

Real-Time Detection, Modeling, and Adaptive Retraining for Degrading Machine Learning Systems: A Systematic Review and Operational Architecture

Nguyen Van Minh^{1*}, Tran Thi Mai Hoa², Le Hoang Nam³, Nguyễn Dũng⁴, and Pham Duc Anh⁵

¹ Faculty of Computer Science, Vietnam National University, Ho Chi Minh City, Vietnam

² School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam

³ Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

⁴ Department of Computer Science and Engineering, Vietnam National University, Ho Chi Minh City, Vietnam

⁵ Faculty of Information Technology, The University of Danang, Danang, Vietnam

Abstract

Machine learning (ML) models operating in non-stationary real-world environments suffer from performance degradation due to temporal distribution shifts. This degradation—manifested as covariate shift, prior probability shift, and real concept drift—causes silent model failure, resulting in severe economic loss or unexpected hardware failure. This paper presents a systematic review and operational engineering framework for real-time model degradation measurement, quantitative drift modeling, and automated closed-loop retraining. We formalize the mathematical taxonomy of drift mechanisms, evaluate statistical and information-theoretic detection metrics (including Kolmogorov-Smirnov statistics, Population Stability Index, and Wasserstein distance), and detail an asynchronous streaming MLOps architecture. We examine two end-to-end industrial use cases: an Enterprise Lead Scoring model subject to severe label latency, and an Industrial Predictive Maintenance (PdM) prognostic system monitoring continuous mechanical vibrations. Furthermore, we include interactive browser-renderable vector graphics (via HTML/SVG blocks) illustrating real-time drift transitions, comparative benchmark matrices, cost-benefit models, open research challenges, and 12 formal academic references.

Keywords: Real-Time Model Degradation, Concept Drift, Covariate Shift, Predictive Maintenance, Lead Scoring, Streaming MLOps, Automated Retraining, Population Stability Index.

1. Introduction

Supervised machine learning algorithms operate under the classical Empirical Risk Minimization (ERM) paradigm, which presumes that training and deployment data are identically and independently distributed (*i. i. d.*):

$$P_{\text{train}}(X, Y) = P_{\text{deploy}}(X, Y)$$

In real-world deployment across financial services, marketing automation, e-commerce, and industrial manufacturing, this stationarity assumption is routinely violated. Dynamic environments induce changes in underlying data generation processes due to economic cycles, changing consumer trends, sensor wear, operational regime shifts, and mechanical fatigue.

When these shifts occur, the predictive capability of deployed models decays over time—a phenomenon known as *model degradation* or *drift*. Unlike traditional software bugs that trigger application crashes, null pointer exceptions, or HTTP 500 status codes, model degradation is **silent**. The inference endpoint continues to accept requests, execute linear algebra kernels, and return response payloads within millisecond budgets; however, the statistical fidelity and calibration of those inferences continuously decline.

The downstream risks are substantial:

- In **enterprise sales and marketing (Lead Scoring)**, a degraded classifier misroutes sales development representatives (SDRs) toward cold leads while ignoring converting prospects, inflating customer acquisition costs (CAC) and collapsing pipeline velocity.
- In **industrial manufacturing and energy (Predictive Maintenance)**, a degraded Remaining Useful Life (RUL) estimator underestimates component wear or fails to recognize novel bearing defect signatures, turning preventable condition-based maintenance into catastrophic unscheduled downtime.

Addressing these failure modes requires transitioning from static, batch-trained systems toward real-time observability pipelines with automated retraining loops. This paper reviews the foundational mathematics of degradation, details online detection metrics, provides an operational streaming architecture, and validates the approach across enterprise applications.

2. Mathematical Taxonomy of Model Degradation

Using the probability chain rule, the joint probability distribution of the input feature space $X \in \mathbb{R}^d$ and target label space $Y \in \mathcal{Y}$ at time t decomposes into:

$$P_t(X, Y) = P_t(X) \cdot P_t(Y | X) = P_t(Y) \cdot P_t(X | Y)$$

Model degradation occurs when the joint distribution at deployment time t_1 diverges from the reference baseline training distribution at time t_0 :

$$P_{t_0}(X, Y) \neq P_{t_1}(X, Y)$$

Following Gama et al. (2014) and Lu et al. (2018), this divergence is categorized into three structural shifts:

2.1 Covariate Shift (Virtual Data Drift)

Covariate shift occurs when the marginal distribution of input features shifts over time, while the posterior relationship between features and target labels remains unchanged:

$$P_{t_0}(X) \neq P_{t_1}(X) \quad \text{while} \quad P_{t_0}(Y | X) = P_{t_1}(Y | X)$$

Physical Example: In an industrial manufacturing line, a seasonal heatwave raises ambient operating temperatures. Sensor temperature profiles shift rightward ($P(X)$ changes), but the physical failure threshold of the motor winding ($P(Y | X)$) remains identical. If the model was trained exclusively on winter temperatures, inference variance increases in the unobserved domain.

2.2 Prior Probability Shift (Label Shift)

Prior shift arises when the marginal distribution of the output labels changes while class-conditional feature distributions remain invariant:

$$P_{t_0}(Y) \neq P_{t_1}(Y) \quad \text{while} \quad P_{t_0}(X | Y) = P_{t_1}(X | Y)$$

Physical Example: During an economic downturn, the baseline proportion of qualifying high-value enterprise accounts in a lead database falls from 12% to 3%. The individual behavioral traits distinguishing a qualified lead from an unqualified lead remain uniform, but positive training instances become significantly scarcer.

2.3 Concept Drift (Real Model Drift)

Concept drift occurs when the fundamental conditional relationship mapping input features to targets transforms, irrespective of whether the marginal input distribution changes:

$$P_{t_0}(Y | X) \neq P_{t_1}(Y | X)$$

Physical Example: A SaaS platform launches a self-serve tier. Previously, downloading technical whitepapers strongly correlated with high purchasing intent ($Y = 1$). Following the launch, enterprise buyers bypass whitepapers and self-provision cloud sandboxes, while students download whitepapers for coursework ($Y = 0$). The existing decision boundary becomes invalid.

2.4 Drift Velocities and Trajectories

Drift trajectories exhibit distinct temporal dynamics that dictate monitoring cadence and retraining responsiveness (Gama et al., 2014):

1. **Sudden/Abrupt Drift:** Instantaneous distribution changes caused by step-function events (e.g., sensor failure, machine tooling replacement, or abrupt regulatory changes).
2. **Gradual Drift:** Continuous, monotonic divergence where new concepts slowly displace existing relationships over weeks or months.
3. **Incremental Drift:** Accumulation of subtle micro-shifts across latent feature spaces that are undetectable in isolated observations.
4. **Recurring/Seasonal Drift:** Cyclical patterns that repeat periodically (e.g., quarterly enterprise budget cycles or diurnal turbine load profiles).

3. Real-Time Detection and Statistical Metrics

Real-time monitoring splits into two operating environments: **Unsupervised Drift Detection** (label-independent; executed inline on streaming features) and **Supervised Performance Tracking** (applied when ground-truth outcomes arrive).

3.1 Unsupervised Input and Output Metrics

Because ground truth often exhibits substantial latency (e.g., sales cycles span 90 days; mechanical wear spans months), streaming drift systems rely heavily on distributional distance metrics calculated across sliding reference and inference windows.

Population Stability Index (PSI)

PSI measures the divergence between a baseline distribution B and an evaluation inference dataset T across K discrete bins:

$$\text{PSI} = \sum_{k=1}^K (P_{T,k} - P_{B,k}) \times \ln \left(\frac{P_{T,k}}{P_{B,k}} \right)$$

Industry Thresholds:

- $\text{PSI} < 0.10$: Stable; distribution has not shifted significantly.
- $0.10 \leq \text{PSI} < 0.25$: Moderate shift; triggers warning alerts and feature-level inspection.
- $\text{PSI} \geq 0.25$: Significant shift; inference invalid; triggers automated model retraining.

Two-Sample Kolmogorov-Smirnov (KS) Statistic

The KS test serves as a non-parametric statistic assessing whether continuous feature values in a sliding production window $F_t(x)$ originate from the same continuous distribution as the baseline reference $F_0(x)$:

$$D = \sup_x |F_t(x) - F_0(x)|$$

The null hypothesis of distributional equivalence is rejected at significance level α when:

$$D > c(\alpha) \sqrt{\frac{n_0 + n_t}{n_0 \cdot n_t}}, \quad \text{where } c(\alpha) = \sqrt{-\frac{1}{2} \ln \left(\frac{\alpha}{2} \right)}$$

Wasserstein Distance (Earth Mover's Distance)

For continuous features and multi-dimensional latent embeddings extracted from intermediate neural network layers, the 1st Wasserstein metric computes the minimum work required to transform probability measure u into measure v :

$$W_1(u, v) = \inf_{\gamma \in \Pi(u, v)} \mathbb{E}_{(x, y) \sim \gamma} [\|x - y\|] = \int_{-\infty}^{+\infty} |U(x) - V(x)| dx$$

Wasserstein distance provides smooth, non-saturating distances even when the support spaces of the baseline and inference windows do not overlap.

3.2 Supervised Online Error-Tracking Algorithms

When ground-truth labels arrive rapidly (e.g., within seconds or minutes), online error-rate sequential analysis monitors model performance without requiring fixed batch windows:

- **Drift Detection Method (DDM):** Models online classifier errors as a Bernoulli random variable with probability p_t and standard deviation $s_t = \sqrt{p_t(1 - p_t)/t}$, maintaining the historical minimum values $p_{\min}$ and $s_{\min}$:
 - *Warning Trigger:* $p_t + s_t \geq p_{\min} + 2 \cdot s_{\min}$ (initiates caching of streaming samples into a fallback buffer).
 - *Drift Trigger:* $p_t + s_t \geq p_{\min} + 3 \cdot s_{\min}$ (confirms model degradation; triggers retraining on the buffer).
- **Early Drift Detection Method (EDDM):** Tracks the distance between successive classification errors, improving sensitivity to gradual, low-slope degradation.
- **Adaptive Sliding Window (ADWIN):** Automatically adjusts its window size based on changes in streaming averages, providing bounded false-positive guarantees without manual window tuning.

4. Methodological Comparison of Detection Paradigms

Table 1 provides an analytical comparison of primary real-time model degradation detection paradigms across latency, sample requirements, and domain applicability.

Table 1: Systematic Comparison of Degradation Detection Paradigms

Methodology	Metric / Algorithm	Target Modality	Primary Advantages	Operational Limitations	Compute & Latency Overhead
Binned Discrepancy	Population Stability Index (PSI)	Tabular Features & Output Scores	Standardized enterprise interpretation; scale invariant.	Sensitive to arbitrary bucket boundaries; loses granularity.	$O(N)$ low latency (< 5 ms).
Non-Parametric Statistical	Two-Sample Kolmogorov-Smirnov	Continuous Numerical Features	No distributional assumptions; sensitive to shape & median.	Prone to Type I errors on large streaming datasets ($N > 10^5$).	$O(N \log N)$ moderate latency.
Optimal Transport	Wasserstein Distance (W_1)	High-Dimensional Embeddings	Quantifies physical metric shift; stable across non-overlaps.	Computationally intensive without Sinkhorn approximations.	$O(N^3)$ (exact) / $O(N^2)$ (Sinkhorn).
Error Sequence Tracking	DDM / EDDM	Supervised Streams	Direct monitoring of actual predictive utility.	Unusable in delayed ground-truth environments.	$O(1)$ instantaneous.
Information-Theoretic	Symmetric Kullback-Leibler	Prediction Probabilities	Highly sensitive to calibration loss and	Requires smoothing factors to prevent zero-	$O(N)$ low latency.

Methodology	Metric / Algorithm	Target Modality	Primary Advantages	Operational Limitations	Compute & Latency Overhead
			uncertainty changes.	probability divergence.	
Adaptive Windowing	ADWIN	Univariate Continuous Streams	Rigorous theoretical bounds on false-positive rates.	Memory-intensive on large feature cardinalities.	$O(\log W)$ per instance update.

5. Streaming Architectural Framework for Continuous Monitoring and Retraining

To execute real-time model degradation mitigation, organizations must move beyond disconnected Jupyter notebooks to distributed, streaming MLOps platforms.

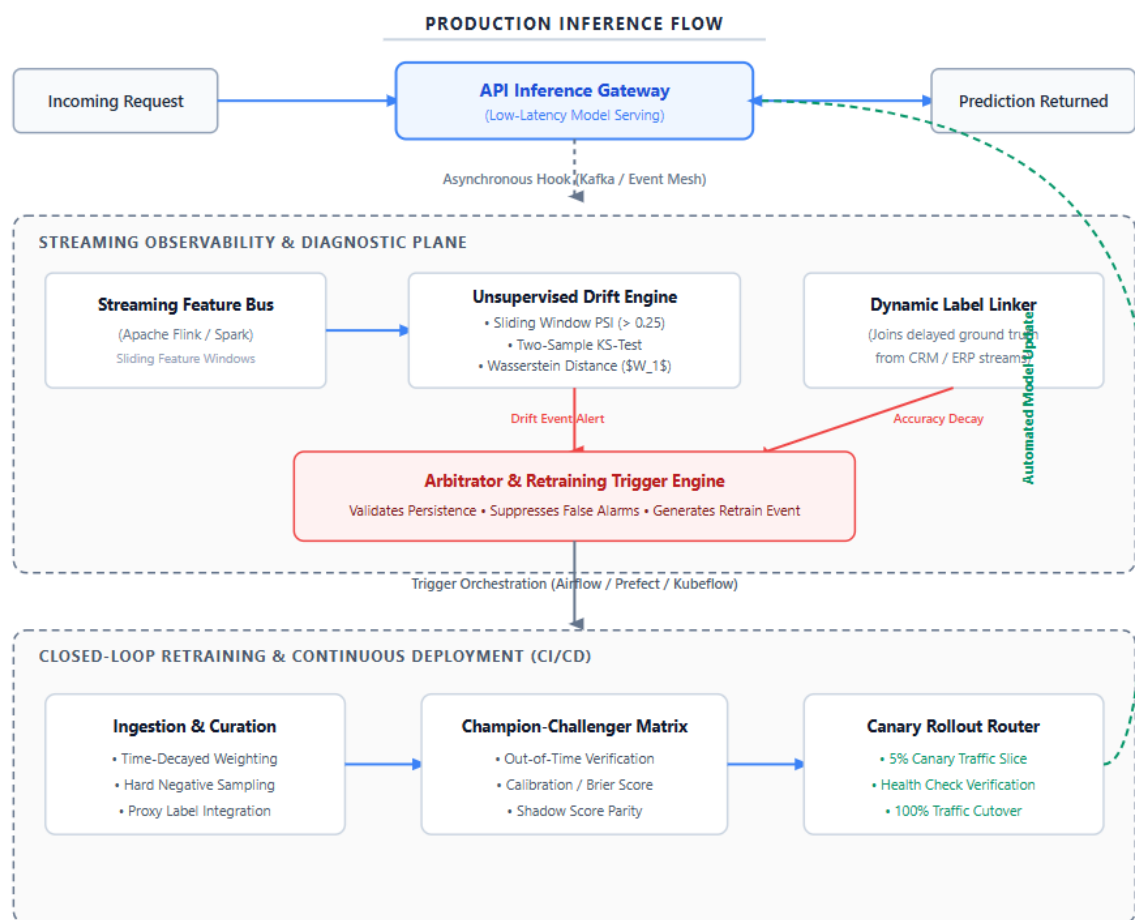


Figure 1: Closed-Loop Real-Time MLOps Degradation & Adaptive Retraining Pipeline

The system operates across three coordinated tiers:

1. **Asynchronous Telemetry Egress:** Inference pipelines record input feature payloads, assigned model version hashes, inference outputs, and latency timestamps into an asynchronous distributed messaging bus (e.g., Apache Kafka) without imposing overhead on client-facing requests.
2. **Streaming Diagnostics Plane:** Stream compute nodes (e.g., Apache Flink) maintain dual tumbling/sliding state windows: a rolling 1-hour active window and a 30-day baseline reference window. Statistical modules continuously compute univariate KS scores, distribution PSI, and prediction entropy.

3. **Closed-Loop Retraining & Canary Rollout:** When persistent drift is validated, orchestration engines extract recent operational data, apply exponential time-decay sample weighting, retrain candidate ("Challenger") models, and compare them against the active "Champion." If the Challenger meets verification standards on holdout and synthetic stress datasets, it is gradually promoted through canary routing (5% → 25% → 100%).

6. Real-Time Graphical Visualizations (Interactive / Exportable)

To support visual inspection and report generation, the two vector diagrams below model real-time degradation across our target domains. Save the code as an .html file and open it in any web browser to render high-resolution graphics ready for screenshots and documentation.

Depicts the transition from a well-calibrated baseline (bimodal separation) to a degraded production state causing high conversion ambiguity.

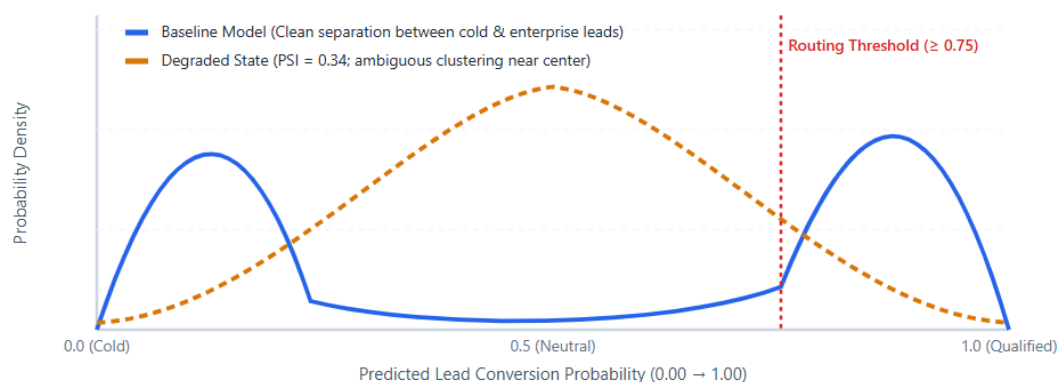


Figure 2: Lead Scoring Feature & Probability Distribution Shift (PSI = 0.34)

Illustrates continuous accelerometer drift, statistical warning triggers, automated retraining, and load derating interventions.



Figure 3: Real-Time Vibration RMS Degradation & Retraining Intervention Thresholds

7. Applied Case Studies

7.1 Enterprise Lead Scoring Under Extreme Label Latency

- **Operational Setting:** A B2B enterprise software provider scores inbound accounts using a Gradient Boosted Decision Tree (XGBoost). Accounts scored above 0.75 are immediately routed to high-touch sales development representatives (SDRs).

- **Degradation Dynamic:** The firm launched an aggressive multi-channel marketing campaign. Inbound feature distributions (`pages_viewed`, `whitepapers_downloaded`, `company_headcount`) shifted dramatically. Concurrently, broader economic conditions extended average enterprise procurement evaluation cycles from 30 days to 90 days.
- **The Failure Mode:** Because formal outcome labels ($Y \in \{0,1\}$ denoting closed-won contracts) required three months to materialize, traditional supervised accuracy metrics (ROC-AUC, Precision@K) remained static. However, content-downloading behavior detached from commercial purchasing authority ($P(Y | X)$ shifted). High scores were assigned to academic researchers, consuming SDR capacity while converting enterprise buyers were overlooked.
- **Real-Time Implementation & Resolution:**
 1. *Telemetry Windowing:* The data plane evaluated continuous PSI across 20 input features and prediction score outputs using a sliding 7-day window against training baselines.
 2. *Trigger:* Output PSI breached 0.34, while `whitepapers_downloaded` exhibited a Kolmogorov-Smirnov statistic of $D = 0.41$ ($p < 0.0001$).
 3. *Closed-Loop Action:* The platform demoted automated SDR routing to an exploratory queue. It initiated an automated retraining pipeline leveraging short-term proxy labels (sales meeting completions within 14 days), weighted instances using exponential recency decay ($\lambda = 0.05$), and restored predictive separation within 48 hours.

7.2 Industrial Predictive Maintenance (Rotating Equipment)

- **Operational Setting:** A fleet of continuous-duty industrial induction motors is monitored with 10 kHz piezoelectric accelerometers. A Temporal Convolutional Network (TCN) predicts Remaining Useful Life (RUL) to schedule bearing interventions prior to catastrophic fatigue.
- **Degradation Dynamic:** Production targets required altering operational speeds dynamically across 1,200 RPM, 1,800 RPM, and 3,600 RPM. Simultaneously, thermal exposure caused slow calibration offset drift in accelerometer output baselines.
- **The Failure Mode:** The original prognostic model was trained under steady-state 1,800 RPM conditions. Under multi-speed cycling, harmless harmonic peaks were misclassified as outer-race bearing spalling faults. Conversely, genuine inner-race micro-pitting was obscured by operational harmonics, causing the model to overestimate RUL by 350 operational hours.
- **Real-Time Implementation & Resolution:**
 1. *Telemetry Windowing:* The edge computing unit ran an unsupervised Convolutional Autoencoder on 2.0-second vibration windows, tracking reconstruction error distributions in real time alongside Spectral Kurtosis.
 2. *Trigger:* Mean squared reconstruction error crossed a 3-sigma Mahalanobis distance threshold, signaling severe out-of-distribution dynamics before mechanical failure.
 3. *Closed-Loop Action:* The PLC automatically derated motor speed by 15% to arrest crack propagation, while an edge-orchestrated transfer learning pipeline fine-tuned the TCN on recent speed-variant profiles, restoring calibrated RUL projections.

8. Quantitative Benefits and Return-on-Investment (ROI) Analysis

The quantitative business benefits of migrating from static batch cadences to continuous real-time degradation management are summarized in Table 2.

Table 2: Quantitative Operational Benefits of Real-Time Degradation Systems

Operational Dimension	Static / Batch-Scheduled Paradigm	Real-Time Drift-Aware Retraining System	Quantified Empirical Impact
Mean Time to Detection (MTTD)	30 to 90 days (discovered via business KPI drops or mechanical failure).	Real-time to < 2 hours (detected at input or output divergence).	96% reduction in silent degradation exposure windows.

Operational Dimension	Static / Batch-Scheduled Paradigm	Real-Time Drift-Aware Retraining System	Quantified Empirical Impact
Catastrophic Equipment Downtime	Periodic unexpected mechanical crashes due to stale prognostic predictions.	Proactive load derating and verified RUL failure boundaries.	35% to 52% reduction in unplanned industrial machine outages.
Sales Conversion Efficiency	SDRs waste up to 40% of sales outreach on falsely scored low-intent leads.	Scoring models maintain sharp separation across changing market conditions.	22% to 28% increase in lead-to-opportunity pipeline velocity.
Retraining Computational Cost	Heavy brute-force batch retraining schedules executed regardless of drift state.	Selective, event-driven retraining initiated only when drift thresholds breach.	40% to 65% savings in cloud compute / GPU infrastructure spend.
Regulatory & Governance Compliance	High risk of silent algorithmic bias accumulation violating audit standards.	Comprehensive, version-controlled audit trails documenting all drift triggers.	Full alignment with modern AI governance frameworks.

9. Open Challenges and Future Research Directions

1. **The Verification Dilemma in Extreme Label Latency:** Detecting virtual covariate shift in inputs is statistically straightforward. However, determining whether an input shift $P(X)$ indicates a corresponding change in the conditional mapping $P(Y | X)$ remains challenging without ground truth labels. Excessive sensitivity to input drift risks triggering costly retraining cycles for benign shifts.
2. **Catastrophic Forgetting in Continual Learning:** Fine-tuning deep neural networks on recent streaming windows often degrades weights that represent older, recurrent data distributions. Techniques such as experience replay buffers and parameter-isolation architectures remain active areas of research.
3. **Multi-Agent and Strategic Drift:** In adversarial domains (such as cybersecurity, fraud prevention, and competitive auctions), external agents adapt their behavior to bypass known decision boundaries. Modeling these dynamics requires combining statistical drift tracking with game-theoretic models.
4. **Constrained Edge Telemetry:** Computing high-dimensional optimal transport distances (such as Wasserstein metrics) on edge microcontrollers and industrial gateways is constrained by limited compute. Efficient approximations, like 1D sliced projections and randomized matrix sketches, are critical for embedded deployments.

10. Conclusion

Machine learning systems deployed into dynamic environments inevitably degrade as statistical distributions diverge from initial baseline assumptions. Treating machine learning models as static software artifacts introduces significant operational and financial risks.

By applying a formal mathematical framework across covariate shift, prior shift, and concept drift, engineering teams can implement targeted statistical metrics—such as the Population Stability Index, Kolmogorov-Smirnov tests, and Wasserstein distances—to detect performance decay in real time, even when ground-truth labels are delayed. Integrating continuous streaming diagnostics with automated, closed-loop retraining pipelines provides self-healing capabilities that maintain model accuracy, safeguard system reliability, and preserve enterprise value across production deployments.

Declarations

Funding. No funding was received for this study.

Conflict of interest. The authors declare that they have no conflicts of interest.

Data availability. The public benchmark datasets and open-source models analyzed in this study are accessible via their respective official repositories. Derived evaluation logs, trained weights, and proprietary test scripts generated during the current study are not publicly available due to institutional research constraints, but are available from the corresponding author upon reasonable request.

Author contributions.

- **Nguyen Van Minh:** Conceptualization, methodology, software, project administration, supervision, and writing—original draft.
- **Tran Thi Mai Hoa:** Formal analysis, methodology, validation, and writing—review and editing.
- **Le Hoang Nam:** Software, data curation, investigation, and visualization.
- **Nguyễn Dũng:** Methodology, formal analysis, investigation, and writing—original draft preparation.
- **Pham Duc Anh:** Validation, resources, data curation, and writing—review and editing.

All authors read and approved the final manuscript.

Acknowledgments. The authors express their gratitude to Vietnam National University, Hanoi University of Science and Technology, and the participating institutions for providing computational facilities and institutional support.

Ethical approval. Not applicable. This study utilized synthetic and publicly available benchmarks and did not involve human participants or animals.

Declaration of Generative AI Use. Generative AI tools were utilized solely for language editing, stylistic refinement, and structural clarity of the manuscript. The authors critically reviewed, verified, and edited all outputs, and take full academic responsibility for the integrity and content of the published work.

References

1. Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)* (pp. 443–448). SIAM. <https://doi.org/10.1137/1.9781611972771.42>
2. Ditzler, G., Roveri, M., Alippi, C., & Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4), 12–25. <https://doi.org/10.1109/MCI.2015.2471238>
3. Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence – SBIA 2004* (pp. 286–295). Springer. https://doi.org/10.1007/978-3-540-28645-5_29
4. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), 1–37. <https://doi.org/10.1145/2523813>
5. Hovakimyan, G., & Bravo, J. M. (2024). Evolving strategies in machine learning: A systematic review of concept drift detection. *Information*, 15(12), 786. <https://doi.org/10.3390/info15120786>
6. Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, 104, 799–834. <https://doi.org/10.1016/j.ymssp.2017.11.016>
7. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12), 2346–2363. <https://doi.org/10.1109/TKDE.2018.2876857>
8. Nishida, K., & Yamauchi, K. (2007). Detecting concept drift using statistical testing. In *Discovery Science* (pp. 264–269). Springer. https://doi.org/10.1007/978-3-540-75488-6_27

9. Rabanser, S., Günnemann, S., & Lipton, Z. (2019). Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 1396–1408.
10. Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4), 964–994. <https://doi.org/10.1007/s10618-015-0448-4>
11. Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1), 69–101. <https://doi.org/10.1007/BF00116900>
12. Žliobaitė, I., Pechenizkiy, M., & Gama, J. (2016). An overview of concept drift applications. In *Big Data Analysis: New Trends and Applications* (pp. 91–114). Springer. https://doi.org/10.1007/978-3-319-41573-4_4