Fault Injection Design Document
WaDi A1 Dataset

CS 6678 - Advanced Machine Learning
Instructor: Dr. Leslie Kerby

George Lake

Purpose and Scope

This document defines the methodology for injecting synthetic sensor failures into the WaDi A1 (Water Distribution, 9 Oct 2017) Industrial Control System dataset. WaDi provides ground-truth labels for normal operation (label=0) and cyberattacks (label=1) but contains no labeled sensor failure events. Injecting realistic synthetic failures creates the third classification class required for the core research question:

> *Can machine learning distinguish between benign sensor failures and malicious cyber attacks in Industrial Control Systems?*

This document covers:

1. A literature-grounded taxonomy and parameters for five fault types.
2. The injection strategy defining which sensors are eligible, how faults are placed, and how fault density is controlled.
3. The labeling scheme for the three-class target variable.
4. A class balance analysis based on actual injection results.
5. Validation checks executed programmatically after injection.

1 Literature-Grounded Fault Taxonomy

Synthetic fault injection is a well-established methodology in ICS and sensor fault detection research. Peer-reviewed literature consistently identifies four to seven canonical fault types that account for the vast majority of real-world sensor failures [1, 2, 3]. This section defines each fault type, grounds its parameters in published evidence, and notes its relevance to the attack-versus-failure confusion problem.

1.1 Canonical Fault Type Classification

The sensor fault literature converges on four primary categories: bias, drift, precision degradation, and complete failure (freezing/stuck-at) [1, 2, 3]. Extended taxonomies add gain faults and intermittent dropouts [4, 5]. Synthetic injection of these fault types into clean sensor data is a standard approach for generating labeled fault datasets where real failure examples are unavailable [6]. This project implements five fault types:

- **Drift**: gradual, monotonic linear deviation from true value over time.
- **Bias (Offset)**: sudden constant shift applied to all readings in the fault window.
- **Precision Degradation (Noise)**: increased measurement variance without systematic shift.
- **Stuck-At (Freeze)**: sensor reports a fixed value regardless of actual process state.
- **Intermittent Dropout**: random NaN values simulating signal loss or transmission fault.

1.2 Parameter Justification and Literature Basis

Table 1 summarizes the implemented parameter ranges for each fault type. All parameters are expressed relative to each sensor's observed standard deviation during normal operation, computed per-sensor from training data. This sensor-relative scaling ensures that fault magnitudes are physically plausible across sensors with different units and operating ranges.

Table 1: Fault type parameter ranges and literature justification

Fault Type	**Parameter**	**Implemented Range**	**Justification**
Drift	Rate	0.5–3.0% of sensor std/min	Gradual calibration degradation in ICS sensors. Rates below 0.5% are undetectable within 2–10 minute windows; rates above 3% trigger threshold alarms within minutes [3].
Bias	Magnitude	±10–40% of sensor std	Sudden calibration shift consistent with electrolyte degradation in electromechanical sensors. Direction (positive or negative) sampled uniformly at random [1].
Precision Degradation	Noise multiplier	3×–8× sensor std	Aging components and loose connections modeled as Gaussian noise scaled to the sensor's baseline standard deviation [5].
Stuck-At	Frozen value	5th–95th percentile of normal range	Sensor freezes at a value drawn from its observed normal operating range. Avoids trivially implausible values at the extremes of the distribution [2].
Intermittent Dropout	Dropout rate	20–60% of rows set to NaN	Communication fault or loose wiring producing random signal loss. Rate sampled uniformly within the window [4].

2 Relevance to Attack-Failure Confusion

Three of the five fault types are particularly relevant to the attack-versus-failure confusion problem because they produce sensor signatures that overlap with known WaDi attack patterns:

- **Stuck-At vs. Sensor Value Freeze Attack:** WaDi attacks include scenarios that maintain a previous sensor value to conceal process manipulation. A stuck-at hardware fault produces an identical time-series signature. This is the highest-difficulty confusion pair and the primary motivation for this research.
- **Drift vs. Setpoint Manipulation Attack:** Slow setpoint manipulation attacks produce a gradual deviation in process variables visually similar to sensor drift. The key discriminating feature is cross-sensor correlation — a cyber attack affecting a setpoint will also perturb related control variables; a hardware drift fault will not.
- **Bias vs. False Data Injection:** A sudden constant offset mirrors false data injection attacks. Discriminating features include whether the offset is physically plausible given the process state at onset, and whether correlated sensors are simultaneously affected.

3 Injection Strategy

3.1 Sensor Eligibility Criteria

Not all sensors are suitable injection targets. Eligibility is determined programmatically using two criteria:

- **Analog sensors only:** Sensors with five or fewer unique values (`BINARY_MAX_UNIQUE = 5`) are classified as binary or status columns and excluded. Injecting continuous fault types such as drift or bias into discrete status columns produces physically implausible results.
- **Minimum data coverage:** Sensors must have at least 10% non-null normal rows (`MIN_COVERAGE_RATIO = 0.10`) to provide a sufficient window for fault placement.

These checks are applied to normal rows only. Sensors that pass both criteria constitute the eligible pool used for all injection events.

3.2 Fault Density and Duration

Fault injection targets approximately 8% of normal rows per split (`TARGET_FAULT_PCT = 0.08`). This target is achieved dynamically: the orchestrator computes the required number of fault events per sensor based on the eligible sensor count, the target percentage, and the average expected fault duration, then places non-overlapping windows up to that count per sensor.

Each fault window duration is sampled uniformly between 120 and 600 seconds (2–10 minutes), reflecting realistic detection and repair windows in operational water distribution systems [2]. Variable duration prevents the model from using window length as a trivial separating feature.

3.3 Temporal Placement Rules

Correct temporal placement is critical to label integrity. The following rules are enforced programmatically:

- **Train split only:** All injections occur within the training split. The test split retains original WaDi labels unmodified. This ensures the evaluation set is never contaminated with synthetic data.
- **No overlap with attack rows:** Fault windows are placed within normal rows only. No fault window can overlap any row where `label = 1`. This is enforced by working exclusively from `df_normal` (rows where `label = 0`) during placement, and verified by post-injection validation.
- **Non-overlapping per sensor:** A sensor may not have two overlapping fault windows. An occupancy array tracks placed windows per sensor and rejects any proposed placement that intersects an existing window. Up to 50 placement attempts are made per event before skipping.
- **Concurrent faults on different sensors:** Simultaneous faults on different sensors are permitted, which is realistic — multiple sensors can fail independently at the same time.

3.4 Fault Type Assignment

Fault type is sampled uniformly at random from the five available types for each injection event. Fault severity parameters are then sampled uniformly within the ranges defined in Table 1. This produces a roughly equal distribution across fault types with natural variation in severity within each type.

3.5 Reproducibility

A fixed global random seed (`RANDOM_SEED = 42`) is set at notebook startup and used to seed a per-split `numpy.random.Generator`. Re-running the notebook with the same seed against the same staged parquet produces bit-identical results. A different seed produces an alternative valid injection suitable for sensitivity analysis.

4 Labeling Scheme

The injection process produces a revised label structure supporting both the primary three-class classification task and secondary analysis by fault subtype. All fault metadata columns are added to the dataframe at injection time. Non-fault rows retain `NaN` in fault-specific columns, preserving the original WaDi labels for normal and attack rows unchanged.

Table 2: Label columns added during fault injection

Column	Type	Values	Description
`label`	int8	0, 1, 2	Primary classification target: 0=normal, 1=cyber attack, 2=sensor failure.
`fault_type`	str / NaN	drift, bias, precision_degradation, stuck_at, intermittent_dropout, NaN	Fault subtype for secondary analysis; NaN for normal and attack rows.
`fault_sensor`	str / NaN	e.g. `1_AIT_001_PV`, NaN	Sensor receiving the injection; NaN for non-fault rows.
`fault_start`	datetime / NaT	UTC timestamp or NaT	Start timestamp of the injected fault window.
`fault_end`	datetime / NaT	UTC timestamp or NaT	End timestamp of the injected fault window.
`fault_severity`	float / NaN	Fault-type specific	Sampled severity parameter: drift rate, bias offset, noise multiplier, stuck percentile, or dropout rate. NaN for non-fault rows.

These metadata columns are retained in the injected parquet for validation and post-hoc error analysis. They must be explicitly excluded from the feature matrix in the feature engineering stage — they are injection metadata, not observable sensor signals.

5 Class Balance

Table 3 summarizes the actual class distribution produced by the injection pipeline on the WaDi A1 dataset.

Table 3: Actual class distribution after fault injection

Class	Source	Rows	Notes
0 — Normal	WaDi A1 normal period (train split)	1,116,251 (80.75%)	Sep 25–Oct 6 training period.
1 — Cyber Attack	WaDi A1 attack period (test split)	172,801 (12.50%)	Oct 6–9 attack period; 15 staged attacks.
2 — Sensor Failure	Synthetic injection into train normal rows	93,350 (6.75%)	≈8% of train normal rows; fault type distributed across 5 types.
Total		**1,382,402**	80.75% / 12.50% / 6.75% split.

The resulting class distribution presents moderate imbalance. The attack class (12.50%) and fault class (6.75%) are both genuine minorities relative to normal, but neither is severely underrepresented. The Random Forest classifier is trained with `class_weight='balanced'`, which scales each class's contribution to the loss function inversely proportional to its frequency. This is the primary imbalance mitigation strategy and requires no resampling.

The fault class is smaller than the attack class, which is notable. This reflects the deliberate decision to target 8% of train normal rows — a conservative injection density chosen to avoid overwhelming the normal class while providing sufficient fault examples for learning.

6 Validation Checks

The following checks are executed programmatically immediately after injection and before writing any output to disk. Any failed check raises a `PipelineError` and halts execution.

Table 4: Label columns added during fault injection

Column	Type	Values	Description
`label`	int8	0, 1, 2	Primary classification target: 0=normal, 1=cyber attack, 2=sensor failure.
`fault_type`	str / NaN	drift, bias, precision_degradation, stuck_at, intermittent_dropout, NaN	Fault subtype for secondary analysis; NaN for normal and attack rows.
`fault_sensor`	str / NaN	e.g. `1_AIT_001_PV`, NaN	Sensor receiving the injection; NaN for non-fault rows.
`fault_start`	datetime / NaT	UTC timestamp or NaT	Start timestamp of the injected fault window.
`fault_end`	datetime / NaT	UTC timestamp or NaT	End timestamp of the injected fault window.
`fault_severity`	float / NaN	Fault-type specific	Sampled severity parameter: drift rate, bias offset, noise multiplier, stuck percentile, or dropout rate. NaN for non-fault rows.

All six checks passed with no warnings on the production run (`RUN_ID: 20260223_145035_utc`).

7 Known Limitations

- **Parameter realism without ground truth:** The chosen parameter ranges are grounded in published literature but are not validated against real WaDi sensor failure data, which does not exist in this dataset. This is an inherent limitation of synthetic fault injection methodology and is acknowledged explicitly in the literature [6].
- **Stuck-at versus attack indistinguishability:** If stuck-at faults are truly indistinguishable from WaDi sensor-freeze attacks at the feature level, the classifier may perform poorly on this specific class pair. This outcome is itself a meaningful finding and would be reported as such rather than treated as a methodology failure.
- **No test set faults:** The test split contains only real attack rows. Fault generalization to real (non-synthetic) sensor failure events is therefore not evaluated in this work and is identified as future work requiring a dataset with labeled real failures.
- **Rolling window boundary effects:** Fault injection occurs before feature engineering. Rolling window features computed near the boundary of a fault window will capture a mix of normal and fault-affected readings. This is realistic — in a real system the exact fault onset time is unknown — but introduces ambiguity near fault boundaries that may modestly degrade classification performance.

References

[1] D. Narzary and K. C. Veluvolu, "Multiple Sensor Fault Detection Using Index-Based Method," vol. 22, no. 20, p. 7988. [Online]. Available: https://www.mdpi.com/1424-8220/22/20/7988

[2] T. Yuan, W. Xu, K. H. Adjallah, H. Wang, L. Liu, and J. Xu, "A Risk Evaluation Framework in System Control Subject to Sensor Degradation and Failure," vol. 24, no. 5, p. 1550. [Online]. Available: https://www.mdpi.com/1424-8220/24/5/1550

[3] L. Zhang and M. Leach, "Evaluate the impact of sensor accuracy on model performance in data-driven building fault detection and diagnostics using Monte Carlo simulation," vol. 15, no. 5, pp. 769–778. [Online]. Available: https://link.springer.com/10.1007/s12273-021-0833-4

[4] E. Balaban, A. Saxena, P. Bansal, K. F. Goebel, and S. Curran, "Modeling, Detection, and Disambiguation of Sensor Faults for Aerospace Applications," vol. 9, no. 12, pp. 1907–1917. [Online]. Available: http://ieeexplore.ieee.org/document/5297818/

[5] J. Kullaa, "Detection, identification, and quantification of sensor fault in a sensor network," vol. 40, no. 1, pp. 208–221. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327013002057

[6] B. De Bruijn, T. A. Nguyen, D. Bucur, and K. Tei, "Benchmark Datasets for Fault Detection and Classification in Sensor Data:," in *Proceedings of the 5th International Confererence on Sensor Networks*. SCITEPRESS - Science and and Technology Publications, pp. 185–195. [Online]. Available: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005637901850195