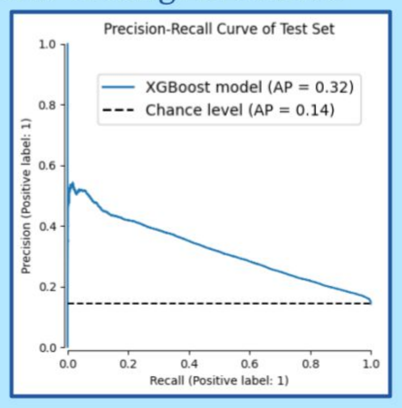
***n* estimators**	**122**	***Number of trees***
Learning rate	**0.102**	***Impact of new trees***
Max depth	**3**	***Max depth of individual trees***
Subsample	0.6455	*Random subset of training rows when building each tree*
Column " by tree	0.709	*Random subset of features*
Scale pos. weight	11	*Handles class imbalance*
Gamma	0.128	*Minimum loss reduction to split*
Alpha	0.936	*L1 regularization*

Post-hoc smoothing & tuning:
Median window size = 29 & threshold = 0.307
After removing marginal gains in scoring:
No median windowing & threshold = 0.317

Test Set Performance

n = 694,537
08 Mar 2018 21:55 *-thru-* 01 Aug 2025 13:00*

PR-curve ☞ of fitted model on the test set (*shows performance across different thresholds*)



	Accuracy	Precision	**Recall**	F1
Defaults	0.667	0.253	**0.686**	0.397
Win = 29	0.669	0.255	0.686	0.371
Win = 29 & Th = 0.307	0.509	0.204	**0.845**	0.329
Th = 0.317	0.518	0.206	**0.838**	0.331
Change	*−0.149*	*−0.047*	*+0.152*	*−0.066*

Both set are **not perfectly continuous every 5 min due to gaps from occasional sensor failure, blips, & missing values*

Interpretations

The model can **identify most true blockages** (*high recall*), but frequently has false alarms (*low precision*).

Threshold tuning improved model performance in identifying true blockages, but median window **smoothing proved ineffective.**

Limitations

The engineered features for XGBoost are **look-behind**, whereas manual corrections often use **look-ahead.**
For example, a steep dropoff in runoff values followed by a calibration point (×) can indicate the end of a blockage.

Other failure modes (*e.g., Spikes*) may result in noisy data that the model struggles with interpreting.

Future Work

➢ Create models for other failure modes
Not all modes will require models as complex as XGBoost due to reliance on fewer features, and it is likely that **different classification algorithms may have to be considered for other failure mode detection models.**

➢ Address data correction automation
If a flagging model has poor performance, it may be necessary to make systems to apply adjustment models to manually-confirmed windows, since **modifying accurate data harms quality.**

Acknowledgements

Capstone advisor: Sriram Iyengar, PhD
(*The University of Arizona College of Medicine, Phoenix*)
Data advisor, source, & access: Steven Paton, MSc
(*Director of the Physical Monitoring Program, STRI*)
GPU assistance: Brian E. McGinnis, PhD

References

[1] *Barro Colorado (Clearing, Lutz, Conrad weir)*. Physical Monitoring.
[2] Larsen, M. C.; Stallard, R. F.; Paton, S. Lutz Creek Watershed, Barro Colorado Island, Republic of Panama. *Hydrological Processes* **2021**, *35* (4), e14157.
[3] *Introduction to Boosted Trees — xgboost 3.1.1 documentation.*