IJC
International Journal of Computer (IJC)
Peer-Reviewed International Journal
ISSN: 2307-4523
https://ijcjournal.org

Predictive Analytics for Operational Intelligence: A Cloud-Based Framework for Equipment Downtime Prediction

Anil Kumar Kolla*, Ameya Kokate

Anil Kumar Kolla: Department of IT, HonorHealth, USA; Ameya Kokate: Department of Data Science and Analytics, HonorHealth, USA

* *Corresponding author: akeeejournals@gmail.com | Ameya Kokate: ameyakok96@gmail.com*

Accepted: 15 October 2026 | Published: 20 November 2026

Abstract

Unplanned equipment downtime imposes costs that extend well beyond repair: lost throughput, expedited logistics, schedule disruption, and quality variability. Predictive maintenance seeks to anticipate failure early enough to intervene, and machine learning applied to condition-monitoring and event-stream data is now a mature research area. Deployment, however, remains difficult, and the difficulties are largely architectural and organisational rather than algorithmic. This paper presents an end-to-end cloud-based framework for equipment downtime prediction spanning telemetry ingestion, stream and batch processing, feature construction, model training and serving, and decision support delivered through operational dashboards. The framework's contributions are: a labelling methodology that makes explicit the prediction horizon, censoring, and failure-definition choices that dominate reported performance and are frequently left unstated; a feature construction approach enforcing point-in-time correctness over event streams so that models are not trained on information unavailable at inference; a decision-layer specification that converts model scores into maintenance actions through cost-weighted thresholds rather than default probability cut-offs; and an evaluation protocol using prognostics-appropriate metrics together with an operational cost model. We describe implementation using managed event ingestion, distributed processing, and a transactional lakehouse. We further report a controlled simulation study which finds that filtering feature windows on event time rather than recording time overstates PR-AUC by approximately 27% relative to the same model's performance under deployment conditions, and that retraining on correctly filtered features does not recover the apparent loss, since the exploited information was never available at inference. We analyse limitations including label scarcity, the survivorship bias induced by successful preventive maintenance, and the feedback loop by which a deployed model degrades the very signal it depends upon.

Keywords: predictive maintenance; equipment downtime prediction; data leakage; machine learning; stream processing; operational intelligence; decision support

1. Introduction

Maintenance strategy is conventionally described in three tiers: reactive repair after failure, preventive maintenance on fixed schedules, and predictive maintenance based on observed condition. The economic argument for the third is straightforward — fixed schedules over-maintain healthy equipment and under-maintain degrading equipment — and the literature supporting it is extensive [1]–[3].

Yet deployment rates lag the research. The gap is instructive. Published studies typically operate on curated benchmark datasets with clean failure labels, complete run-to-failure trajectories, and a well-posed remaining-useful-life target [4]. Industrial reality differs on every count. Failures are rare; maintenance records are inconsistent and often free-text; equipment is repaired before failing, censoring the very trajectories a model needs; sensor coverage varies across an asset fleet; and the deployment

target is not a prediction but a decision — whether to schedule an intervention — made under a cost asymmetry that no accuracy metric captures.

This paper treats those conditions as the design problem rather than as noise. Contributions:

A labelling methodology making explicit the horizon, censoring, and failure-definition decisions that determine whether a reported result is meaningful.

A point-in-time-correct feature construction approach for event streams, addressing the leakage that arises when maintenance records are back-dated.

A decision layer translating scores to actions via cost-weighted thresholds, with alert budgeting to manage operator load.

An evaluation protocol combining prognostic metrics with an operational cost model and a deployment-degradation monitor.

Contribution type and scope. This paper combines a framework contribution with one controlled empirical result. Sections 3 to 6 specify the architecture, the labelling methodology, and the decision layer; Section 5II reports a controlled simulation study quantifying the optimism introduced by back-dated maintenance records, which is the framework's most consequential individual claim. The simulation uses synthetic data generated under a stated model, and it is a demonstration of a mechanism rather than a performance result on operational equipment. The remaining claims are specified for measurement in Section 5I but are not empirically evaluated here.

2. Background and Related Work

2.1. Predictive Maintenance

Jardine *et al.* [1] survey diagnostics and prognostics for condition-based maintenance, establishing the distinction between fault detection, diagnosis, and prognosis. Lee *et al.* [2] situate prognostics within cyber-physical manufacturing systems. Carvalho *et al.* [3] provide a systematic review of machine learning methods applied to predictive maintenance, documenting the prevalence of tree-ensemble and neural approaches and — importantly for this paper — the inconsistency of evaluation practice across the literature.

Susto et al. [5] proposed a multiple-classifier approach that explicitly incorporates the cost asymmetry between missed failures and unnecessary maintenance, which is the design stance adopted in Section 5. Zhang et al. [6] survey data-driven methods with attention to industrial deployment constraints.

Benchmark datasets, notably the C-MAPSS turbofan degradation data [4], have supported extensive methodological comparison. Their limitation for deployment guidance is that they contain complete run-to-failure trajectories under controlled conditions — precisely what industrial datasets lack.

2.2. Streaming Architectures

Event-driven telemetry architectures build on durable, partitioned logs [7], with stream processing frameworks providing windowed computation and event-time semantics [8], [9]. The event-time versus processing-time distinction formalised by Akidau *et al.* [8] is directly consequential here: industrial telemetry arrives late and out of order, and features computed on processing time will differ at inference from those computed on event time during training.

2.3. Modelling Approaches

Gradient-boosted trees [10] and random forests [11] remain strong baselines on tabular sensor-derived features and offer practical advantages in training cost and interpretability. Recurrent architectures, particularly long short-term memory (LSTM) [12], model temporal degradation directly and are common in remaining-useful-life estimation. Feature attribution methods [13] support the

explanation requirements that operational users impose — a maintenance planner asked to take equipment offline reasonably expects to know why.

2.4. Standards

Condition monitoring and diagnostics standards [14], [15] define an architectural decomposition of data acquisition, manipulation, state detection, health assessment, prognostics, and advisory generation, together with condition-monitoring programme requirements. The framework presented here maps onto this decomposition and adopts its terminology where applicable.

2.5. Gap

The literature is rich in modelling methods and comparatively sparse in deployment architecture, labelling practice, and decision-layer design. Where reported results omit the labelling decisions, they are not comparable and arguably not interpretable. This paper foregrounds those decisions.

3. Framework Architecture

3.1. Ingestion Layer

Equipment telemetry, controller events, and machine state transitions are published to a managed, partitioned event ingestion service providing durability and replay. Partitioning by asset identifier preserves per-asset ordering, which matters because degradation features are inherently sequential.

Three additional source classes are integrated: the maintenance management system, supplying work orders and their completion records; the manufacturing or operations execution system, supplying production context such as load, product type, and shift; and static asset master data, supplying make, model, installation date, and location.

The maintenance management system is the label source and is therefore the most consequential and least reliable input. Its records are entered by technicians after the fact, are frequently back-dated, and frequently describe the intervention rather than the fault. Section 4 addresses the consequences.

3.2. Processing Layer

Streaming computation produces low-latency health indicators — rolling statistics, threshold excursions, state-duration features — on event time with a bounded lateness allowance and explicit watermarking [8]. Batch computation produces the training feature store over historical data using identical logic. Maintaining a single logical definition for streaming and batch feature computation is essential; divergence between the two is a common and difficult-to-detect source of training–serving skew.

Data lands in a layered lakehouse [16], [17]: raw immutable telemetry, conformed per-asset time series with unit normalisation and sensor identity resolution, and curated feature and label tables. Metadata-driven approaches to the raw-to-conformed transition, in which column mappings and declarative data-quality rules are held in a central metadata repository and records failing validation are quarantined rather than dropped or allowed to fail the pipeline [18], are directly applicable to this layer; quarantine is particularly relevant for industrial telemetry, where sensor dropouts and malformed readings are routine and should not halt ingestion.

3.3. Feature Construction

Features fall into four families: (i) statistical aggregates of sensor signals over multiple windows; (ii) trend and change-point features capturing degradation direction; (iii) operational context features including cumulative load, duty cycle, and time since last intervention; and (iv) event-derived features such as fault-code frequencies and state-transition rates.

Point-in-time correctness is enforced by constructing every feature vector using only records whose *recording* timestamp precedes the reference time — not their *event* timestamp. This distinction is the framework's most important implementation detail. A work order describing a fault may be entered days after the event it describes and back-dated; a naive event-time filter admits it into the feature window, producing a model that appears to predict failures it is in fact observing. Every feature table therefore carries both timestamps, and training uses recording time.

3.4. Modelling Layer

The framework supports two complementary formulations:

Classification: will this asset fail within horizon H? Directly actionable and tolerant of censored data.

Survival or remaining-useful-life regression: how much operating time remains? More informative, but requires either complete trajectories or explicit censoring treatment.

The classification formulation is recommended as the initial deployment target because it degrades more gracefully under label scarcity and because its output maps more directly onto the maintenance scheduling decision.

Training uses temporal splitting — train on earlier periods, validate on later — and grouping by asset so that no asset appears in both training and validation. Random splitting on time-series-derived features produces optimistic results that do not survive deployment, and grouped temporal splitting is a minimum standard for credible reporting.

3.5. Serving and Decision Layer

Scores are produced on a schedule aligned to maintenance planning cadence rather than continuously; sub-minute latency confers no benefit when the resulting action is scheduled days ahead. Scores, feature vectors, model version, and the resulting decision are persisted for audit and for subsequent evaluation.

3.6. Decision Support Layer

Outputs surface in an operational dashboard oriented to the planner's workflow: a ranked risk view across the fleet, per-asset detail with contributing factors [13] and recent signal history, and integration that permits work-order creation from the interface. A prediction that requires the planner to leave the tool to act on it will not be acted upon. Figure 1 summarizes the end-to-end framework.

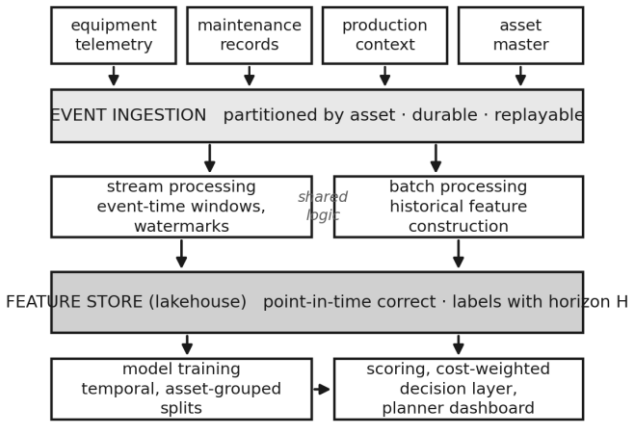


Figure 1. End-to-end framework. Streaming and batch paths share a single logical feature definition to avoid training–serving skew.

4. Labelling Methodology

Reported predictive maintenance performance is dominated by labelling choices. We specify four that must be stated explicitly.

1. Failure definition. What constitutes the positive event: unplanned stoppage exceeding a duration threshold, a specific fault code, a corrective work order of a defined type, or a quality excursion. Different definitions produce different base rates and non-comparable results. Where free-text work-order descriptions are the only source, the classification rules applied to them must be documented, as they materially affect the label set.

2. Prediction horizon and lead time. The horizon *H* over which failure is predicted, and the minimum actionable lead time *L* below which a correct prediction is operationally useless. Predictions made within *L* of failure should be scored as failures to predict, not as successes. Omitting *L* is the most common way that reported performance overstates operational value.

3. Censoring treatment. Assets undergoing successful preventive maintenance never fail, and their trajectories are right-censored. Treating them as negative examples teaches the model that pre-failure degradation signatures are benign — a survivorship bias that is severe precisely in organisations with effective existing maintenance programmes. Censored trajectories must be handled explicitly, either by exclusion of the post-intervention window or by survival methods that accommodate censoring.

4. Intervention exclusion window. Records within a defined window following any maintenance intervention are excluded from both training and inference, since asset behaviour immediately post-intervention is not representative and often includes commissioning anomalies that resemble faults. Figure 2 summarizes the labeling scheme.

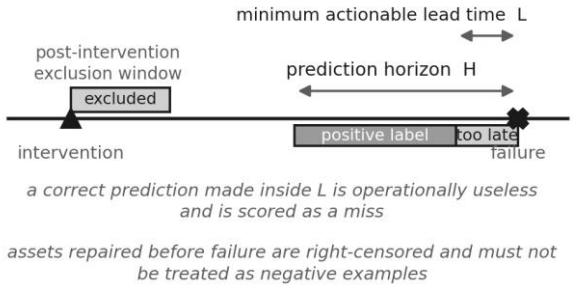


Figure 2. Labelling scheme showing the prediction horizon, the minimum actionable lead time, the post-intervention exclusion window, and right-censoring.

5. Decision Layer and Cost Model

A model produces a score; an organisation needs an action. Converting one to the other by a default 0.5 threshold implicitly asserts that false positives and false negatives cost the same, which is essentially never true.

Let *C_FN* be the expected cost of an unanticipated failure (lost production, expedited repair, secondary damage, schedule disruption), *C_FP* the cost of an unnecessary intervention (labour, parts, planned downtime), and *C_TP* the cost of a successful anticipated intervention. The decision threshold minimising expected cost follows from the ratio of these quantities rather than from any property of the model. Because *C_FN* typically exceeds *C_FP* substantially, the optimal operating point sits at a

considerably lower threshold — and considerably lower precision — than accuracy-maximising selection would choose. Reporting a model at its accuracy-optimal threshold therefore misrepresents its operational configuration.

Two practical constraints qualify this. First, alert budgeting: maintenance capacity is finite, and an alert volume exceeding it produces triage failure and alert fatigue, after which the effective response rate falls regardless of model quality. The threshold should be constrained so that expected alert volume remains within capacity, and this constraint frequently binds before the cost-optimal point is reached. Second, cost heterogeneity: C_FN varies by asset criticality and production schedule, so thresholds should be set per asset class rather than globally. Figure 3 illustrates the threshold-cost relationship schematically.

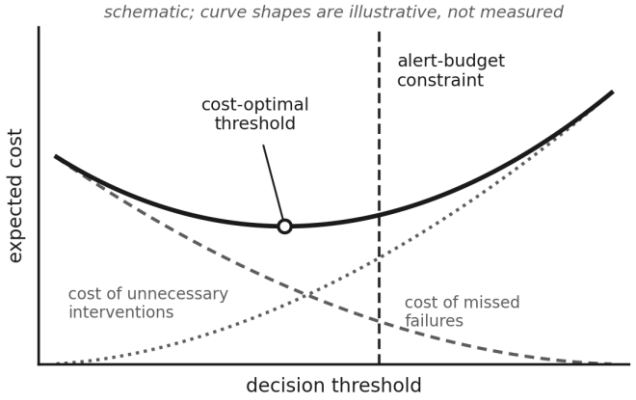


Figure 3. Schematic relationship between decision threshold and expected cost. Curve shapes are illustrative; the operating point must be derived from measured costs.

6. Evaluation Protocol

Implementers should report the following on their own data. *This paper specifies the measurements, not the results.*

Predictive performance. Precision, recall, and precision-recall area under the curve (PR-AUC) at the deployed operating threshold, reported with the class base rate. receiver operating characteristic area under the curve (ROC AUC) alone is inadequate under severe class imbalance and should not be reported in isolation. For remaining-useful-life formulations, report both symmetric root mean squared error (RMSE) and an asymmetric prognostic score penalising late predictions more heavily than early ones, consistent with established prognostics practice [4].

Timeliness. Distribution of lead time on true positives, with the proportion falling below the actionable threshold L reported explicitly.

Operational. Change in unplanned downtime hours; change in preventive intervention count; alert volume against maintenance capacity; planner action rate on alerts; and the proportion of acted-upon alerts where degradation was confirmed on inspection. The last is the closest available proxy for real-world precision and is more informative than any offline metric.

Economic. Net benefit under the Section 5 cost model, with sensitivity analysis over the cost parameters, which are usually estimated rather than measured.

Stability. Feature and prediction drift over time; calibration drift; and performance stratified by asset class, age, and sensor coverage, since fleet-level averages conceal that models often work only for well-instrumented recent assets.

Baselines. Comparison against the incumbent preventive schedule and against simple threshold-based condition alarms. The second baseline is frequently omitted and frequently competitive; a machine learning system that does not beat a well-tuned threshold alarm has not justified its maintenance burden.

7. Controlled Simulation Study of Documentation-Lag Leakage

Section 3.3 claims that filtering feature windows on event time rather than recording time admits back-dated maintenance documentation and inflates apparent performance. This section quantifies that effect under controlled conditions. The study uses synthetic data; it demonstrates a mechanism and its magnitude under a stated generative model, and it is not a performance result for any real asset fleet.

7.1. Generative Model

A fleet of 600 assets is simulated over 540 days. Each asset accumulates latent degradation at an asset-specific rate and fails when degradation crosses an asset-specific threshold, after which it is repaired and the threshold is redrawn. A sensor observes degradation with additive noise. Two to ten days before each failure, a precursor fault is noticed and written up as a work order with probability 0.8; the work order carries the day the fault was noticed as its event time, but is entered into the maintenance system two to nine days later. Routine, non-predictive work orders are added as background. Because the precursor is only partially reflected in the sensor stream, the work order carries information the telemetry does not — which is precisely the condition under which the filtering choice matters.

7.2. Procedure

Two feature sets are constructed that are identical in every respect except the timestamp used to gate work-order visibility: the naive set admits a work order from its event time, the correct set from its recording time. Features comprise rolling sensor means and standard deviations over 7, 30, and 60 days, a 30-day slope, a short-minus-long delta, cumulative load, and two work-order features (count and recency within 14 days). The label is failure within a 14-day horizon. Training and test sets are split both temporally, at day 378, and by asset, with no asset appearing in both. A gradient-boosted tree classifier is fitted with identical hyperparameters in all configurations. Three configurations are compared:

7.1. naive features in training and in evaluation — the configuration an unwary practitioner produces.

7.2. the model trained in A, evaluated on recording-time features — what that model actually achieves once deployed, since at inference the back-dated record has not yet been entered.

7.3. recording-time features in training and evaluation throughout.

The experiment is repeated across five random seeds. Table 1 reports the resulting metrics.

7.3. Results

Table 1. Simulation results over five seeds. The positive base rate is 0.057. Values are from synthetic data generated as described in Section 7.1.

Configuration	PR-AUC (mean)	PR-AUC (min–max)	Precision at 60% recall
A. naive features, naive evaluation	0.395	0.335 – 0.435	0.308
B. naive-trained model, deployment conditions	0.311	0.255 – 0.348	0.264
C. recording-time features throughout	0.302	0.244 – 0.346	0.258

The naive configuration overstates PR-AUC by approximately 27% relative to what the same model achieves under deployment conditions, and the gap is consistent in sign across all five seeds. Expressed in the terms a maintenance planner would use, the optimism is not a rounding difference: it is the

difference between a model that appears to justify deployment and one that may not. Figure 4 summarizes the five-seed comparison.

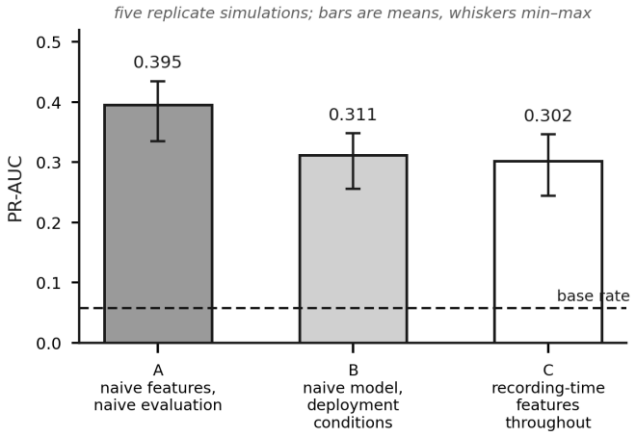


Figure 4. Simulation results over five seeds. Configuration A is the measurement an unwary evaluation produces; B is what the same model achieves once deployed. Synthetic data, generated as described in Section 5II-A.

The second observation is less comfortable and is reported because it is what the data show. Configuration C does not outperform B; the two are within run-to-run variation. Retraining on correctly filtered features did not recover the apparent performance, because the information the naive model exploited was never available at inference time and no amount of retraining creates it. The practical implication is that point-in-time correctness is a matter of honest measurement rather than a technique for improving models. Its value is that it prevents a project from proceeding on a false premise — which is worth more than a modest accuracy gain, but is a different kind of benefit and should not be advertised as the former.

7.4. Threats to Validity

The generative model is stipulated rather than estimated from operational data, and the magnitude of the effect depends directly on two stipulated quantities: the proportion of failures preceded by a written-up precursor, and the amount of information in that precursor that is absent from the telemetry. Both are properties of a particular organisation's documentation practice. The study therefore establishes that the effect exists and can be large, not that it takes this magnitude anywhere in particular. Replication on operational maintenance records, where the documentation lag distribution can be measured directly rather than assumed, is the necessary next step and is not claimed here.

8. Discussion and Limitations

More broadly, the deployment-side concerns discussed here are consistent with the hidden technical-debt perspective on machine-learning systems [19]. Label scarcity is the binding constraint. Failures are rare by design. Most industrial datasets contain too few positive examples for the model complexity researchers prefer, and this — not algorithm selection — determines achievable performance. Transfer across similar assets and semi-supervised approaches are the plausible mitigations.

Deployment degrades its own signal. A successful model prevents failures, removing positive examples from subsequent training data. Over time the model is retrained on data from which its own successes have been erased, and measured performance declines even as operational value is realised. Evaluation must therefore track prevented failures — alerts acted upon with degradation confirmed — as positive outcomes rather than treating the absence of failure as evidence of a false positive. Without this, an effective deployment will appear to fail.

Maintenance record quality bounds everything. Labels derived from inconsistent free-text records carry irreducible noise. Investment in structured fault coding at the point of work-order closure typically yields more improvement than any modelling change, and should be assessed as a first-order intervention rather than a data-hygiene afterthought.

Sensor coverage heterogeneity. Fleets mix instrumentation generations. Models trained on well-instrumented assets do not transfer to sparsely instrumented ones, and fleet-average reporting conceals this. Stratified reporting is required.

Explanation is a requirement, not a preference. Planners will not remove equipment from production on an unexplained score. Attribution methods [13] provide contributing-factor views, but their fidelity as explanations is itself limited and should not be overstated to operational users.

Generalisation. Results are specific to asset class, operating context, and instrumentation. Cross-site transfer should be treated as a research question in each instance rather than assumed.

9. Conclusion

This paper presented a cloud-based framework for equipment downtime prediction spanning ingestion, stream and batch processing, point-in-time-correct feature construction, modelling, cost-weighted decision-making, and operational decision support. It also reported a controlled simulation quantifying one of its claims: that documentation lag in maintenance records inflates measured performance, and that the inflation is an artefact of evaluation rather than a recoverable modelling deficit. The paper's broader argument is that the difficulty of deploying predictive maintenance lies predominantly outside the model: in labelling decisions that determine whether reported performance means anything, in leakage arising from back-dated maintenance records, in the cost asymmetry that governs the operating point, and in the feedback loop by which successful prevention erases the evidence of its own value. The framework makes each of these explicit and measurable.

Future work should address censoring-aware learning under heavy preventive maintenance regimes, transfer learning across heterogeneously instrumented fleets, and joint optimisation of prediction and maintenance scheduling, where the decision problem is treated as constrained resource allocation over the fleet rather than as independent per-asset thresholding.

REFERENCES

[1] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006.

[2] J. Lee, F. Wu, W. Zhao, M. Ghaffari, L. Liao, and D. Siegel, "Prognostics and health management design for rotary machinery systems — Reviews, methodology and applications," *Mechanical Systems and Signal Processing*, vol. 42, no. 1–2, pp. 314–334, 2014.

[3] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, art. 106024, 2019.

[4] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. Int. Conf. Prognostics and Health Management (PHM)*, 2008, pp. 1–9.

[5] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine learning for predictive maintenance: A multiple classifier approach," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812–820, 2015.

[6] W. Zhang, D. Yang, and H. Wang, "Data-driven methods for predictive maintenance of industrial equipment: A survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213–2227, 2019.

[7] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proc. NetDB Workshop*, 2011.

[8] T. Akidau *et al.*, "The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1792–1803, 2015.

[9] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache Flink: Stream and batch processing in a single engine," *IEEE Data Engineering Bulletin*, vol. 38, no. 4, pp. 28–38, 2015.

[10] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 4765–4774.

[14] *Condition Monitoring and Diagnostics of Machines — Data Processing, Communication and Presentation — Part 1: General Guidelines*, ISO 13374-1:2003, International Organization for Standardization, Geneva, Switzerland, 2003.

[15] *Condition Monitoring and Diagnostics of Machines — General Guidelines*, ISO 17359:2018, International Organization for Standardization, Geneva, Switzerland, 2018.

[16] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics," in *Proc. Conf. Innovative Data Systems Research (CIDR)*, 2021.

[17] M. Armbrust *et al.*, "Delta Lake: High-performance ACID table storage over cloud object stores," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 3411–3424, 2020.

[18] A. K. Kolla, A. Kokate, R. Vellanki, R. Lolam, and P. Bandi, "A metadata-driven framework for automated bronze-to-silver data ingestion, validation and loading on Databricks," *European Journal of Information Technologies and Computer Science*, vol. 6, no. 3, pp. 1–7, 2026, doi: 10.24018/compute.2026.6.3.70578.

[19] D. Sculley *et al.*, "Hidden technical debt in machine learning systems," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 2503–2511.