International Journal of Advanced Engineering, Management and Science (IJAEMS)
Peer-Reviewed Journal
ISSN: 2454-1311 | Vol-12, Issue-3; May-Jun, 2026
Journal Home Page: https://ijaems.com/
DOI: https://dx.doi.org/10.22161/ijaems.123.37

# Continuous Integration and Delivery of Machine Learning Models in Real-Time Financial Decision Systems

Kaleshwar Aryasomayajula

Sr Specialist Software Developer, Charles Schwab, San Tan Valley, Arizona, USA

Received: 20 May 2026; Received in revised form: 15 Jun 2026; Accepted: 19 Jun 2026; Available online: 23 Jun 2026

***Abstract*** *— Real-time financial decision systems depend on machine learning models that change after release because input data, customer behavior, and portfolio risk move faster than traditional model governance cycles. This article examines continuous integration and delivery for models used in credit, fraud, eligibility, and limit decisions. The aim is to build an analytical framework for CI/CD in financial machine learning without presenting experimental claims. The study draws on ten recent sources on MLOps, data quality, concept drift, non-traditional credit data, and AI risk governance. Comparative source analysis, conceptual synthesis, classification, and analytical generalization guide the work. The results define three requirements: controlled pipeline promotion for code, data, features, and model artifacts; monitoring that separates data failures from model deterioration; and release governance that connects automation with audit evidence. Practitioners can use the framework to design model promotion, rollback, threshold review, and real-time trace logging for financial decision services under supervisory oversight.*

***Keywords*** *— **MLOps, CI/CD, machine learning models, financial decision systems, model risk management, credit scoring, data drift, real-time analytics, explainable AI, deployment governance.***

## I. INTRODUCTION

Financial institutions now place credit scoring, fraud screening, transaction authorization, customer eligibility checks, and limit assignment inside digital decision flows. A model release in this environment affects customer access, credit exposure, compliance records, and response time in the same operational cycle. The release process governs the movement of decision logic through development, validation, deployment, monitoring, rollback, and later threshold review.

The problem sharpens when different data sources enter the decision path at different speeds. Credit bureau files, account records, wallet activity, merchant transactions, device signals, and application events age at unequal rates. A model that depends on recent behavioral traces and a model that relies on older bureau observations carry different operational risks. CI/CD for financial machine learning has to control code, data, features, model binaries, thresholds, explanations, and approval rules as linked release artifacts.

This article aims to develop an analytical framework for continuous integration and delivery of machine learning models in real-time financial decision systems.

The first objective is to define how CI/CD changes when machine learning artifacts become part of the financial decision infrastructure. The second objective is to identify monitoring requirements that connect model drift, data quality, latency, and business risk. The third objective is to formulate a governance sequence for model deployment and updating that

does not weaken auditability, explainability, or model risk control.

The novelty of the article lies in treating financial ML CI/CD as an engineering and governance mechanism. The proposed view connects automated pipeline gates with model risk evidence, drift monitoring, decision traceability, and threshold control. The hypothesis states that CI/CD strengthens real-time financial decision systems only when release automation remains tied to model risk governance, observable data quality, and auditable deployment evidence.

## II. MATERIALS AND METHODS

The author selected sources published between 2022 and 2026. The selection covered peer-reviewed articles, conference papers, and regulatory materials directly related to MLOps, CI/CD automation, data quality monitoring, drift detection, credit scoring with non-traditional data, and governance of machine learning in financial systems. The corpus covers data quality and MLOps in production environments [1], regulatory treatment of machine learning in IRB credit models [2], non-traditional data in credit scoring [3], concept drift in evolving environments [4], MLOps adoption and maturity assessment [5], MLOps architecture [6], systematic review evidence on MLOps practices and tools [7], cross-sector AI risk management [8], MLOps tools and operational challenges [9], and MLOps taxonomy and methodology [10].

The author used comparative source analysis to separate ordinary software CI/CD from ML-specific CI/CD, source analysis to extract design requirements from the selected works, conceptual synthesis to connect MLOps pipelines with financial decision governance, classification to group deployment risks, and analytical generalization to formulate a model promotion sequence. The methods correspond to three research tasks: pipeline transformation, runtime monitoring, and release governance.

## III. RESULTS

Recent MLOps research describes production machine learning as a coordinated system of software engineering, data engineering, model training, serving, monitoring, and feedback. Kreuzberger et al. define MLOps through principles, components, roles, architecture, and workflows that help teams move models from development into operation [6]. Testi et al. organize the MLOps literature into a taxonomy and propose methodology elements for ML pipeline releases [10]. Symeonidis et al. focus on definitions, tools, and operational challenges, including deployment, monitoring, retraining, and experiment tracking [9]. For real-time financial decisions, these studies lead to a concrete requirement: the pipeline must validate the scoring service and data transformations before a credit or fraud endpoint receives live traffic.

Systematic review evidence narrows the same requirement. Lima et al. identify practices, maturity models, roles, tools, and challenges related to the automation of ML operationalization, based on a broad review of the literature [7]. The review does not provide financial institutions with a single, universal lifecycle. It does, however, justify a controlled internal sequence where each model candidate passes the same evidence gates: data lineage, feature validation, training reproducibility, metric comparison, latency testing, explanation checks, approval logging, and rollback planning. Financial teams need this sequence because model promotion affects decisions that customers may later dispute.

MLOps adoption research adds the organizational layer. John et al. provide an empirically validated framework and maturity model for adopting, assessing, and advancing MLOps practices across companies [5]. Their findings point to the influence of maturity, use-case classification, and existing engineering routines. A fraud detection model, a credit eligibility model, and a collections prioritization model do not require identical release gates. A personalization model may enter production through a lighter path. A model that declines credit applications needs tighter validation, adverse action traceability, and approval evidence before release.

Financial credit-scoring research explains why fast-changing data sources complicate the pipeline. Gambacorta et al. compare machine learning credit-scoring models using non-traditional transaction-level data with traditional loss and default models [3]. Their work provides the present article with a focused operational implication: financial institutions that use behavioral and transaction signals need release checks

to ensure data freshness, feature stability, and population movement. A technically correct model can yield poor decisions if the input environment shifts before deployment.

Regulatory materials give a separate constraint. The European Banking Authority reports that institutions use, or plan to use, machine learning in selected parts of internal ratings-based credit modeling, particularly in probability-of-default estimation and risk differentiation. The same report discusses concerns regarding model complexity, explainability, overfitting, skilled labor, traceability, and interpretation of material model changes [2]. These concerns change the meaning of CI/CD. Automated release cannot end with a passing build, a valid container image, or a healthy endpoint. The pipeline has to generate evidence for model change classification, independent validation, approval responsibility, and later audit review.

A comparison of MLOps studies and financial regulation reveals the first analytical result. MLOps researchers describe automation as a way to reduce manual friction in model operationalization [6; 7; 9; 10]. Financial supervisors and risk frameworks place pressure on traceability, model change control, and accountable use [2; 8]. Financial ML CI/CD needs two operating layers. Engineers automate repeatable tests, artifact movement, staging promotion, monitoring hooks, and rollback triggers. Risk owners retain control over thresholds, protected-group impact assessments, adverse-action logic, and material model-change decisions.

Runtime monitoring forms the second result. Hinder et al. review concept drift in evolving environments and connect drift detection with monitoring, anomaly detection, and reliable use of machine learning in changing settings [4]. In real-time finance, drift has several forms. Applicant income patterns, merchant composition, repayment timing, device usage, and macroeconomic pressure can change the meaning of model features. A scoring endpoint may still meet latency targets while score interpretation deteriorates. CI/CD needs post-release monitors for training and live feature distributions, score distributions, rejection rates, override rates, delinquency signals, and delayed outcome labels.

Production data quality research sharpens that point. Bayram et al. propose an end-to-end framework that combines data quality assessment, drift detection, adaptive data quality metrics, and MLOps in production environments [1]. The financial implication concerns failure diagnosis. A decline in decision quality may come from model deterioration, missing transaction fields, delayed account updates, changed vendor encoding, duplicate events, or a broken feature job. A real-time CI/CD system must monitor data quality before model performance, because retraining cannot fix an upstream data defect.

Figure 1 presents the control loop derived from the MLOps architecture literature and adapted to financial decision services. The figure uses the architecture logic of pipeline automation, artifact management, serving, monitoring, and feedback from MLOps research [6].

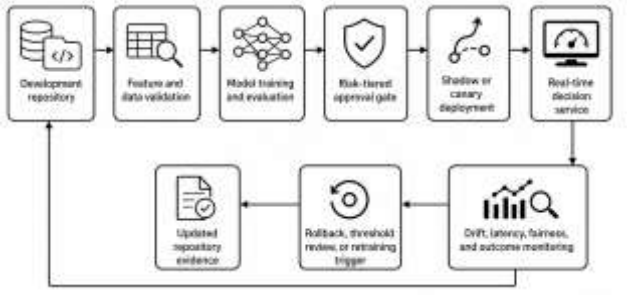


*Fig. 1: CI/CD control loop for machine learning models in real-time financial decision systems (adapted from MLOps architecture components in [6])*

Developers move code through software checks. Data teams move features through quality, lineage, and distribution checks. Model owners move model artifacts through predictive, operational, and governance checks. The three movements meet at the serving layer. A service release can pass technical tests and still fail if live feature distributions differ from those in the training data. A model can perform well in validation and still fail if its explanation schema cannot support adverse action review. A governance-approved model can fail in production if feature retrieval exceeds the service's target latency.

A second comparison across sources clarifies the monitoring problem. Hinder et al. focus on detecting and explaining change in evolving data streams [4]. Bayram et al. connect drift detection with production data quality [1]. John et al. frame MLOps adoption in

terms of maturity and organizational practices [5]. The EBA report frames machine learning in credit risk in terms of explainability, validation, and model change concerns [2]. The combined conclusion is narrow: financial monitoring must not reduce model health to a single generic indicator. Operators need separate signals for data validity, population movement, latency, calibration, threshold stability, explanation completeness, and downstream credit performance.

Governance gives the third result. The NIST AI Risk Management Framework organizes AI risk work around the functions of govern, map, measure, and manage [8]. Financial CI/CD can translate this structure into release controls. Govern covers ownership, approval authority, inventory, and accountability. Map covers use-case classification, affected customers, model boundaries, and decision impact. Measure covers validation metrics, drift tests, fairness checks, latency profiles, and explanation quality. Manage cover rollbacks, escalations, threshold revisions, retraining, and retirements.

A model candidate needs a governed state transition. The candidate moves from development to staging after reproducibility, data lineage, and baseline metrics are entered into the record. It moves from staging to shadow mode after integration tests, latency tests, and trace tests pass. It moves from shadow mode to limited production after challenger comparison and live monitoring confirms acceptable behavior. It moves from limited production to broader use after governance review confirms rollback readiness and decision evidence. This sequence follows from the convergence of MLOps architecture, adoption maturity research, drift detection, data quality studies, credit-risk regulation, and AI risk management.

## IV. DISCUSSION

Financial ML CI/CD needs a release design that reflects the decision's impact on customers and risk exposure. Ordinary software CI/CD focuses on build reliability, automated tests, and delivery speed. Financial ML CI/CD governs the movement of decision logic that changes with data, model assumptions, policy thresholds, and supervisory expectations. The pipeline has to serve as both a release and a risk-control process.

The implementation model starts with a model inventory and use-case classification. Each candidate receives a risk tier before engineering teams prepare for deployment. The tier depends on customer impact, legal sensitivity, decision reversibility, exposure size, dependence on alternative data, algorithmic opacity, and the level of automation in the final decision. This classification decides which gates enter the pipeline. Low-impact analytical models pass through standard tests and monitoring. Credit decline, fraud blocking, dynamic limit assignment, and adverse pricing models require stricter gates.

Table 1 compares the CI/CD gates needed in real-time financial ML systems. The table shows how one pipeline can preserve release speed while applying different controls to code, data, model behavior, and governance evidence.

The table separates failure surfaces that teams often merge during rushed deployment. A model can fail the data gate despite strong validation metrics. It can pass the explanation gate and then fail latency testing. It can pass technical gates and still require governance review because a threshold change affects approval volume or protected customer groups. Separate gates reduce incident confusion because engineers, validators, and business owners can locate the failure source without treating every alert as model decay.

Staged exposure gives the pipeline a safer release route. Shadow deployment lets the candidate score live applications without influencing the final decision. Canary deployment sends a limited share of low-risk traffic through the new model under strict rollback rules. Full production follows only after the monitoring window confirms stable score distributions, acceptable latency, completed explanations, and controlled decision mix. Credit, fraud, and eligibility models need this staged route because repayment outcomes, fraud confirmations, and complaints arrive after the original decision.

*Table 1. CI/CD gates for real-time financial machine learning deployment*

| Gate | Main object checked | Evidence produced | Failure response |
|---|---|---|---|
| Code integration gate | Application code, scoring service, API contract | Build logs, unit tests, and integration test reports | Block merge or return to development |
| Data and feature gate | Source freshness, missingness, schema, feature distribution | Data lineage record, quality profile, drift report | Stop deployment or isolate upstream data failure |
| Model validation gate | Candidate model, challenger baseline, calibration, stability | Validation report, metric comparison, threshold sensitivity note | Reject the candidate or require recalibration |
| Latency and resilience gate | Serving endpoint, feature retrieval, fallback logic | Load test result, timeout profile, failover record | Keep the model in staging or reduce the deployment scope |
| Explainability gate | Reason codes, trace schema, customer-facing rationale | Explanation sample set, audit trace, policy mapping | Prevent production use for adverse decisions |
| Governance approval gate | Model change type, affected portfolio, and responsible owners | Approval record, release note, rollback plan | Hold release for review |
| Post-deployment gate | Live drift, score movement, decision mix, outcome signals | Monitoring dashboard, alert log, and incident register | Rollback, canary freeze, threshold review, retraining trigger |

Monitoring metrics require separation into technical, statistical, decision, and governance groups. Technical metrics cover latency, timeout rate, service availability, feature store response time, and queue delays. Statistical metrics cover population stability, feature drift, score drift, calibration drift, missing-value changes, and label delay. Decision metrics cover approval rate, decline rate, manual review rate, override rate, average limit, rejection reason concentration, and segment-level threshold movement. Governance metrics cover model inventory status, unresolved validation findings, explanation trace completeness, expired approvals, and open incidents.

Table 2 presents a monitoring matrix for deployment operation. The matrix compares what each metric group detects and what action follows from the signal.

*Table 2. Monitoring matrix for CI/CD operation in financial ML systems*

| Metric group | What it detects | Typical trigger | Operational response |
|---|---|---|---|
| Technical performance | Service degradation and infrastructure failure | P95 latency above service target, timeout spike, failed feature lookup | Activate fallback, freeze rollout, inspect serving path |
| Data quality | Upstream data breakage or unstable feature supply | Schema mismatch, missingness increase, stale source timestamp | Stop model promotion, repair data pipeline, and rerun validation |
| Distribution stability | Population shift and changing applicant mix | Feature drift, score distribution movement, and segment imbalance | Move candidate to observation mode, review threshold sensitivity |

| | | | |
|---|---|---|---|
| Predictive performance | Model deterioration after outcomes mature | Calibration error, AUC decline, and higher loss on recent cohorts | Trigger challenger test, recalibrate, retrain, or rollback |
| Decision behavior | Unintended business effect of model release | Approval rate jump, decline reason concentration, review queue overload | Adjust rollout share, inspect policy rules, and review segment thresholds |
| Explainability and audit | Incomplete evidence for adverse decisions | Missing reason codes, invalid trace schema, weak policy mapping | Block adverse-action use, repair trace logic, record exception |
| Governance control | Weak ownership and unresolved release risk | Expired approval, open validation issue, missing rollback owner | Escalate to model governance, pause deployment state transition |

The matrix turns monitoring into operational routing. A drift alert does not automatically justify retraining. A missingness alert first directs engineers to the data pipeline, while a latency alert points to feature retrieval and serving resources. An explanation alert blocks adverse-decision use even when predictive metrics remain stable. This separation prevents a common deployment error: treating retraining as the default cure for every production anomaly.

A workable implementation sequence contains six steps. First, the institution builds a model inventory that identifies the decision purpose, owner, customer effect, input sources, explanation requirement, and rollback mechanism. Second, the engineering team creates an artifact registry for code, data snapshots, feature definitions, model binaries, validation reports, and approval records. Third, CI checks verify reproducibility, data quality, schema stability, and security controls before staging. Fourth, CD promotes the candidate into shadow mode and then canary mode under monitoring rules. Fifth, governance reviewers approve or reject broader production use through pipeline evidence. Sixth, post-deployment monitoring routes alerts to rollback, recalibration, retraining, or threshold review.

Threshold governance needs its own control line. Real-time credit and fraud systems often combine model scores with policy thresholds. A model release may leave the model unchanged while changing the threshold. That change can alter approval volume, expected losses, false positives, and customer access. The CI/CD pipeline needs to have version thresholds, approval rules, and reason-code mappings with the same discipline as for model artifacts. A threshold change without a model change still alters customer-facing decision logic.

Decision-level explainability has higher practical value than model-level explanation alone. Model-level explanations describe broad behavior. Decision traces show what happened in a specific application. A real-time financial system needs to record the model version, the feature values used for scoring, the score output, the threshold applied, policy overrides, reason codes, and a timestamped decision path. Compliance teams need that record for disputes. Engineers need the same record to reproduce incidents after customer complaints.

The framework has limits. Independent validation remains necessary. Fairness testing still needs a separate design. Delayed credit outcomes still complicate performance assessment after deployment. The framework reduces operational ambiguity by consolidating release evidence, monitoring signals, and governance decisions into a single lifecycle. Its value depends on disciplined ownership. Business teams that adjust thresholds outside the artifact registry, vendors that change schemas without notice, and alerts without escalation owners can weaken any CI/CD design.

## V. CONCLUSION

CI/CD changes when machine learning models enter real-time financial decision infrastructure. The release object spans software code, data, features, model artifacts, thresholds, reason codes, and audit records.

A financial ML pipeline needs gates for reproducibility, data quality, validation, latency, explainability, governance approval, and post-deployment monitoring.

Monitoring extends beyond technical availability. Real-time decision systems require separate monitoring of data quality, feature drift, score changes, latency, calibration, decision mix, explanation completeness, and governance status. This separation helps institutions distinguish between model deterioration, data failure, and infrastructure degradation, as well as policy threshold effects.

Deployment governance works best as a staged state transition: development, staging, shadow mode, canary release, limited production, full production, and monitored operation. Each transition needs evidence proportional to the model risk tier and decision impact. The hypothesis is confirmed: CI/CD strengthens real-time financial decision systems when release automation remains bounded by model risk governance, observable data quality, and auditable deployment evidence.

## REFERENCES


[1] Bayram, F., Ahmed, B. S., & Hallin, E. (2026). End-to-end data quality-driven framework for machine learning in a production environment. Heliyon, 12(1), e44416. https://doi.org/10.1016/j.heliyon.2025.e44416

[2] European Banking Authority. (2023). Follow-up report on machine learning for IRB models. European Banking Authority. https://www.eba.europa.eu/sites/default/files/document_library/Publications/Reports/2023/1061483/Follow-up%20report%20on%20machine%20learning%20for%20IRB%20models.pdf

[3] Gambacorta, L., Huang, Y., Qiu, H., & Wang, J. (2024). How do machine learning and non-traditional data affect credit scoring? New evidence from a Chinese fintech firm. Journal of Financial Stability, 73, 101284. https://doi.org/10.1016/j.jfs.2024.101284

[4] Hinder, F., Vaquet, V., & Hammer, B. (2024). One or two things we know about concept drift: A survey on monitoring in evolving environments. Part A: Detecting concept drift. Frontiers in Artificial Intelligence, 7, 1330257. https://doi.org/10.3389/frai.2024.1330257

[5] John, M. M., Holmström Olsson, H., & Bosch, J. (2025). An empirical guide to MLOps adoption: Framework, maturity model and taxonomy. Information and Software Technology, 183, 107725. https://doi.org/10.1016/j.infsof.2025.107725

[6] Kreuzberger, D., Kühl, N., & Hirschl, S. (2023). Machine learning operations (MLOps): Overview, definition, and architecture. IEEE Access, 11, 31866–31879. https://doi.org/10.1109/ACCESS.2023.3262138

[7] Lima, A., Monteiro, L., & Furtado, A. (2022). MLOps: Practices, maturity models, roles, tools, and challenges: A systematic literature review. In Proceedings of the 24th International Conference on Enterprise Information Systems, Volume 1 (pp. 308–320). SciTePress. https://doi.org/10.5220/0010997300003179

[8] National Institute of Standards and Technology. (2023). Artificial intelligence risk management framework (AI RMF 1.0). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.100-1

[9] Symeonidis, G., Nerantzis, E., Kazakis, A., & Papakostas, G. A. (2022). MLOps: Definitions, tools and challenges. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (pp. 453–460). IEEE. https://doi.org/10.1109/CCWC54503.2022.9720902

[10] Testi, M., Ballabio, M., Frontoni, E., Iannello, G., Moccia, S., Soda, P., & Vessio, G. (2022). MLOps: A taxonomy and a methodology. IEEE Access, 10, 63606–63618. https://doi.org/10.1109/ACCESS.2022.3181730