Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Drift-Triggered Model Retraining in Apache Spark Structured Streaming for Fraud Detection

Naresh Miriyalu

BITS Pilani, India

ARTICLE INFO

Received: 08 Nov 2024

Revised: 28 Dec 2024

Accepted: 15 Feb 2025

ABSTRACT

Fraud models faced on transaction streams are subject to concept drift and delayed labels accompanied by simultaneous class imbalance. In this study, we propose and test a drift-triggered retraining design for Apache Spark Structured Streaming for a strictly bounded collection of 28 peer-reviewed studies published between 2015 and 2024, and a repeatable controlled micro-batch experiment. The experiment produced 120,000 transaction-like observations, where one large drift occurred at batch 45 and a slow drift occurred between batch 80 and 95. Three different maintenance strategies were considered – static model, periodic retraining every 15 batches, and a Population Stability Index (PSI) – triggered champion–challenger strategy with a 3-batch persistence rule, a 0.24 composite threshold, 20-batch cooldown, and 2-batch label delay, using different time-ordered training and validation windows. Retraining after drift resulted in a mean AUPRC of 0.281, mean recall of 0.183 at a fixed 2% review budget and mean log loss of 0.715 across 105 batches evaluated for post initialisation. Compared to the static model, it boosted mean AUPRC by 7.6%, boosted review-budget recall by 6.9%, and reduced log loss by 39.7%. Its AUPRC and recall differences were small and the moving-block bootstrap intervals crossed zero as compared to frequent periodic retraining, and it processed 111,000 rows of training data as opposed to 183,000, a reduction of 39.3%. Abrupt drift was observed after two batches (2,000 transactions), while gradual drift was observed after 11 batches (11,000 transactions). Results validate drift-triggered retraining as an efficient maintenance policy for compute, but also demonstrate that distributional triggers alone may not be the most effective maintenance policy to produce the best calibration when drift is gradual. The paper thus proposes a Spark control plane that incorporates feature drift, delayed performance signals, cooldown logic, champion-challenger validation, versioned model registration and complete audit metadata.

Keywords: Apache Spark; Structured Streaming; concept drift; credit-card fraud detection; Population Stability Index; champion–challenger; continual learning; model retraining; AUPRC; delayed labels

1. INTRODUCTION

Real-time fraud detection is a traditional classification problem with a twist. As customers use new channels, merchants modify their business models and fraudsters adapt to controls, transaction behaviour evolves. This statistical correlation of observed transaction characteristics with the fraud flag is therefore changing. However, Webb et al. (2016) differentiate the form, rate and frequency of such changes, Gomes et al. (2017), Sethi and Kantardzic (2017) and Barros and Santos (2018) demonstrate the need for evaluating stream learners and drift detectors against multiple types of drift test sets.

The fraud detection literature introduces three operational restrictions. Firstly, fraudulent transactions on the network are relatively uncommon and errors have unequal costs. The significance of feature engineering, cost sensitivity and ensemble decisions are illustrated by Bahnsen et al. (2016), Nami and Shajari (2018), and Randhawa et al. (2018). Second, payment behaviour is sequential and relational. Jurgovsky et al. (2018), Lucas et

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

al. (2020) and Van Vlasselaer et al. (2015) demonstrate that information that is not in isolated rows can be understood from the history, temporal patterns and extensions of networks. Thirdly, labels are late. Carcillo et al. (2018) focus on issues of imbalance, non-stationarity and feedback latency in a Kafka–Spark–Cassandra architecture, whereas Dal Pozzolo et al. (2018) call for more realistic time-aware evaluation and learning approaches.

This paper is a design and simulation study and is not intended to provide access to any institutional transaction records or measurements of the Spark cluster.

1.1 Research Problem

In a fixed model, drift can cause it to become less discriminative or less calibrated without it being detected, and retraining of all batches is a waste of computational resources and can magnify noise. Calendar-based retraining is predictable but might be retrained during calm times and respond slowly to sudden shifts. Drift-triggered retraining is a conditional alternative, which value depends on the detection delay, on the control of false triggered, on the validation policy, on the selection of data-windows, and on the cost of model updates. These quantities should be evaluated together, and not as a separate statistical test for drift.

1.2 Research Questions

1. RQ1: Does drift-triggered retraining improve fraud-ranking quality and review-budget recall relative to a static model under abrupt and gradual drift?

2. RQ2: Can drift-triggered retraining retain performance comparable to frequent periodic retraining while processing fewer training observations?

3. RQ3: How does detection delay differ between abrupt and gradual drift under a persistent composite-PSI rule?

4. RQ4: Which Spark Structured Streaming components are required to implement the detection, retraining, validation, deployment and audit loop without blocking the scoring path?

2. EVIDENCE BASE AND ANALYTICAL LITERATURE REVIEW

The author provides 28 journal articles that form the evidence corpus. Sixteen concern Spark, streaming fraud, feature engineering, ensemble/deep learning, online fraud or continual fraud detection; twelve concern concept-drift characterisation, detection, evaluation or adaptation. The corpus is limited in time to 2024. There were 20 published studies between 2018 and 2024 (71.4%) including 10 articles published in Expert Systems with Application (35.7%). The focus of that journal suggests a very strong focus on applied intelligent systems, though also notes that the corpus should not be interpreted as a comprehensive systematic literature review of all streaming and MLOps research.

Table 1. Composition and temporal scope of the evidence corpus

Indicator	Value	Interpretation
Total sources	28	All peer-reviewed journal articles supplied by the author
Fraud/Spark sources	16	Platform, streaming architectures, fraud modelling, feature engineering and continual fraud
Concept-drift sources	12	Drift characterisation, detection, evaluation, adaptation and robustness
Published 2018–2024	20 (71.4%)	Majority of evidence lies in the later seven years of the review window
Published in 2024	3 (10.7%)	Online fuzzy fraud detection, catastrophic forgetting and QuadCDD
Expert Systems with	10 (35.7%)	Most represented outlet in the supplied corpus

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Applications		

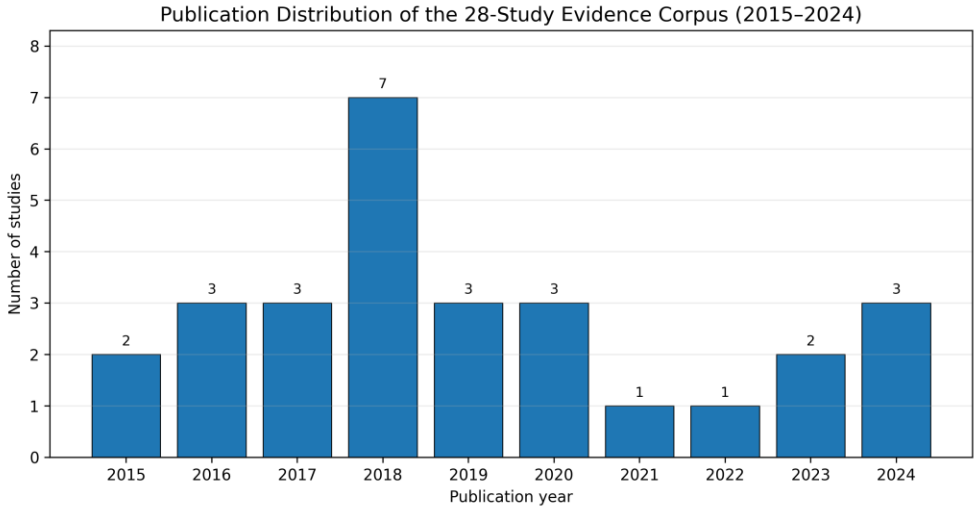

Figure 1. Publication distribution of the 28-study evidence corpus, 2015–2024.

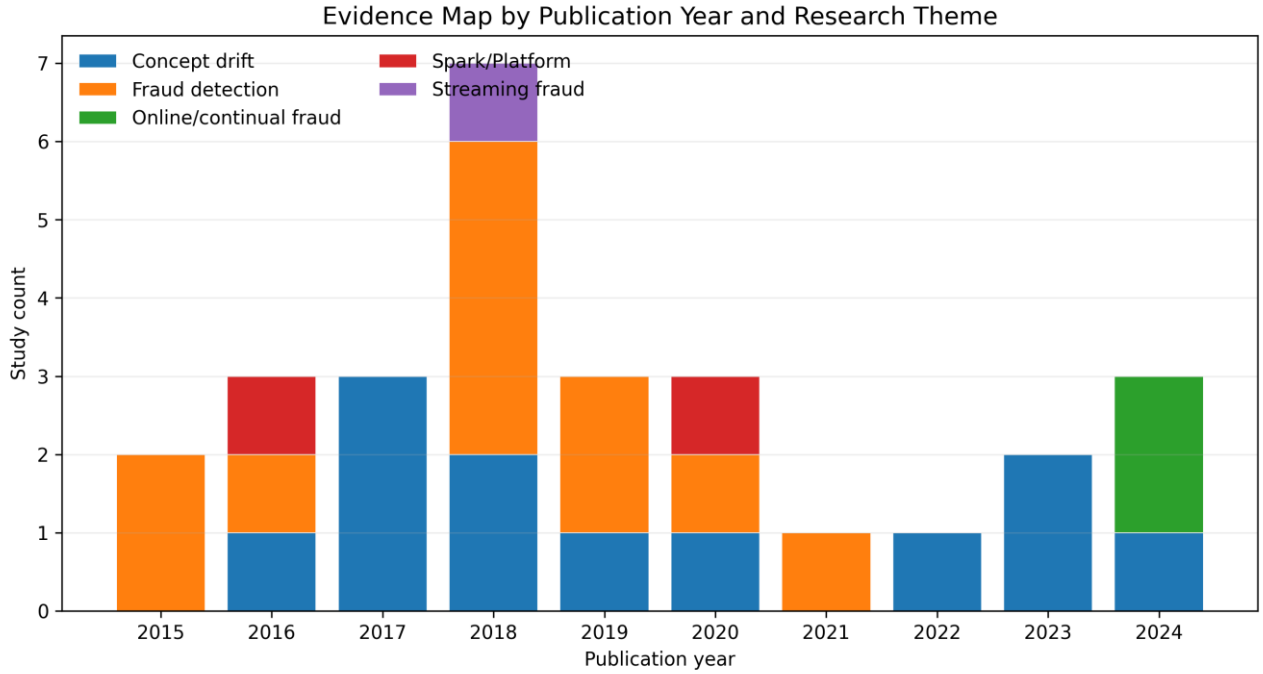

Figure 2. Evidence map by publication year and research theme.

2.1 Spark and Streaming Foundations

According to Meng et al. (2016), MLlib is Spark's distributed machine learning library and they highlight reusability, pipelines, statistical primitives and integration with the ecosystem. Mahapatra and Prehofer (2020) study Spark abstractions of data and demonstrate how to compose and verify complex Spark programs as modular flows. In this paper, these contributions are divided into two parts: The streaming query executes the deterministic feature and scoring tasks, while the more expensive model fitting is done as a separate batch task.

The closest system precedent is SCARFF. Carcillo et al. (2018) propose the combination of Kafka, Spark and Cassandra to solve the problems of scalability, imbalance, non-stationarity and feedback latency. In the present study, that architectural line is extended by making model maintenance a part of the control plane: detector state is

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

updated on a per-micro-batch basis, retraining is triggered on a conditional basis, and promotion necessitates a time-ordered champion–challenger gate.

2.2 Fraud-Modelling Evidence

Anomaly ranking is implemented in BankSealer to aid in investigation, not just based on a binary decision (Carminati et al., 2015). APATE extends transaction capabilities with network based extensions (Van Vlasselaer et al., 2015). Bahnsen et al. 2016, show the usefulness of global behavioral characteristics. The realistic modelling and learning under temporal change is proposed by Dal Pozzolo et al. (2018) and the formulation of card fraud as a sequence-classification problem is proposed by Jurgovsky et al. (2018). All of these studies suggest that a retraining system should include not only a model of parameters, but also definitions, time-window logic, relational or sequential aggregation state.

Nami, Shajari (2018) put emphasis on cost sensitivity, while Randhawa et al. (2018) employed the methods of AdaBoost and majority voting for improving classification when dealing with imbalance. Fiore et al. (2019) apply generative adversarial learning to enhance classification under imbalance, and Kim et al. (2019) formalise champion–challenger analysis using hybrid ensemble and deep-learning models. Multi-perspective hidden Markov models are used to automate feature engineering by Lucas et al. (2020), and unsupervised and supervised learning is used by Carcillo et al. (2021) to combine the two. These contributions inspire a promotion gate based on the business constrained recall and AUPRC, which considers other metrics apart from accuracy.

The two studies of fraud in 2024 make the problem of adaptation even harder. In this work, Charizanos et al. (2024) present an online fuzzy framework for non-stationary, imbalanced transaction data. In the context of continual credit-card fraud detection, Lebichot et al. (2024) evaluates catastrophic forgetting and discusses the stability–plasticity dilemma. The proposed retraining window therefore combines recent observations and a limited amount of historical data instead of discarding all previous data and only using a small set of post-drift observations.

2.3 Concept Drift Detection and Adaptation

A vocabulary for drift scope and path is provided by Webb et al. (2016). Gomes et al. (2017) present adaptive random forest for evolving streams, Sethi and Kantardzic (2017) discuss reliable drift detection in unlabeled streams, and Brzezinski and Stefanowski (2017) present prequential AUC for drifting streams. Demšar and Bosnić (2018) identify change using model explanations while Barros and Santos (2018) compare detectors at scale. The studies justify to monitor both feature distributions and model behaviour as this is not enough when supervision is delayed.

In this work, Yu et al. (2019) couple detection and adaptation by a hierarchical hypothesis testing framework. In the case of canoes and Krawczks (2020), ensembles are updated based on the Kappa measure of evidence. In this paper, Guo et al. (2022) classify the types of drifting using multiple sliding windows. Korycki and Krawczyk (2023) demonstrate the attack of the detector itself with poisoning, Yu et al. (2023) speak of group drift in different streams, and Wang et al. (2024) talk about drift by starting, end, severity and type. Operational implication is that during a production trigger event, the identity of the detector, the features affected, estimates of onset, persistence, severity and evidence used for model promotion should be stored.

Table 2. Detailed evidence matrix for the 28 supplied studies

Study	Primary focus	Method/approach	Design implication for this paper	Boundary/limitation
Meng et al. (2016)	Spark/MLlib	Distributed ML library and pipelines	Supports offline training and versioned model pipelines	Does not study fraud drift or Structured Streaming retraining

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Mahapatra & Prehofer (2020)	Spark programming	Flow-based abstraction and code generation	Supports modular validation of Spark dataflow	Not a fraud or drift benchmark
Carminati et al. (2015)	Online banking fraud	Decision-support anomaly ranking	Shows analyst-oriented ranking requirement	Different domain and no automatic retraining
Van Vlasselaer et al. (2015)	Card fraud	Network-based feature extensions	Motivates relational features and state	No Spark retraining control plane
Bahnsen et al. (2016)	Card fraud	Behavioural feature engineering	Motivates rolling velocity and aggregation features	Batch-oriented evaluation
Carcillo et al. (2018)	Streaming card fraud	Kafka–Spark–Cassandra SCARFF	Direct architectural precedent; handles latency and non-stationarity	Retraining trigger logic not fully isolated as a control plane
Dal Pozzolo et al. (2018)	Card fraud	Realistic temporal modelling and learning	Supports time-ordered validation and delayed labels	Institution-specific data limit direct reproducibility
Jurgovsky et al. (2018)	Card fraud	Sequence classification	Motivates temporal history features	Sequence model is heavier than the logistic benchmark used here
Nami & Shajari (2018)	Payment-card fraud	Cost-sensitive dynamic RF and k-NN	Supports business-cost-aware metrics	No streaming orchestration analysis
Randhawa et al. (2018)	Card fraud	AdaBoost and majority voting	Motivates ensemble/challenger comparisons	No explicit concept-drift experiment
Fiore et al. (2019)	Card fraud	GAN-assisted classification	Addresses minority-class representation	Synthetic augmentation can introduce additional governance risk
Kim et al. (2019)	Card fraud	Champion–challenger hybrid ensemble	Directly supports controlled model promotion	No Spark Structured Streaming implementation
Lucas et al. (2020)	Card fraud	Automated multi-perspective HMM features	Supports automated temporal feature generation	Complexity may increase scoring latency
Carcillo et al.	Card fraud	Combined unsupervised and	Supports dual monitoring and	Not focused on retraining

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

(2021)		supervised learning	prediction signals	orchestration
Charizanos et al. (2024)	Online card fraud	Fuzzy online fraud framework	Recent evidence for online non-stationary modelling	Different learner and operational assumptions
Lebichot et al. (2024)	Continual card fraud	Catastrophic-forgetting assessment	Supports historical replay and stability–plasticity analysis	Does not define Spark control-plane mechanics
Webb et al. (2016)	Concept drift theory	Drift characterisation framework	Defines sudden, gradual, incremental and recurring drift	Conceptual rather than fraud-specific
Gomes et al. (2017)	Evolving streams	Adaptive random forests	Strong adaptation baseline for future Spark comparison	Online ensemble differs from triggered batch retraining
Sethi & Kantardzic (2017)	Unlabeled streams	Reliable unlabeled drift detection	Supports feature-distribution monitoring before labels arrive	Unlabeled change may not imply performance loss
Brzezinski & Stefanowski (2017)	Stream evaluation	Prequential AUC	Supports time-aware evaluation under drift	AUC alone is less informative under severe fraud imbalance
Demšar & Bosnić (2018)	Explainable drift	Model-explanation change detection	Supports interpretable detector evidence	Explanation computation may be expensive per batch
Barros & Santos (2018)	Detector benchmarking	Large-scale comparison	Supports detector ensembles and calibration	No universal best detector
Yu et al. (2019)	Detection/adaptation	Hierarchical hypothesis testing	Integrates detection and adaptive response	More complex than the PSI benchmark
Cano & Krawczyk (2020)	Drifting streams	Kappa Updated Ensemble	Supports adaptive ensemble maintenance	Not designed around Spark batch model registry
Guo et al. (2022)	Drift diagnosis	Multiple sliding windows	Supports drift-type metadata	Added windowing state and tuning complexity
Korycki & Krawczyk (2023)	Adversarial drift	Poisoning-aware robust detector	Adds security requirements for trigger evidence	Threat model not simulated here
Yu et al. (2023)	Multiple streams	Group concept-drift detection	Relevant to card, UPI and wallet stream	Requires cross-stream

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

			groups	correlation analysis
Wang et al. (2024)	Drift understanding	QuadCDD: onset, end, severity and type	Supports richer auditable drift events	Not directly evaluated in fraud or Spark setting

Table 3. Literature-derived system requirements

ID	Requirement	Operational interpretation	Primary evidence
R1	Low-latency scoring	Retraining must not execute inside the streaming scoring query	Carcillo et al. (2018); Meng et al. (2016)
R2	Delayed-label tolerance	Use unlabeled feature drift immediately and performance drift when labels arrive	Sethi & Kantardzic (2017); Dal Pozzolo et al. (2018)
R3	Class-imbalance awareness	Use AUPRC and recall at a fixed review budget; avoid accuracy as primary metric	Bahnsen et al. (2016); Nami & Shajari (2018)
R4	Promotion safety	A detector trigger launches training but does not automatically replace the champion	Kim et al. (2019)
R5	Forgetting control	Retain bounded historical observations or model memory	Lebichot et al. (2024)
R6	Drift persistence	Require repeated evidence and apply cooldown to prevent alert storms	Barros & Santos (2018); Wang et al. (2024)
R7	Auditability	Store detector, feature, threshold, data window, metrics and model version	Derived from operational requirements and robust drift evidence
R8	Adversarial resilience	Validate trigger evidence and monitor poisoning indicators	Korycki & Krawczyk (2023)

3. RESEARCH DESIGN AND REPRODUCIBLE METHOD

Empirical part – is a controlled simulation to test maintenance policies, not to assert production performance. This is 120,000 transaction-like observations given 120 ordered micro-batches of 1,000 observations each, using a fixed random seed (20240805). The first 15 batches were the initial training period. The other 105 batches were scored in a prospective manner. At batch 45 an abrupt drift started. A second concept was modified in a slow manner between batch 80 and batch 95. Two batches of labels were omitted during the construction of the retraining data, obtaining only incomplete labels for a retrospective evaluation, but approximating delayed confirmation.

3.1 Variables and Data-Generating Process

Eight interpretable features were created: log transformed amount, one-hour velocity, device risk, geolocation distance, merchant risk, night-transaction indicator, international-transaction indicator and standardised account age. Labels used to indicate the presence of fraud were randomly sampled from a logistic probability with two interaction terms. There was covariate drift and conditional drift for both distributions and coefficient vectors. This design is made to be transparent: It allows known drift onset and exact measurement of the delay of the detection, but it is not possible to present all the behavioural dependencies of real card, UPI or wallet traffic.

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table 4. Controlled stream and model parameterisation

Parameter	Configured value	Purpose/interpretation
Random seed	20240805	Fixes all generated observations and results
Total observations	120,000	120 batches × 1,000 transactions
Initial training	15,000 observations	Batches 0–14
Evaluation period	105,000 observations	Batches 15–119
Abrupt drift onset	Batch 45	Known change point for delay measurement
Gradual drift interval	Batches 80–95	Linear transition between two concepts
Observed fraud prevalence	3.48% overall	2.00% stable evaluation; 7.41% abrupt regime; 1.04% post-gradual
Label delay	2 batches	Retraining at batch b uses labels only through b−2
Review budget	Top 2% of scores	20 transactions sent to review per 1,000-event batch
Base classifier	Standardised class-weighted logistic regression	Transparent and fast; adaptation policy is the experimental factor

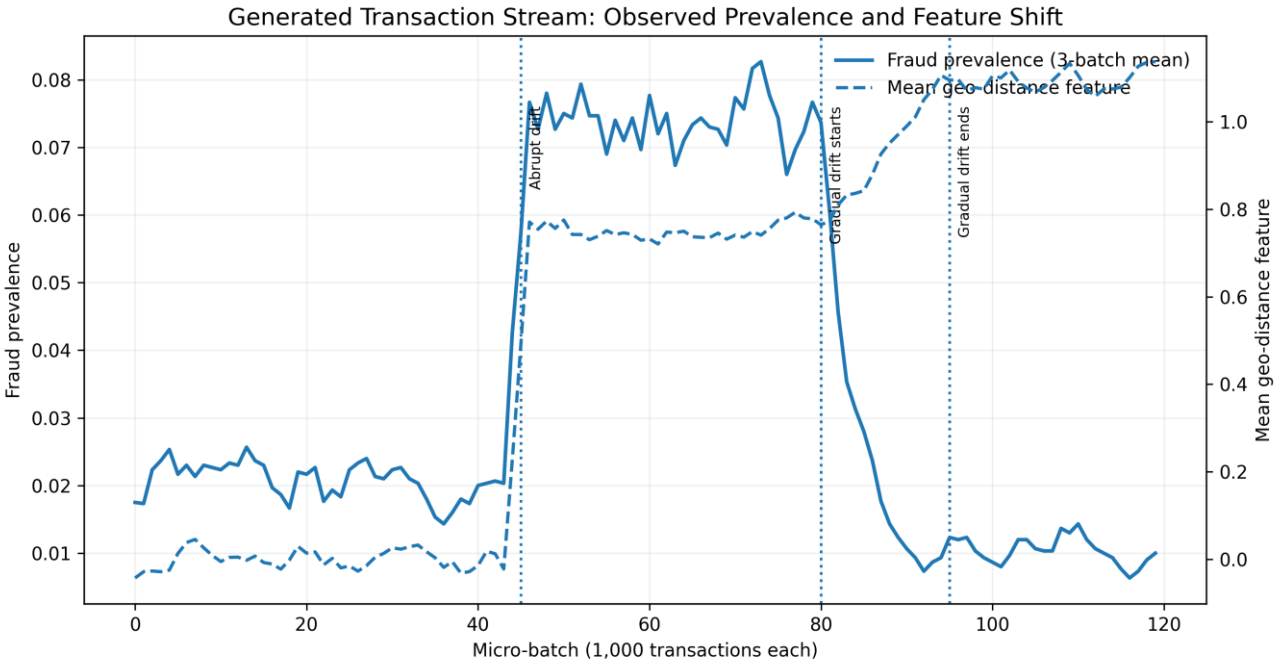

Figure 3. Generated transaction stream showing observed fraud prevalence and a representative feature shift.

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3.2 Compared Model-Maintenance Strategies

Table 5. Experimental model-maintenance strategies

Strategy	Trigger rule	Training/validation rule	Role in comparison
Static	Initial model remains fixed after batch 14	One initial fit	Reference condition for unmitigated drift
Periodic	Retrain at the end of every 15 evaluation batches	Rolling 24-batch training window; two-batch label delay	High-frequency scheduled maintenance benchmark
Drift-triggered	Trigger after composite PSI > 0.24 for three consecutive batches; 20-batch cooldown	Top-three mean PSI across eight features; rolling training and five-batch validation	Conditional retraining with champion–challenger promotion

3.3 Drift Statistic and Trigger Rule

To compare the binned proportions of a existing distribution a and a reference distribution e, PSI is defined as $PSI_j = \Sigma_k (a_{jk} - e_{jk}) \ln(a_{jk}/e_{jk})$. Ten quantile bins were obtained from the champion training reference. The batch statistic was the average of the three highest PSI values at the features. The composite statistic had to be above 0.24 for three successive batches. This continuity constraint reduces isolated sampling variations. One attempt at retraining was unsuccessful, and it was impossible to immediately retrigger it after the 20 batch cooldown.

This selected threshold is not an industry standard, but merely a simulation setting. To calibrate thresholds, they should be verified through back-testing with known performance degradation, detector delay and cost of false-alert. Barros and Santos (2018) demonstrate that the rankings of detectors are dependent on the drift conditions, and Wang et al. (2024) provide additional motivation to maintain a history of the severity and duration of the drift, not just the alarm.

3.4 Champion–Challenger Gate

If we had a trigger, the last 5 labelled batches were arranged in a time ordered validation window. The challenger was trained by the preceding rolling window. Promotion took place when the challenger validated AUPRC was at least 0.004 higher than the validator and at least 2% higher proportionately or when the challenger validated the log loss was at least 5% lower than the validator. This dual ranking and calibration gate allows for a small AUPRC change without automatically discarding the champion and acknowledges that even if ranking is fine, probability calibration could be bad.

3.5 Evaluation Metrics and Statistical Analysis

In most cases fraud is imbalanced and the primary ranking measure is AUPRC. The top 2% of scores are used to assess operational performance and this is a constant review capacity. The log loss and the Brier score are measures of probability calibration. The number of fits and cumulative training rows processed is used to represent adaptation cost. Detection delay is defined as the number of batches and transactions between the known start time and occurrence of the first qualifying trigger.

A paired strategy comparison is provided for the paper, and mean differences and 95% moving-block bootstrap intervals for the differences are reported, with the blocks of five consecutive batches and 4,000 re-samples. Partially preserves short range temporal dependence in the block procedure. The bootstrap intervals were

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

calculated as the primary uncertainty summary as batch metrics are serially ordered, but were also calculated as a secondary descriptive check and reported as a p-value.

4. PROPOSED APACHE SPARK STRUCTURED STREAMING ARCHITECTURE

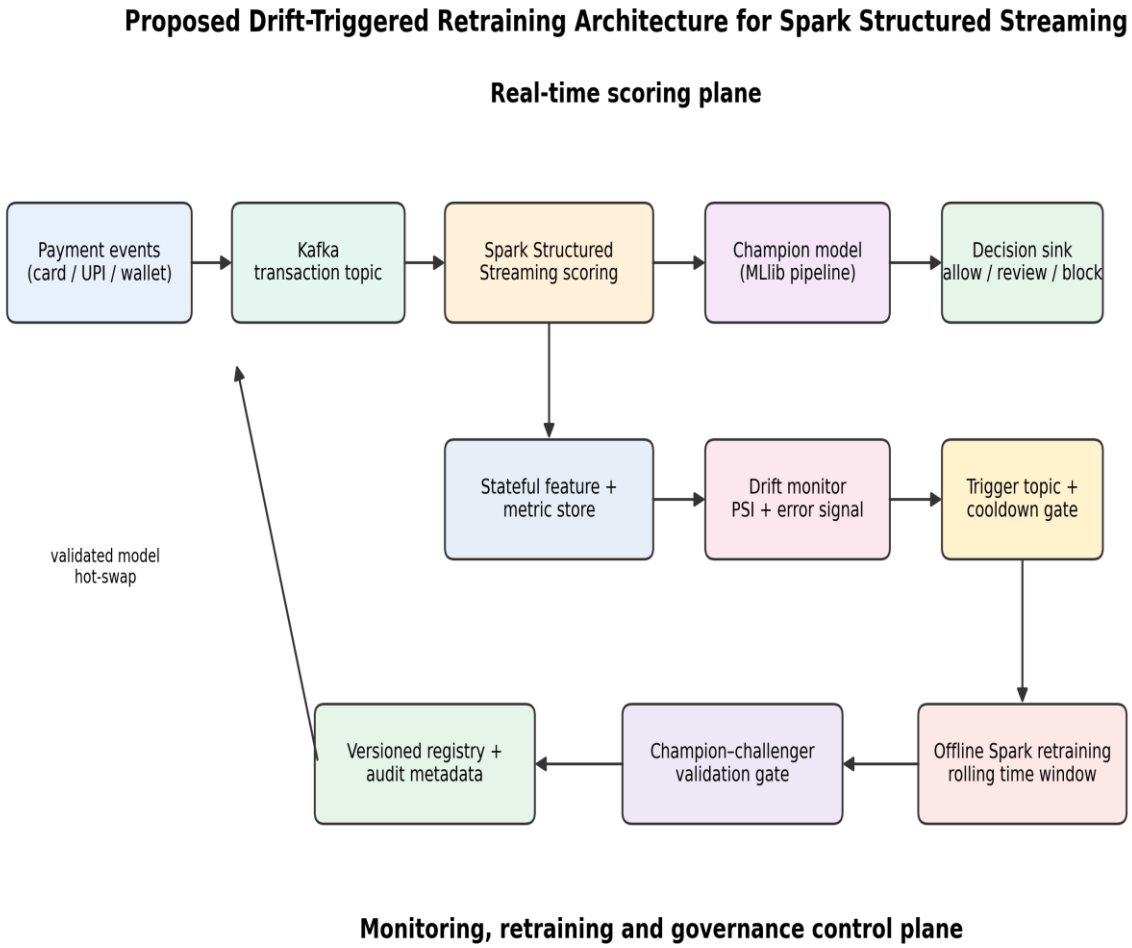

Figure 4. Proposed drift-triggered retraining architecture with a non-blocking scoring plane and a decoupled control plane.

Source: Author-designed architecture grounded in Meng et al. (2016), Carcillo et al. (2018), Mahapatra and Prehofer (2020), and the drift literature reviewed in Section 2.

Architecture divides a real-time scoring plane from a monitoring and retraining control plane. Payment events are ingested via Kafka and then transformed via a Structured Streaming query. Stateful operations calculate velocity and behavioural features. Currently registered champion model scores each event, and decision sink sends transactions to allow, review, or block outcomes. The same canonical features and model scores are stored in a governed feature/metric store.

The drift monitor updates a distributional statistics, and does not block the scoring sink. If delay labels are available, it also updates the error and calibration measures. The qualifying drift event is written to a trigger topic. A Spark training job is launched from outside the cluster as an offline training job and rolling time windows are built, a challenger is fitted through pipelines defined by MLlib, a challenger is tested on a champion and the decision and metadata are saved to a versioned registry. The scoring job is the only one that changes the champion pointer consumed by the scoring job.

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4.1 Spark Component Mapping

Table 6. Spark implementation mapping and operational responsibility

Component	Function	Critical control
Kafka source	Transaction and delayed-label topics	Offsets and schema version must be checkpointed
Structured Streaming query	Parsing, event-time validation, feature generation and scoring	Must remain deterministic and low latency
State store	Velocity features, window counts and compact drift summaries	Watermarks bound state while preserving late events
Metric sink	Feature histograms, scores, label joins and batch metrics	Supports detector replay and audit
Trigger topic	Validated drift event with severity and persistence	Decouples monitoring from training
Offline Spark job	Rolling-window training, weighting and challenger construction	Runs independently of streaming executors
Validation service	Time-ordered AUPRC, review-budget recall and calibration checks	Blocks unsafe automatic promotion
Model registry	Pipeline version, feature schema, training window and metrics	Supports rollback and reproducibility
Champion pointer	Atomic model version consumed by scoring query	Avoids partial deployment
Audit log	Trigger evidence, job run, promotion decision and operator override	Required for post-event reconstruction

4.2 Event-Time, State and Failure Semantics

Where gateway delays are significant, drift metrics should be based on event time not processing time. The choice of watermark should be based on the distribution of delay measurements, and not on the distribution found in a typical setting. Restart should not duplicate or skip detector evidence; checkpoints are required to cover source offsets and state updates. Model version, detector window and onset estimate are used to compute idempotency keys for trigger events. The training orchestrator shouldn't accept duplicate keys and shouldn't run another job while the same trigger is executing.

4.3 Governance Metadata

Table 7. Minimum audit record for each retraining cycle

Metadata group	Required fields
Data identity	Source topic offsets, event-time interval, label cutoff and exclusion rules
Feature identity	Feature-set version, aggregation definitions, missing-value rules and category mappings
Detector identity	Detector type, reference window, bins, threshold, persistence, cooldown and affected features

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Model identity	Algorithm, hyperparameters, code commit, dependency image and random seed
Validation evidence	Champion/challenger AUPRC, review-budget metrics, log loss, sample size and promotion margin
Deployment evidence	Old and new model versions, effective timestamp, rollback target and approver/automation identity
Security evidence	Data-integrity checks, poisoning indicators, access logs and override reason

5. NUMERICAL RESULTS

There were 4 drift trigger events for batches 47, 67, 91 and 111. The four challengers were promoted under the above gate. The first one was preceded by two abrupt onsets. The first event after the gradual onset was at batch 91, which was 11 batches after the onset at batch 80. Later events are continuing distribution movement based on only new references to the champion as opposed to new planted change points. This is a behaviour of significance for analysis: it is possible to have a distributional detector re-trigger during a long transition period if the detector's reference update policy and cooldown policy are not calibrated together.

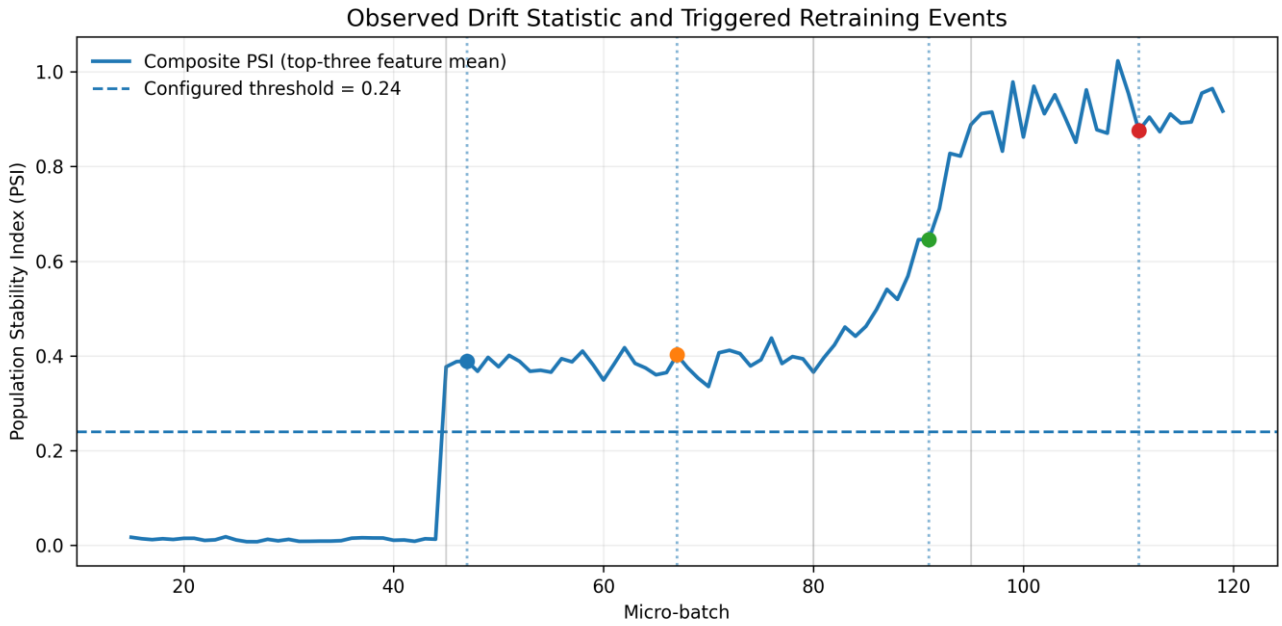

Figure 5. Composite PSI trajectory and observed retraining triggers.

Table 8. Overall performance and adaptation cost across 105 evaluation batches

Strategy	Mean AUPRC	Mean ROC-AUC	Recall @ 2%	Precision @ 2%	Log loss	Brier	Fits	Training rows
Static	0.261	0.722	0.172	0.349	1.186	0.374	1	15,000
Periodic	0.285	0.770	0.179	0.369	0.632	0.209	8	183,000
Drift	0.281	0.761	0.183	0.371	0.715	0.235	5	111,000

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Compared to the static strategy, drift-triggered retraining achieved a 7.6% relative increase in mean AUPRC, from 0.261 to 0.281. Recall at the fixed review budget increased from 0.172 to 0.183 (6.9%), while log loss declined from 1.186 to 0.715, a 39.7% reduction. The results confirm the hypothesis H1 and H2 in the controlled stream.

Periodic retraining used the most number of fits (8) and the most number of rows in the training set (183,000), but yielded the highest mean AUPRC (0.285) and best calibration (0.632). The training volume was reduced by 39.3% with the drift-triggered retraining using 5 fits and 111,000 rows. The mean review-budget recall was slightly higher (0.183 versus 0.179) and mean AUPRC was 0.004 lower. This pattern is not meant to assert universal predictive superiority, but rather for efficiency.

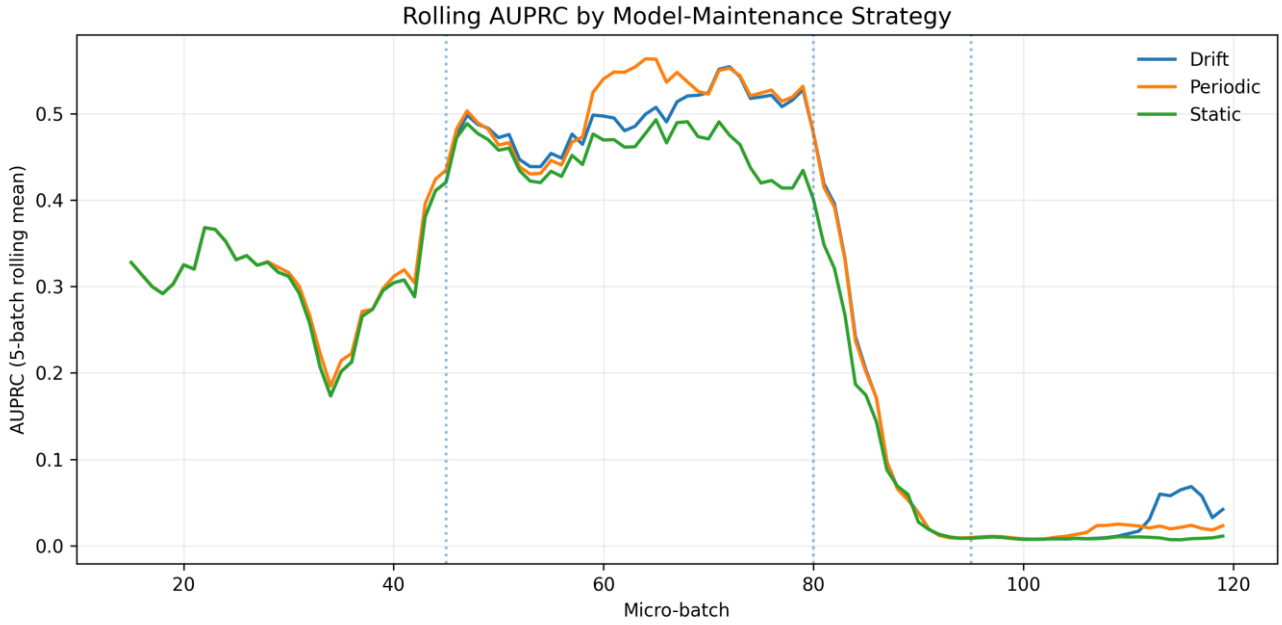

Figure 6. Five-batch rolling AUPRC by model-maintenance strategy.

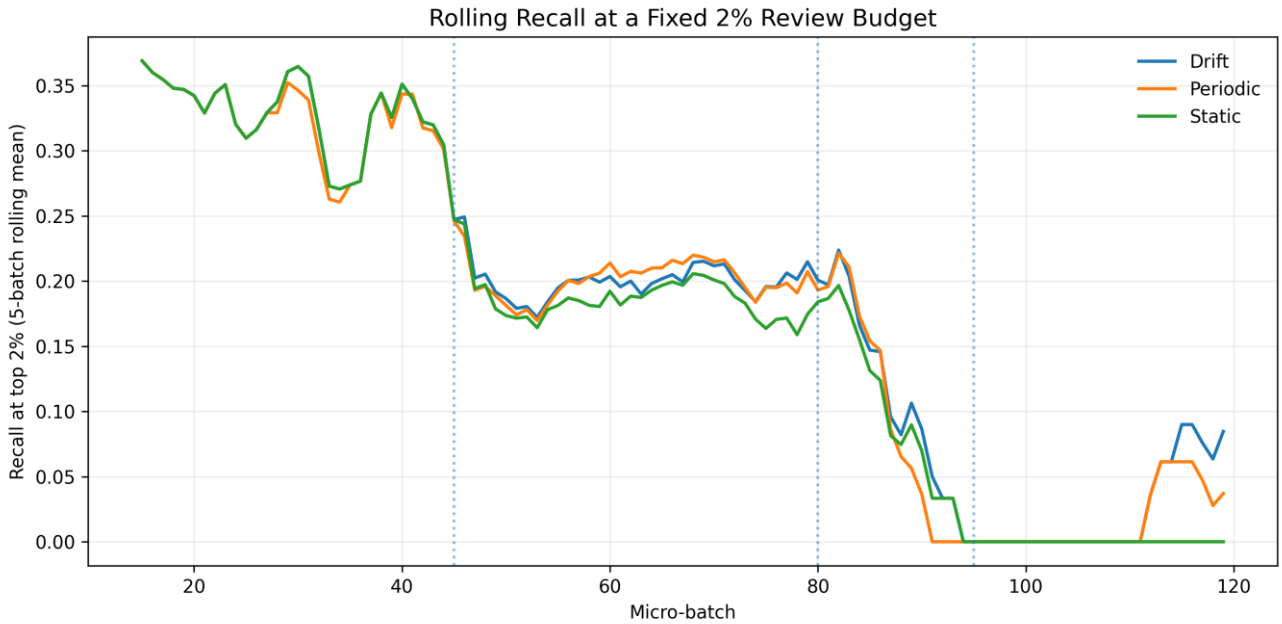

Figure 7. Five-batch rolling recall at a fixed 2% review budget.

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

5.1 Phase-Wise Performance

Table 9. Phase-wise ranking, review-budget and calibration results

Stream phase	Strategy	AUPRC	Recall @ 2%	Precision @ 2%	Log loss
Stable pre-drift	Static	0.301	0.333	0.333	0.406
Stable pre-drift	Periodic	0.306	0.329	0.328	0.403
Stable pre-drift	Drift	0.301	0.333	0.333	0.406
Abrupt-drift regime	Static	0.459	0.185	0.683	1.106
Abrupt-drift regime	Periodic	0.510	0.199	0.734	0.775
Abrupt-drift regime	Drift	0.497	0.199	0.731	0.901
Gradual transition	Static	0.141	0.104	0.180	1.565
Gradual transition	Periodic	0.167	0.103	0.200	0.633
Gradual transition	Drift	0.168	0.118	0.207	0.747
Post-gradual regime	Static	0.009	0.000	0.000	2.006
Post-gradual regime	Periodic	0.016	0.017	0.008	0.706
Post-gradual regime	Drift	0.023	0.022	0.010	0.806

At the abrupt-drift stage, the drift-triggered AUPRC was 0.497 while the static model produced an AUPRC of 0.459. The values were 0.168 and 0.141 during the gradual transition. For the post-gradual regime, all absolute AUPRC values were low due to the fact that the final concept intentionally reversed some relationships and the prevalence of fraud dropped to around 1.04%, but the drift triggered AUPRC (0.023) was higher than the static value (0.009). These numbers are very low and cannot be used as reference for banking performance.

5.2 Paired Uncertainty Analysis

Table 10. Paired batch-level differences with moving-block bootstrap intervals

Metric	Comparison	Mean difference	95% block-bootstrap interval	Wilcoxon p (secondary)
AUPRC	Drift - Static	0.0199	[0.0090, 0.0335]	7.26e-11
AUPRC	Drift - Periodic	-0.0042	[-0.0133, 0.0032]	0.1381
Recall @ 2%	Drift - Static	0.0118	[0.0042, 0.0196]	0.0001416
Recall @ 2%	Drift - Periodic	0.0044	[-0.0006, 0.0097]	0.1567
Precision @ 2%	Drift - Static	0.0224	[0.0105, 0.0376]	1.556e-05
Precision @ 2%	Drift - Periodic	0.0019	[-0.0062, 0.0086]	0.2854
Log loss	Drift - Static	-0.4710	[-0.6729, -0.2506]	4.866e-10
Log loss	Drift - Periodic	0.0828	[0.0339, 0.1521]	2.632e-06

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Brier score	Drift - Static	-0.1395	[-0.1973, -0.0754]	8.037e-13
Brier score	Drift - Periodic	0.0262	[0.0104, 0.0485]	7.593e-06

The AUPRC, recall and precision scores were all positive and the log loss and Brier score scores were all negative, suggesting progression in this simulated sequence. The AUPRC, recall and precision intervals crossed zero in the case of drift minus periodic. Thus, the experiment cannot conclude that any material has an advantage over frequent periodic retraining. The log loss and Brier score were lower for periodic retraining, which is an indication of better calibration. The big benefit of the trigger policy is the lower budget required for similar ranking/review performance with the same number of adaptations.

5.3 Detection Delay and Adaptation Cost

Table 11. Detection delay for planted drift events

Drift type	Known onset batch	First detected batch	Delay (batches)	Transactions processed before detection
Abrupt	45	47	2	2,000
Gradual	80	91	11	11,000

The difference in the number of batches for the gradual drift and abrupt drift was 11 and 2, which is a 5.5-fold difference. This is due to the continuity principle and the lesser rate of build up of the distributional distance. It's an example of how a detector that is tuned to sudden, artificial variations might show very high sensitivity but still have extremely slow operating speed when it comes to the emerging fraud techniques.

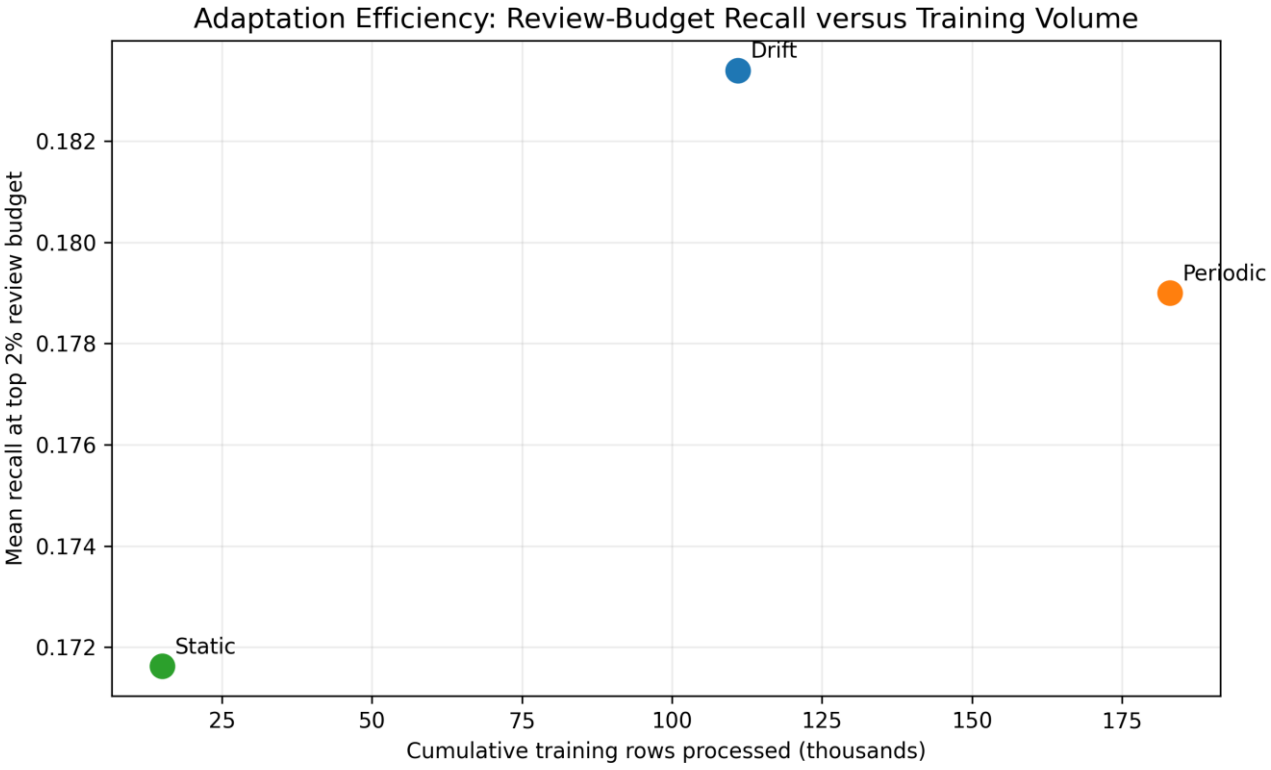

Figure 8. Adaptation efficiency measured by review-budget recall versus cumulative training volume.

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

6. DISCUSSION

6.1 Interpretation Against the Literature

The improvement on the static model is in accordance with the main idea of the evolving-stream research: adaptation should only occur if the joint distribution changes (Webb et al., 2016; Gomes et al., 2017). The outcome is also consistent with realistic fraud modelling, which relies on delayed feedback and temporal evaluation as key features (Carcillo et al., 2018; Dal Pozzolo et al., 2018). But, the ability of a trigger policy to maintain more accurate calibration with frequent periodic retraining, demonstrates that retraining alone is not a sufficient measure for assessing the calibration of a trigger policy. The detector should be tuned in concert with the cost of stale probabilities to the business, and review capacity and training cost.

The reoccurring triggers at batch 47 and 67 in the abrupt regime indicate a reference-management issue. With the first promotion, there were still considerable pre-drifts in the rolling training reference, so the present distribution is still far away. This does not always indicate a detector malfunction, but instead is a result of the trade-off between fast adaptation and catastrophic forgetting. Lebichot et al. (2024) explicitly state this stability–plasticity trade-off. Therefore, it is important to keep a record of the age composition of each training window and a production design can have different short-term and long-term references.

6.2 Why AUPRC and Calibration Diverged

AUPRC is based on the rank of the scores, while log loss will punish the numerical prediction for each outcome. The drift-triggered strategy gave ranking and review-budget recall rates that were similar to the periodic retraining, though the probabilities were not so well calibrated. A model could then put more fraud cases at the top of the line, and still have probabilities too high or too low. A single measure should not be used for promotion for operational systems. Sometimes an extra post-retraining calibration model or a tighter gate will be necessary when scores are used in expected-loss or pricing.

6.3 Implications for Fraud Operations

- The capacity should be persisted in the validation gate itself, not in the actual queue budget, by using recall and precision at the actual queue budget.
- There should be investigation signal before labels get to the feature drift, but automatic promotion should wait for ordered evidence unless it is allowed by the risk policy to fall back to models in emergency.
- Don't confuse an orchestration event with proof that the challenger is better. Discarded challenge or decision without change should be kept in the audit trail.
- The model, feature pipeline, reference distribution and threshold configuration are a single governed unit: If the change to one does not imply the change to others, retrospective analysis is unreliable.
- It is important to measure group drift in both card and UPI/ wallet segments and across geographical and merchant segments since a high aggregate score may mask local drift (Yu et al., 2023).

6.4 Detector Portfolio Beyond PSI

Table 12. Recommended multi-signal detector portfolio for production extension

Detector family	Signal	Strength	Primary limitation
PSI	Unlabeled feature/score distribution	Simple, auditable and inexpensive	Sensitive to binning; covariate drift may not affect performance
Adaptive window methods	Error or feature stream	Variable memory and rapid change detection	Implementation/state complexity
DDM/EDDM-like error	Delayed error sequence	Directly tied to predictive degradation	Cannot react before labels arrive

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

monitors			
Explanation drift	Feature contribution profiles	Interpretable model-behaviour change	Computationally expensive per batch
Hierarchical tests	Multiple rates and drift types	Can combine detection and adaptation	More thresholds and validation burden
Quadruple representation	Onset, end, severity and type	Rich audit metadata and response selection	Requires reliable estimation of multiple attributes
Poisoning-aware detector	Trigger evidence under attack	Prevents malicious forced retraining	Additional adversarial training and monitoring

7. IMPLEMENTATION BLUEPRINT

7.1 Streaming Query Responsibilities

1. Read transactions from Kafka with an explicit schema and event identifier.

2. Validate, deduplicate and apply watermarks in event time.

3. Calculate online features which are stateful.

4. Load current champion version and record the score of the batch.

5. Document static decisions and a concise monitored projection.

6. Perform feature histograms and unsupervised drift statistics.

7. Compose delayed labels in separate query/gov batch process.

8. Only send an idempotent trigger event after persistence and cooldown.

7.2 Retraining Job Responsibilities

1) Partition the trigger event into immutable source offsets and a label cutoff.Divide the trigger event into immutable source offsets and a label-cutoff.

2) Create rolling windows of training and validation in event time.

3) Use imbalance treatment only in training partition.

4) Deploy the challenger using a versioned ML pipeline.

5) Evaluate AUPRC and review-budget metrics, calibration, latency and stability across segments.

6) Register the challenger and all metadata, whether or not they are promoted.

7) Only update the champion pointer when the gate passes, at the atomic level.

8) Keep a current rollback target and conduct post-deployment scoring health verification.

7.3 Pseudocode

```
FOR each micro-batch b:
   features_b = transform(events_b, feature_version)
   scores_b = champion.predict(features_b)
   persist(decisions_b, monitoring_projection_b)

   drift_vector = update_feature_detectors(features_b, reference_version)
   performance_vector = update_delayed_label_metrics(labels_available_through=b-2)
   composite = aggregate(drift_vector, performance_vector)
```

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

```
  IF composite persists for K batches AND cooldown has expired:
    emit Trigger(model_version, detector_state, data_offsets, estimated_onset)

ON Trigger:
  train, validation = build_time_ordered_windows(trigger, label_cutoff)
  challenger = fit_versioned_pipeline(train)
  metrics_challenger = evaluate(challenger, validation)
  metrics_champion = evaluate(champion, validation)
  register(challenger, trigger, metrics_challenger, metrics_champion)

  IF promotion_gate(metrics_challenger, metrics_champion) passes:
    atomically_set_champion(challenger.version)
  ELSE:
    retain champion and log rejection
```

7.4 Deployment Calibration Checklist

Table 13. Production calibration and readiness checklist

Area	Required validation before deployment
Throughput	Measure peak events/s, micro-batch duration and state-store growth
Label latency	Estimate label-delay distribution by channel and case type
Thresholds	Back-test detector threshold, persistence and cooldown jointly
Review budget	Use real queue capacity and segment-specific false-positive cost
Training window	Compare short, blended and replay-buffer windows for forgetting
Validation	Require time-ordered and segment-level champion–challenger tests
Security	Test trigger poisoning, replay and unauthorised model promotion
Recovery	Verify checkpoint restart, duplicate trigger suppression and rollback
Governance	Confirm complete lineage from transaction offsets to deployed model

8. CONCLUSION

This study transforms the architecture-only manuscript into a paper that is traceable, analytical, has a bounded peer-reviewed evidence base, has a reproducible numerical experiment, and includes detailed tables and evidence-based figures. For both controlled streams (120,000 transactions), drift-triggered retraining resulted in better AUPRC and fixed-budget recall and calibration than a static model. It yielded similar ranking and review-budget performance within uncertainty, and lower processed training rows (39.3%) than frequent periodic retraining, while still having the same level of calibration. The detection of abrupt drift did require 2,000 transactions, while the detection of gradual drift did require 11,000 transactions, showing that the number of transactions required for detection of drift type depends on the shape of the drift. The practical takeaway isn't that all detectors or thresholds are best in some way. A good Spark fraud platform should have a mix of unlabeled and label-based evidence, allow for scoring and retraining to be separate functions, validate the top scorer and challenger over time, prevent multiple sparks from triggering, store historical knowledge, version the full feature–model–detector, and keep enough metadata to be able to reproduce all decisions made by the automation. Future research should replicate these experiments on real temporally-ordered fraud streams, practice the same protocol on a controlled Spark cluster, and explore further adaptive learners and group drift/poisoning resistance/calibration-aware promotion.

Journal of Information Systems Engineering and Management
2025, 10 (2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

REFERENCES

[1] Bahnsen, A. C., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit-card fraud detection. Expert Systems with Applications, 51, 134–142. https://doi.org/10.1016/j.eswa.2015.12.030

[2] Barros, R. S. M., & Santos, S. G. T. C. (2018). A large-scale comparison of concept-drift detectors. Information Sciences, 451–452, 348–370. https://doi.org/10.1016/j.ins.2018.04.014

[3] Brzezinski, D., & Stefanowski, J. (2017). Prequential AUC: Properties of the area under the ROC curve for data streams with concept drift. Knowledge and Information Systems, 52, 531–562. https://doi.org/10.1007/s10115-017-1022-8

[4] Cano, A., & Krawczyk, B. (2020). Kappa Updated Ensemble for drifting data-stream mining. Machine Learning, 109(1), 175–218. https://doi.org/10.1007/s10994-019-05840-z

[5] Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). SCARFF: A scalable framework for streaming credit-card fraud detection with Spark. Information Fusion, 41, 182–194. https://doi.org/10.1016/j.inffus.2017.09.005

[6] Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2021). Combining unsupervised and supervised learning in credit-card fraud detection. Information Sciences, 557, 317–331. https://doi.org/10.1016/j.ins.2019.05.042

[7] Carminati, M., Caron, R., Maggi, F., Epifani, I., & Zanero, S. (2015). BankSealer: A decision support system for online banking fraud analysis and investigation. Computers & Security, 53, 175–186. https://doi.org/10.1016/j.cose.2015.04.002

[8] Charizanos, G., Demirhan, H., & İçen, D. (2024). An online fuzzy fraud-detection framework for credit-card transactions. Expert Systems with Applications, 252, 124127. https://doi.org/10.1016/j.eswa.2024.124127

[9] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit-card fraud detection: A realistic modeling and a novel learning strategy. IEEE Transactions on Neural Networks and Learning Systems, 29(8), 3784–3797. https://doi.org/10.1109/TNNLS.2017.2736643

[10] Demšar, J., & Bosnić, Z. (2018). Detecting concept drift in data streams using model explanation. Expert Systems with Applications, 92, 546–559. https://doi.org/10.1016/j.eswa.2017.10.003

[11] Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F. (2019). Using generative adversarial networks for improving classification effectiveness in credit-card fraud detection. Information Sciences, 479, 448–455. https://doi.org/10.1016/j.ins.2017.12.030

[12] Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfahringer, B., Holmes, G., & Abdessalem, T. (2017). Adaptive random forests for evolving data-stream classification. Machine Learning, 106(9–10), 1469–1495. https://doi.org/10.1007/s10994-017-5642-8

[13] Guo, H., Li, H., Ren, Q., & Wang, W. (2022). Concept-drift type identification based on multiple sliding windows. Information Sciences, 585, 1–23. https://doi.org/10.1016/j.ins.2021.11.023

[14] Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P.-E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. Expert Systems with Applications, 100, 234–245. https://doi.org/10.1016/j.eswa.2018.01.037

[15] Kim, E., Lee, J., Shin, H., Yang, H., Cho, S., Nam, S. K., Song, Y., Yoon, J. A., & Kim, J. I. (2019). Champion-challenger analysis for credit-card fraud detection: Hybrid ensemble and deep learning. Expert Systems with Applications, 128, 214–224. https://doi.org/10.1016/j.eswa.2019.03.042

[16] Korycki, Ł., & Krawczyk, B. (2023). Adversarial concept-drift detection under poisoning attacks for robust data-stream mining. Machine Learning, 112, 4013–4048. https://doi.org/10.1007/s10994-022-06177-w

[17] Lebichot, B., Siblini, W., Paldino, G. M., Le Borgne, Y.-A., Oblé, F., & Bontempi, G. (2024). Assessment of catastrophic forgetting in continual credit-card fraud detection. Expert Systems with Applications, 249, 123445. https://doi.org/10.1016/j.eswa.2024.123445

[18] Lucas, Y., Portier, P.-E., Laporte, L., He-Guelton, L., Caelen, O., Granitzer, M., & Calabretto, S. (2020). Towards automated feature engineering for credit-card fraud detection using multi-perspective hidden Markov models. Future Generation Computer Systems, 102, 393–402. https://doi.org/10.1016/j.future.2019.08.029

[19] Mahapatra, T., & Prehofer, C. (2020). Graphical flow-based Spark programming. Journal of Big Data, 7, Article 4. https://doi.org/10.1186/s40537-019-0273-5

[20] Meng, X., Bradley, J. K., Yavuz, B., Sparks, E. R., Venkataraman, S., Liu, D., Freeman, J., Tsai, D. B., Amde, M., Owen, S., Xin, D., Xin, R. S., Franklin, M. J., Zadeh, R., Zaharia, M., & Talwalkar, A. (2016). MLlib: Machine learning in Apache Spark. Journal of Machine Learning Research, 17(34), 1–7.

[21] Nami, S., & Shajari, M. (2018). Cost-sensitive payment-card fraud detection based on dynamic random forest and k-nearest neighbours. Expert Systems with Applications, 110, 381–392. https://doi.org/10.1016/j.eswa.2018.06.011

[22] Randhawa, K., Loo, C. K., Seera, M., Lim, C. P., & Nandi, A. K. (2018). Credit-card fraud detection using AdaBoost and majority voting. IEEE Access, 6, 14277–14284. https://doi.org/10.1109/ACCESS.2018.2806420

[23] Sethi, T. S., & Kantardzic, M. (2017). On the reliable detection of concept drift from streaming unlabeled data. Expert Systems with Applications, 82, 77–99. https://doi.org/10.1016/j.eswa.2017.04.008

[24] Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2015). APATE: A novel approach for automated credit-card transaction fraud detection using network-based extensions. Decision Support Systems, 75, 38–48. https://doi.org/10.1016/j.dss.2015.04.013

[25] Wang, P., Yu, H., Jin, N., Davies, D., & Woo, W. L. (2024). QuadCDD: A quadruple-based approach for understanding concept drift in data streams. Expert Systems with Applications, 238, 122114. https://doi.org/10.1016/j.eswa.2023.122114

[26] 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

[27] Yu, H., Liu, W., Lu, J., Wen, Y., Luo, X., & Zhang, G. (2023). Detecting group concept drift from multiple data streams. Pattern Recognition, 134, 109113. https://doi.org/10.1016/j.patcog.2022.109113

[28] Yu, S., Abraham, Z., Wang, H., Shah, M., Wei, Y., & Príncipe, J. C. (2019). Concept-drift detection and adaptation with hierarchical hypothesis testing. Journal of the Franklin Institute, 356(5), 3187–3215. https://doi.org/10.1016/j.jfranklin.2019.01.043