Data contracts and evaluation leakage in historical bike-rental analytics

Technical research manuscript - draft, not peer reviewed

Prepared for the portfolio of Hanumantha Rao Vulli | 18 October 2026

Abstract

This reproducible case study evaluates a local data-quality pipeline and retrospective count estimation on 17,379 real hourly bike-rental observations. We compare a training-mean baseline, a random forest using calendar and observed weather, and an intentionally invalid target-component control. Across three seeds, random-split forest mean absolute error (MAE) is 26.034 rentals/hour, compared with 45.687 under a chronological split. The target-component control produces much lower error but uses information that directly constitutes the outcome. In 54 controlled fault-policy evaluations, domain checks detect invalid hours but miss appended duplicates and count inconsistencies; explicit reconciliation and key checks detect the injected cases. These results illustrate evaluation and contract risks in this dataset; they do not establish a new algorithm, production readiness, or generalization to other systems.

1. Practical question and prior work

Can a small, inspectable data contract expose faults missed by domain checks, and how much do evaluation design and target leakage change apparent predictive quality?

Breck et al. (2019) describe data validation as an important part of machine-learning infrastructure and distinguish within-batch from across-batch issues. Our study addresses a narrower, local case: row constraints and transparent evaluation, without reproducing that production system or comparing against TensorFlow Data Validation. The UCI Bike Sharing dataset supplies real historical counts and contextual variables (Fanaee-T, 2013). Scikit-learn's evaluation guidance explains why model assessment must keep training and test data distinct; our chronological comparison additionally respects record order (scikit-learn developers, n.d.).

The contribution is an original portfolio implementation and a reproducible case study combining data-contract checks, a read-only API, and explicit leakage controls. The general methods are established. No state-of-the-art or worldwide-novelty claim is made.

2. Data and engineering method

The redistributed hourly CSV contains 17,379 records from 2011–2012, 731 observed dates, and 3,292,679 rentals. It has 165 missing date-hour slots across those dates. Missing slots are not imputed as zero. File hashes and licensing attribution appear in projects/mobility-quality/data/SOURCE.md. The source landing-page count differs by ten rows from the actual file; analyses use the measured file count.

The local pipeline checks schema, parsing, numeric domains, calendar consistency, cnt = casual + registered, and unique IDs/date-hour keys. It quarantines violations by input line. Valid records and a batch audit are committed to SQLite in a single transaction. Input hashes identify active snapshots and enable repeat-safe loading. No original row violates the implemented rules; this does not prove that every observation is accurate.

3. Evaluation protocol

Retrospective estimation

Sort observations by date and hour. The chronological split uses the earliest floor(0.8N) observations for training and the remainder for testing: 13,903 and 3,476 rows. Random splits use the same approximate 80/20 proportions. Seeds are 11, 23, and 37. The chronological boundary can split a day; future work should use whole-day rolling-origin folds.

The baseline predicts the training-set mean. The forest has 60 trees, maximum depth 16, minimum leaf size 2, one worker, and otherwise library defaults. Parameters were fixed for this illustrative experiment; no hyperparameter selection is claimed. Legitimate predictors are season, year indicator, month, hour, holiday, weekday, working-day indicator, weather category, normalized temperature, apparent temperature, humidity, and windspeed. Record ID, date string, casual, registered, and cnt are excluded from this feature set.

The positive control deliberately adds casual and registered, whose sum equals cnt. Its low error is invalid as evidence of predictive performance. Since contemporaneous observed weather is used, all comparisons are retrospective estimation, not an operational future-weather forecast. Outcomes are MAE and root mean squared error (RMSE), in rentals per hour.

Controlled fault injection

For each of three seeds and rates 1%, 5%, and 10%, select floor(rate × N) source rows without replacement. In separate scenarios, set selected hours to 25; increase selected cnt values by 100; or append selected original rows. Compare domain-only validation against domain plus reconciliation and duplicate-key validation. This yields 3 × 3 × 3 × 2 = 54 fault-policy evaluations. Row detection recall and false positives are measured against known injected positions. The underlying dataset is real; injected faults are synthetic stress tests.

4. Results

Predictive assessment

Split	Model	Mean MAE	MAE standard deviation	Mean RMSE
Chronological	Training mean	174.985	0.000	232.608
Chronological	Forest	45.687	0.168	70.723
Chronological	Leaking control - invalid	2.414	0.048	8.477
Random	Training mean	142.566	0.833	180.843
Random	Forest	26.034	0.406	44.067
Random	Leaking control - invalid	0.937	0.047	2.634

Values are means and sample standard deviations over three seeds. Chronological baseline variation is zero because its split and constant predictor do not change. These standard deviations are not confidence intervals. Random and chronological tests cover different periods/distributions, so their difference cannot be attributed solely to leakage or interpreted as a causal estimate. Nevertheless, the discrepancy demonstrates that reporting only a shuffled split would answer a materially different assessment question.

Fault detection

Across tested seeds and rates, domain-only recall is 1.0 for invalid hours and 0.0 for count mismatch or appended duplicates. Extended-policy recall is 1.0 for each injected fault type. Both policies have zero observed false positives on the unmodified records under these tests. These faults are explicitly covered by the rules, making successful detection a contract check, not evidence of an intelligent general anomaly detector.

Rejecting corrupted original rows can bias aggregates downward. Perfect detection does not reconstruct missing valid demand. Quarantine is an investigation mechanism, not an automatic correction policy. Raw per-run metrics, including aggregate differences, are committed in results/evaluation.json.

5. Validity, ethics, and limitations

This is one historical, already aggregated public dataset. It does not represent current ridership, station availability, or commercial production. Three seeds and one temporal holdout do not establish robust forecasting generalization. No significance test or best-model claim is made. Fault classes were selected to match explicit rules; unknown faults, adversarial errors, valid distribution shifts, and mixed faults are not evaluated. Range checks can miss plausible but wrong values. The data contain no named customer-level records, but public data should still be handled according to its license.

The in-memory parser and local SQLite/API design support inspection, not distributed scale. Cloud deployment, authentication, service latency guarantees, and real client adoption are outside the executed scope. Code-level tests and retrospective metrics are evidence of this implementation only.

6. Reproducibility and next work

Run python pipeline.py, python -m unittest -v, and python evaluate.py from projects/mobility-quality after installing its pinned dependencies. evaluation.json records dataset hash, versions, seeds, feature set, predictions' error metrics, and fault-policy outcomes. Six system tests include actual HTTP status checks and snapshot preservation. Exact runtimes vary by host.

The next defensible study should add rolling-origin evaluation across whole days, a seasonal-naive baseline with explicitly aligned lags, multiple datasets, policy sensitivity, and block-bootstrap uncertainty. Those experiments have not been performed here.

References

Breck, E., Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2019). Data validation for machine learning. Proceedings of the 2nd SysML Conference. https://mlsys.org/Conferences/2019/doc/2019/167.pdf

Fanaee-T, H. (2013). Bike Sharing [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5W894

Scikit-learn developers. (n.d.). Cross-validation: Evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html (accessed October 18, 2026).