Open Access Library Journal
2026, Volume 13, e16096
ISSN Online: 2333-9721
ISSN Print: 2333-9705

Selective Updating for Customer Segmentation under Concept Drift

Maria El Maaroud, Chi Zhang

School of Computer Science and Artificial Intelligence, Hubei University of Technology, Wuhan, China
Email: Elmaaroudmaria@gmail.com

How to cite this paper: El Maaroud, M. and Zhang, C. (2026) Selective Updating for Customer Segmentation under Concept Drift. *Open Access Library Journal*, **13**: e16096.
https://doi.org/10.4236/oalib.1116096

Received: September 29, 2026
Accepted: November 8, 2026
Published: November 11, 2026

Copyright © 2026 by author(s) and Open Access Library Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

Scheduled refitting can replace an effective customer segmentation even when a newly fitted model offers little improvement. We examine selective replacement for batchwise customer segmentation under concept drift, comparing BIRCH and MiniBatchKMeans under static, periodic, quality-only, and reward-based policies on Instacart, Ta-Feng, and Olist. The reward policy has the highest reported mean deployed silhouette in all six dataset-algorithm combinations; its advantage over quality-only is 0.0045 - 0.0170 silhouette points. Five supplied paired seed effects per combination are positive for reward versus each baseline, with seed-bootstrap intervals excluding zero. These same-snapshot comparisons do not establish temporally disjoint gains or isolate the reward's stability term. The raw run record is not yet archived; label-aligned stability and denominator-aware event analyses are not reported.

Subject Areas

Computer Science, Data Mining, Machine Learning

Keywords

Customer Segmentation, Concept Drift, Model Replacement, BIRCH, MiniBatchKMeans

1. Introduction

Customer segmentation is a model-management problem as well as a clustering problem. A static model can become stale under distribution change, while unconditional or scheduled replacement can introduce unnecessary churn. Concept-drift research separates recognition of change from the decision to adapt a deployed learner [1]. This paper studies the second decision: after a candidate segmentation model has been fitted, when should it replace the current model?

The evaluation uses recency and frequency with a dataset-specific third feature. Ta-Feng and Olist use monetary value, following the RFM interpretation connected to customer-value analysis [2]. Instacart contains no observed price field: each product row starts with unit quantity and a simulated unit weight of one, stored in the UnitPrice field. Injected drift changes both quantity and this weight. We call the sum of quantity times simulated unit weight drift-weighted item quantity; it equals product-row count only before perturbation and does not measure actual spending. Clustering quality still depends on representation and the structure sought in the data [3]; improved internal geometry does not by itself demonstrate retention, response, churn reduction, future spend, intervention lift, or revenue impact.

Figure 1 summarizes the evaluation workflow. Candidate fitting and model deployment are separate decisions; the reported scores assess the deployed model on cumulative customer-feature snapshots.

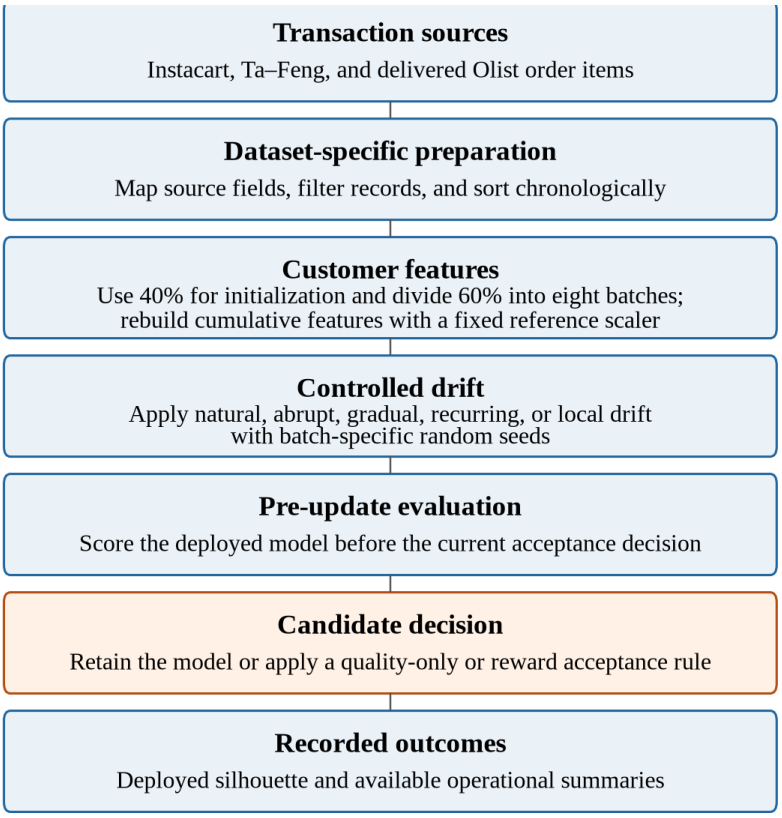


Figure 1. Evaluation workflow. A fitted candidate replaces the deployed model only when its acceptance rule permits an update.

We compare Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [4] and MiniBatchKMeans (MBK) [5] under a static policy, blind periodic refitting, two selective candidate-acceptance gates, and an MBK-only incremental comparator. The chronological ordering follows the prequential principle of evaluating the current model before learning from newly available data [6]. Sil-

houette measures internal cluster separation [7], and the adjusted Rand index (ARI) measures partition agreement while remaining invariant to numeric cluster labels [8].

This study separates candidate fitting from deployment and compares quality-only and continuity-aware acceptance against static and periodic policies. It reports paired seed-level reward contrasts for all three baselines. Label-aligned accepted-update stability, denominator-aware event accounting, disjoint temporal evaluation, and end-to-end cost measurement remain proposed analyses, not findings of this study. The policy comparisons do not establish superiority over dedicated stream-clustering methods.

2. Related Work

Dedicated stream-clustering methods maintain evolving summaries or structures. CluStream stores online micro-clusters for subsequent clustering [9]; DenStream handles density-based evolving streams with noise [10]; D-Stream uses decayed density grids [11]; ClusTree supports anytime micro-cluster maintenance [12]; and StreamKM++ forms compact weighted summaries [13]. These methods address stream representation and cluster evolution directly. The present study instead augments conventional batchwise clustering with a candidate-deployment decision. Dedicated stream-clustering methods are not included in the empirical comparison and therefore remain important baselines for future work.

Recent work emphasizes that stream-clustering performance depends jointly on the summary structure, temporal window, outlier treatment, and refinement strategy. A broad empirical study found that no single design choice dominates across workloads [14], while MOStream dynamically reconfigures these components as stream characteristics change [15]. These results motivate evaluating deployment gates against modern adaptive stream baselines rather than only against fixed batch policies.

Change monitoring is also distinct from replacement. Adaptive Windowing (ADWIN) changes its effective window after detected distribution change [16], while kernel two-sample testing provides another family of distribution comparisons [17]. The implementation uses Kolmogorov-Smirnov and Wasserstein diagnostics, but its acceptance gates are driven by cluster-quality deterioration and candidate evaluation. A monitor alarm therefore neither guarantees replacement nor demonstrates that replacement is beneficial.

Partition agreement does not fully establish clustering validity. Hennig [18] shows that stability and clustering quality are separate concerns. The Massive Online Analysis framework provides a broader basis for reproducible stream evaluation [19]. This paper uses ARI as one component of a deployment rule, not as proof of semantic or business stability.

3. Datasets and Preprocessing

The experiment uses the Instacart market-basket release [20], the Ta-Feng grocery

transactions [21], and the Olist Brazilian e-commerce release [22]. The reported Instacart and Ta-Feng experiments use 500,000 transaction lines each. The standard Olist order-items release has 112,650 source rows, of which 110,197 remain after the documented delivered-order and validity filters; a 500,000-row input cap does not increase the number of available order items. Transaction-line, customer, order, and RFM-record counts are distinct units. Table 1 reports the transaction-line counts.

Table 1. Transaction-line records used in each experiment.

Dataset	Transaction rows
Instacart	500,000
Ta-Feng	500,000
Olist	110,197

The described Olist conversion joins the order-items file to the orders and customers files from the standard release, retains delivered orders, and removes records with missing required identifiers, timestamps, or prices and records with nonpositive product prices. It does not intentionally resample or duplicate order-item rows and uses no additional transaction source. The stated 112,650 raw and 110,197 retained row counts refer to this conversion, not to 500,000 distinct Olist items. The input file hashes and full source-data manifest for the reported run are not archived, so the run's source files and filtering outcomes cannot yet be independently traced. The dataset URLs in the references identify public releases, not immutable versions of the files used in that run.

Source-to-Feature Mapping

Table 2 describes the transformation from each source dataset to the common transaction schema. The reconstructed fields are material to interpretation: Instacart has synthetic chronology and no monetary information; Ta-Feng reconstructs baskets at customer-day resolution; and Olist includes freight in item value.

For Instacart, orders are sorted within each user_id by order_number. Missing days_since_prior_order values are set to zero, and the customer-relative intervals are cumulatively summed. Each order receives the synthetic timestamp 1 January 2017 plus that cumulative number of days plus its order_hour_of_day (missing hours are set to zero). The same arbitrary 2017 origin is used for every customer; it is not an observed calendar date or a known alignment of different customers' purchasing histories. Joined product rows are then ordered by this synthetic timestamp, order ID, and add-to-cart position before the chronological initial/stream split. Consequently, recency and batch membership reflect the shared synthetic timeline rather than an observed cross-customer chronology.

For customer i at batch t, the cumulative representation is

Table 2. Dataset-specific conversion into the common transaction schema.

Source	Invoice/basket ID	Customer ID	Timestamp	Quantity and value	Limitation
Instacart	order_id → InvoiceNo	user_id → CustomerID	Synthetic reconstructed date → InvoiceDate	Initially Quantity = 1 and simulated UnitPrice = 1 per product row; both change under injected drift	Chronology is synthetic; drift-weighted item quantity is neither observed spend nor unchanged product-row count.
Ta–Feng	Customer ID plus transaction date → reconstructed basket ID	CUSTOMER_ID → CustomerID	TRANSACTION_DT → InvoiceDate	AMOUNT → Quantity; SALES_PRICE/AMOUNT → UnitPrice	Purchases by one customer on the same date form one basket, which can merge multiple same-day visits.
Olist	order_id → InvoiceNo	customer_unique_id → CustomerID	order_purchase_timestamp → InvoiceDate	Quantity = 1 per item; product price plus freight → UnitPrice	Only delivered orders are retained; freight is included in the monetary feature.

$$x_{i,t} = \left[\log\left(1+R_{i,t}\right), \log\left(1+F_{i,t}\right), \log\left(1+V_{i,t}\right)\right]^{\mathrm{T}} \quad (1)$$

where recency $R_{i,t}$ is time since the latest purchase, frequency $F_{i,t}$ counts unique orders or reconstructed baskets, and $V_{i,t}$ is the cumulative third feature. For Ta-Feng and Olist, $V_{i,t}$ is monetary value. For Instacart, $V_{i,t} = \Sigma(Q_\ell P_\ell)$ over rows $\ell \in H_{i,t}$ is drift-weighted item quantity: Q_ℓ is row quantity and P_ℓ is a simulated unit weight stored as UnitPrice. Both equal one initially and are perturbed on drifted rows. This sum equals product-row count before perturbation, but is neither row count nor observed spending afterward. The analysis fits a standard scaler on the initial 40% and holds it fixed across the eight subsequent cumulative snapshots. Section 4.4 defines the corresponding window-local representation for temporal evaluation. Here $H_{i,t}$ is the set of product rows for customer i included up to batch t.

4. Methodology

4.1. Initial Model Selection

The described initial-model selection considers $k \in \{2, 3, 4, 5, 6\}$ on the first 40% cumulative customer-feature snapshot. The selected per-seed values, BIRCH threshold grid, tie-breaking rule, and original validation record are not available with the reported results; their precise selection cannot be verified here. A disjoint temporal evaluation would select and freeze hyperparameters using chronological validation entirely within the initial training period. The present results do not test cluster splits, merges, or changing cluster counts.

The implementation uses scikit-learn components [23]. Incremental evaluation is included only for MBK. Although the scikit-learn BIRCH interface also exposes partial_fit [24], a matched streaming-BIRCH comparison is outside the scope of the present experiments.

4.2. Synthetic Drift Protocol

Let $z_t \in [0,1]$ be the drift intensity for batch t. Natural drift uses $z_t = 0$ throughout. Abrupt drift uses full intensity from batch 3 onward. Gradual drift uses zero intensity before batch 3 and

$$z_t = \min\left(1,(t-2)/4\right) \tag{2}$$

thereafter, giving 0.25, 0.50, 0.75, and 1.00 at batches 3 - 6. Recurring drift has full intensity at batches 3, 4, 6, and 7, with onsets at batches 3 and 6. Local drift has full intensity from batch 3 onward but affects only a stable hash-selected 20% of customer identifiers.

For medium severity, quantity Q and unit price P are transformed as

$$\tilde{Q} = \max\left(1, Q\left[1 - z_t\left(1 - 0.70\right)\right] + \varepsilon_Q\right) \tag{3}$$

$$\tilde{P} = \max\left(0.01, P\left[1 - z_t\left(1 - 0.75\right)\right] + \varepsilon_P\right) \tag{4}$$

Here the simulated noise terms ε_Q and ε_P have normal distributions with standard deviations $0.60 z_t$ and $0.20 z_t$, respectively. Total price is recomputed as QP. The batch-specific generator seed is

$$10{,}000 + \text{model_seed} + 100t \tag{5}$$

No rounding is applied after adding Gaussian noise, so Q is a continuous simulated feature, not a literal purchased-unit count. Instacart starts with the placeholder $P = 1$; the reported drift scenarios perturb both Q and P, recompute QP, and thereby change the evaluated third feature even though the number of product rows is unchanged. Changes to P are synthetic feature perturbations, not economic price drift. No quantity-only Instacart condition is reported.

Table 3 makes the five scenario schedules explicit. In particular, it distinguishes permanent abrupt and local changes from the temporary recurring pattern and the monotone gradual ramp used to evaluate replacement decisions.

Table 3. Synthetic-drift intensity by stream batch. Local uses the abrupt intensity schedule on 20% of customers.

Scenario	1	2	3	4	5	6	7	8
Natural	0	0	0	0	0	0	0	0
Abrupt	0	0	1	1	1	1	1	1
Gradual	0	0	0.25	0.50	0.75	1	1	1
Recurring	0	0	1	1	0	1	1	0
Local	0	0	1	1	1	1	1	1

4.3. Same-Snapshot Acceptance Rules

Let D_t denote the cumulative standardized customer-feature snapshot (monetary RFM for Ta-Feng and Olist; drift-weighted item quantity for Instacart) at batch t. The current pre-update silhouette is

$$\bar{p}_t = S\left(M_{t-1}, D_t\right) \tag{6}$$

The notation p_{t-1} denotes the post-decision deployed silhouette recorded at

the previous batch. The deterioration gate compares that previous deployed score with the current pre-update score:

$$g_t = 1\left[p_t^- < 0.995 p_0 \text{ or } p_{t-1} - p_t^- > 0.003 \right]. \quad (7)$$

When $g_t = 1$, a candidate M_t^c is fitted on D_t. Its same-snapshot silhouette is $p_t^c = S\left(M_t^c, D_t\right)$, and a_t is the ARI between current and candidate assignments. Quality-only acceptance requires $p_t^c > p_t^-$. Reward acceptance uses

$$R_{\text{keep}} = 0.85 p_t^- + 0.10, \quad (8)$$

$$R_{\text{update}} = 0.85 p_t^c + 0.10 a_t - 0.05 \rho, \quad \rho = 0.005. \quad (9)$$

The term ρ is a normalized switching regularizer, not a measured business or infrastructure cost. Its fixed reward contribution is 0.05 × 0.005 = 0.00025, equivalent to approximately 0.000294 silhouette units after division by 0.85. A sensitivity analysis would vary the stability weight and regularizer, report the fraction of decisions changed, and select settings on training or validation scenarios rather than final test scenarios; no such analysis is reported here.

The static policy never replaces the initial model. The periodic policy is scheduled every four batches, but the implementation permits an update only when a following batch exists. Consequently, in the eight-batch protocol, batch 4 is the only eligible scheduled refit and batch 8 is evaluation-only. The two selective policies apply the quality-only and reward rules above subject to the same final-batch restriction. The MBK incremental comparator calls partial_fit each batch without the same deployment gate.

4.4. Proposed Temporal Evaluation

The reported same-snapshot evaluation fits a candidate on cumulative snapshot D_b, evaluates it on D_b, and uses that score in the acceptance decision. A post-decision silhouette on D_t is therefore a selection score rather than an unbiased estimate of temporal generalization. A cumulative next-batch comparison on D_{t+1} likewise retains substantial overlap with candidate-training history. The following disjoint-window procedure has not been evaluated for the reported policy scores.

The proposed procedure would use a one-window decision delay and a window-local outcome. Let W_t contain only transactions arriving in chronological window t, and let X_t be its standardized customer-feature set. For each customer active in W_b recency would run from that customer's latest transaction to the end of W_b while frequency and value would use only baskets and transactions inside W_t. Thus, Equation (1) would be applied within each window rather than over cumulative history. Customers without transactions in W_t would have no window-local vector; their number and the silhouette denominator would be reported. A reference scaler would be fitted once on window-local vectors from the initial training period and shared by active and candidate models.

The initial period would be divided into windows of the same duration, and candidates would be fitted on historical customer-window vectors $X_{1:t}$. Repeated

observations of one customer in different windows would be separate training instances. The gate would compare active and candidate models on X_{t+1}, and an accepted candidate would become active for W_{t+2}. A customer would be "returning" if its identifier occurred before W_{t+1} and "new" otherwise; a new customer's vector would use only W_{t+1}. The proposed outcome would be measured on X_{t+2}, not the gate window W_{t+1}. With T stream windows, only origins $t \leq T-2$ would yield a fully evaluated candidate.

Table 4 distinguishes the proposed fitting, gating, and outcome windows. Unlike the cumulative comparisons in the reported results, the outcome window would not be used to select its deployment state.

Table 4. Proposed information flow for one rolling origin; no disjoint-window outcomes are reported.

Stage	Available data	Operation
Fit at t	Initial period, $W_{1:t}$	Fit candidate
Gate at $t+1$	New W_{t+1} only	Keep or accept
Outcome at $t+2$	New W_{t+2} only	Score deployed policy

A secondary as-of cumulative analysis could rebuild returning-customer features using history available at each cutoff; it would measure history-overlapping persistence rather than disjoint generalization. A future report should distinguish window-local, new-customer, returning-customer, and cumulative gains.

For this proposed analysis, silhouette would be defined only when evaluated assignments contain at least two nonempty clusters and fewer clusters than observations. A candidate with an undefined silhouette would be rejected. If the active score were undefined but the candidate score valid, either selective gate would accept the candidate; if both were undefined, the model would be retained and the decision recorded as lacking a valid quality comparison rather than assigned an artificial score. Numerical gate or reward comparisons would apply only when both scores are valid. A future run record should specify the distance metric, silhouette subsample size and seed, and whether active and candidate scores use the same sampled customers.

4.5. Proposed Aligned Stability and Event Denominators

Because cluster identifiers are arbitrary, a future migration analysis would align labels before comparing assignments. For old labels y and candidate labels z, a contingency matrix and the Hungarian algorithm would find the relabeling π with maximum overlap. The proposed minimum error distance is

$$\mathrm{MED}(y,z) = \min_{\pi} (1/n) \sum_{i=1}^{n} 1\left[y_i \neq \pi(z_i) \right]. \tag{10}$$

When cluster counts differ, the contingency matrix would be padded to $K = \max(k_y, k_z)$ rows and columns. The proposed normalization is

$$\mathrm{NMED}(y,z) = \mathrm{MED}(y,z) / (1 - 1/K) \quad \text{if} \quad K > 1; 0 \text{ if } K = 1. \tag{11}$$

This scales the $1-1/K$ worst-case assignment-error bound to one. ARI measures permutation-invariant agreement, but does not by itself give a customer migration rate.

A future event record should distinguish candidate agreement (ARI and MED whenever a candidate is evaluated) from accepted-update stability (only accepted candidates). Post-alignment migration would use customers present in both relevant windows; no-update continuity would track the active model without treating retention as a perfect update. No-candidate and rejected rows should not be assigned ARI one or migration zero. Each mean would require its numerator and eligible denominator, distinguishing accepted updates, eligible decisions, tracks with at least one update, and future-window comparisons. These event-level results are not available here.

4.6. Proposed Reward Sensitivity and Runtime

For a future temporal evaluation, let y_{t+1}^{keep} and y_{t+1}^{cand} denote active and candidate assignments on X_{t+1}, and define $m_{t+1} = \text{NMED}\left(y_{t+1}^{\text{keep}}, y_{t+1}^{\text{cand}}\right)$. An interpretable proposed reward is

$$\begin{aligned} R_{\text{keep}} &= S\left(M_{t-1}, X_{t+1}\right), \\ R_{\text{update}} &= S\left(M_t^c, X_{t+1}\right) - \lambda_m m_{t+1} - \lambda_c \tilde{C}. \end{aligned} \tag{12}$$

For measured end-to-end candidate cost C, the proposed normalized cost would be $C = \min(1, C/C_{0.95})$, with $C_{0.95}$ estimated only from training or validation origins and then frozen. Values of λ_m and λ_c would be selected outside final test scenarios; their decision-change rates are not reported here.

A future runtime analysis should separate feature construction, active prediction, candidate fitting, candidate evaluation, decision logic, and reporting; it should include rejected candidates, total batch time, and peak memory. Only model-fit times are presented here, not end-to-end latency.

4.7. Proposed Preprocessing Sensitivity and Comparators

Source-specific feature constructions remain a sensitivity-analysis target rather than results reported here. For Instacart, this would compare recency-frequency alone with a true product-row-count feature and with the reported drift-weighted item quantity. Ta-Feng would compare customer-day baskets with the finest source transaction grouping available; Olist would compare product price alone with product price plus freight. Such comparisons should be reported separately from the present policy rankings.

The reported comparisons comprise static, periodic-4, quality-only, and reward policies for both algorithms, plus MBK partial_fit. Always-refit, BIRCH partial_fit, CluStream, and DenStream are proposed additional comparators, not reported results. Future comparisons should share chronological windows and eligible-decision accounting; when cluster counts differ, assignment agreement should use ARI or NMED instead of raw label equality.

5. Results

5.1. Comparison Units and Aggregation

The comparison unit is a matched dataset-algorithm-scenario-seed track, not an individual batch row. Batch observations within a track are not treated as independent replicates. This paired-unit treatment follows the general caution against treating correlated experimental observations as independent comparisons [25]. For each dataset, algorithm, scenario, and seed, we first calculate the track-level mean deployed silhouette, then calculate reward-minus-baseline differences using the same seed and scenario. Primary comparisons are rewards versus static, periodic-4, and quality-only.

For each primary reward contrast, the five paired seed effects in Tables 5-10 average matched scenario differences within a seed. Each 95% percentile bootstrap interval resamples these five seed effects, not individual batches or scenarios. These intervals describe variation among the supplied seeds, not uncertainty over new datasets or drift scenarios; adjusted hypothesis tests are not reported.

Within each dataset-algorithm-policy-scenario-seed track, deployed silhouettes are averaged over batches with defined scores (up to eight). Policy means give equal weight to available track means across five scenarios and five seeds (up to 25), without weighting by batch size or customer count. Undefined batch scores are omitted, not set to zero. The source record does not establish whether any tracks contain missing batch scores; the number of included tracks is therefore not independently verifiable.

Table 5. Mean deployed silhouette on fixed-reference-scaled customer features: monetary RFM for Ta-Feng and Olist, and drift-weighted item quantity for Instacart. Higher is better.

Dataset	Algorithm	Static	Periodic-4	Quality-only	Reward	Incremental
Instacart	BIRCH	0.8795	0.8920	0.9185	0.9230	—
Instacart	MiniBatchKMeans	0.3690	0.5120	0.7150	0.7320	0.3980
Ta-Feng	BIRCH	0.5060	0.5210	0.5580	0.5640	—
Ta-Feng	MiniBatchKMeans	0.5060	0.5150	0.5780	0.5890	0.5180
Olist	BIRCH	0.8635	0.8780	0.9050	0.9120	—
Olist	MiniBatchKMeans	0.4565	0.5630	0.6820	0.6980	0.4750

Table 6. Descriptive reward-policy advantage in mean deployed silhouette. Entries are absolute silhouette-point differences, not percentages.

Dataset	Algorithm	Reward - static	Reward - periodic-4	Reward - quality-only
Instacart	BIRCH	+0.0435	+0.0310	+0.0045
Instacart	MiniBatchKMeans	+0.3630	+0.2200	+0.0170
Ta-Feng	BIRCH	+0.0580	+0.0430	+0.0060
Ta-Feng	MiniBatchKMeans	+0.0830	+0.0740	+0.0110
Olist	BIRCH	+0.0485	+0.0340	+0.0070
Olist	MiniBatchKMeans	+0.2415	+0.1350	+0.0160

Table 7. Paired reward-minus-quality-only differences in mean deployed silhouette (absolute silhouette points). Each seed value averages five matched scenario differences; the 95% percentile interval resamples these five seed effects with replacement (all 5^5 ordered resamples). Positive values favor reward. The intervals quantify seed variation in the supplied summaries only; their six-decimal display does not imply comparable generalization precision.

Dataset	Algorithm	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean [95% interval]
Instacart	BIRCH	0.004261	0.004755	0.004735	0.004008	0.004741	0.004500 [0.004205, 0.004745]
Instacart	MBK	0.016999	0.016200	0.016740	0.016870	0.018191	0.017000 [0.016494, 0.017662]
Ta-Feng	BIRCH	0.005389	0.005962	0.006340	0.006448	0.005861	0.006000 [0.005674, 0.006309]
Ta-Feng	MBK	0.010978	0.011143	0.010467	0.010572	0.011840	0.011000 [0.010611, 0.011447]
Olist	BIRCH	0.007041	0.006487	0.007621	0.006372	0.007479	0.007000 [0.006552, 0.007448]
Olist	MBK	0.015289	0.014504	0.015889	0.016745	0.017573	0.016000 [0.015095, 0.016951]

Table 8. Paired reward-minus-static differences in mean deployed silhouette (absolute silhouette points). Seed effects and 95% intervals are computed as in Table 7.

Dataset	Algorithm	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean [95% interval]
Instacart	BIRCH	0.043420	0.041763	0.040339	0.046166	0.045812	0.043500 [0.041525, 0.045475]
Instacart	MBK	0.386071	0.397450	0.346127	0.361716	0.323636	0.363000 [0.340248, 0.385752]
Ta-Feng	BIRCH	0.062028	0.061359	0.060158	0.054684	0.051771	0.058000 [0.054271, 0.061386]
Ta-Feng	MBK	0.082053	0.078478	0.086376	0.088294	0.079799	0.083000 [0.079721, 0.086279]
Olist	BIRCH	0.052238	0.044205	0.043488	0.052192	0.050377	0.048500 [0.045153, 0.051847]
Olist	MBK	0.252140	0.218601	0.265358	0.221600	0.249801	0.241500 [0.225909, 0.256959]

Table 9. Paired reward-minus-periodic-4 differences in mean deployed silhouette (absolute silhouette points). Seed effects and 95% intervals are computed as in Table 7.

Dataset	Algorithm	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean [95% interval]
Instacart	BIRCH	0.032959	0.030637	0.030534	0.032838	0.028032	0.031000 [0.029494, 0.032450]
Instacart	MBK	0.202092	0.237649	0.233832	0.222537	0.203890	0.220000 [0.206900, 0.233100]
Ta-Feng	BIRCH	0.043684	0.042452	0.045695	0.037886	0.045283	0.043000 [0.040279, 0.045128]
Ta-Feng	MBK	0.074552	0.065475	0.078747	0.074128	0.077098	0.074000 [0.069530, 0.077428]
Olist	BIRCH	0.035279	0.035602	0.034344	0.031751	0.033024	0.034000 [0.032711, 0.035221]
Olist	MBK	0.121782	0.127540	0.139388	0.137404	0.148886	0.135000 [0.126455, 0.142790]

Table 10. Supplied operational summaries for BIRCH with the reward policy. Deployed and Static are mean silhouette; Diff. is their difference. Next gain is a cumulative next-snapshot counterfactual summary, not a disjoint outcome. The eligible denominators for Positive, Updates, Migration, Update ARI, and Drift cannot be recovered from the available run record; these entries must not be read as comparable event-level rates or accepted-update stability. Fit s records model fitting only, not end-to-end latency.

Dataset	Deployed	Static	Diff.	Next gain	Positive	Updates	Migration	Update ARI	Drift	Fit s
Instacart	0.9230	0.8795	+0.0435	0.26	65.3%	16.7%	0.0198	0.9712	33.8%	0.0028
Ta-Feng	0.5640	0.5060	+0.0580	0.23	61.2%	14.4%	0.0041	0.9921	34.2%	0.0045
Olist	0.9120	0.8635	+0.0485	0.29	69.8%	24.1%	0.0223	0.9534	38.6%	0.0091

5.2. Policy Comparisons

The policy aggregates use same-snapshot gating, cumulative next-batch comparisons, unaligned migration, and incomplete candidate timing. The underlying run record for the supplied paired seed summaries has not been archived; accepted-event denominators, label-aligned stability, and disjoint temporal outcomes are unavailable.

The reward policy has the highest mean deployed silhouette in all six dataset–algorithm cells (Table 5). Table 6 gives its absolute differences from static, periodic-4, and quality-only; Tables 7-9 give matched seed contrasts and bootstrap intervals.

Against static, the reward policy's absolute mean improvement ranges from 0.0435 to 0.3630; against quality-only it is 0.0045 - 0.0170 silhouette points. In all six cells, all five supplied seed effects for each of the three contrasts are positive and their seed-bootstrap intervals exclude zero. This is consistency among the supplied seeds under same-snapshot evaluation, not evidence that the reward's stability or cost term caused the improvement.

Figure 2 shows the four policy means by dataset and algorithm. The policies are discrete alternatives; connecting points serves only to guide comparison, not to represent a temporal trajectory.

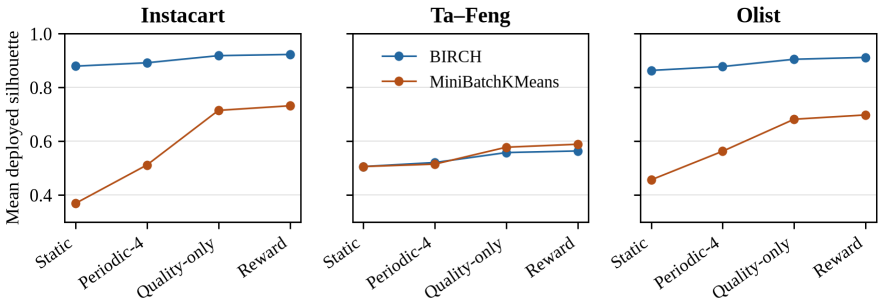


Figure 2. Mean deployed silhouette by replacement policy. Blue lines denote BIRCH and orange lines denote MiniBatchKMeans; each line connects discrete policy means for one dataset-algorithm combination from Table 5, not successive time points.

Table 10 retains supplied BIRCH summaries for context, not an event-level comparison. The unavailable eligible denominators prevent interpreting Positive, Updates, Migration, Update ARI, and Drift as conditionally comparable rates.

The next-gain values compare cumulative snapshots with overlapping customer histories rather than a disjoint outcome window; they cannot establish a future causal benefit. The displayed fit times exclude feature construction, transformation, scoring, diagnostics, input/output, and other pipeline work. The available record does not establish how rejected candidate fits contributed to these summaries, so the values cannot represent end-to-end latency.

Figure 3 displays the small reward-quality-only differences on a scale suited to those contrasts; the paired seed intervals are reported separately in Table 7.

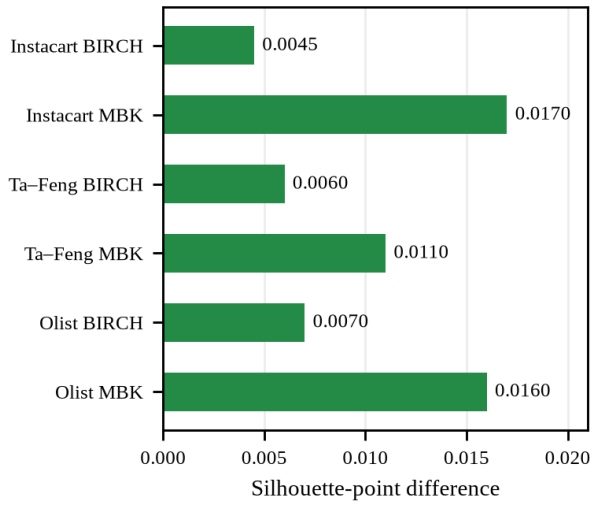


Figure 3. Reward-quality-only silhouette difference by dataset-algorithm combination. Bars reproduce the descriptive values in **Table 6**; they do not show confidence intervals.

6. Discussion

The comparisons distinguish fitting a candidate from deploying it. The reward gate improves internal silhouette relative to quality-only in the supplied summaries, but the small gap and absence of an ablation leave the stability term's contribution unresolved. A deployment rule should consider both partition changes and candidate cost; the present summaries do not measure either as a calibrated operational trade-off.

The fixed regularizer contributes just 0.00025 to the reward, so it cannot be read as a measured switching cost. Aligned migration and measured end-to-end cost would allow that trade-off to be tested without assuming that a silhouette gain also improves customer or business outcomes.

7. Limitations

Instacart's synthetic chronology and drift-weighted item quantity limit comparison with monetary RFM datasets: its third feature measures neither observed spend nor the unchanged number of product rows. Ta-Feng's customer-day baskets may merge visits, and Olist includes freight in item value. A window-local analysis would answer a different question from cumulative customer-value segmentation and has not been run here.

Cluster counts are fixed after initial selection, so splits and merges are not tested. The selected hyperparameters, source-file hashes, and original run-level record are not archived with the manuscript. Without that record, the seed effects cannot be traced to individual tracks, and operational denominators and fit-time scope cannot be resolved. Disjoint temporal outcomes, aligned accepted-update stability, dedicated streaming baselines, and downstream behavioral outcomes are also absent.

8. Conclusion

Reward-gated replacement has the highest reported mean deployed silhouette in all six dataset-algorithm comparisons. Its advantage over quality-only is small, and the supplied paired-seed intervals describe only five seeds under same-snapshot evaluation. A traceable run record and temporally disjoint outcomes are needed before claiming a more general deployment benefit.

Author Contributions

Maria El Maaroud designed and conducted the study and drafted the manuscript. Chi Zhang supervised the research and reviewed the manuscript.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. and Bouchachia, A. (2014) A Survey on Concept Drift Adaptation. *ACM Computing Surveys*, **46**, 1-37. https://doi.org/10.1145/2523813

[2] Fader, P.S., Hardie, B.G.S. and Lee, K.L. (2005) RFM and CLV: Using Iso-Value Curves for Customer Base Analysis. *Journal of Marketing Research*, **42**, 415-430. https://doi.org/10.1509/jmkr.2005.42.4.415

[3] Jain, A.K. (2010) Data Clustering: 50 Years beyond K-Means. *Pattern Recognition Letters*, **31**, 651-666. https://doi.org/10.1016/j.patrec.2009.09.011

[4] Zhang, T., Ramakrishnan, R. and Livny, M. (1996) BIRCH: An Efficient Data Clustering Method for Very Large Databases. *Proceedings of the* 1996 *ACM SIGMOD International Conference on Management of Data—SIGMOD*'96, Montreal, 4-6 June 1996, 103-114. https://doi.org/10.1145/233269.233324

[5] Sculley, D. (2010) Web-Scale K-Means Clustering. *Proceedings of the* 19*th International Conference on World Wide Web*, Raleigh, 26-30 April 2010, 1177-1178. https://doi.org/10.1145/1772690.1772862

[6] Dawid, A.P. (1984) Present Position and Potential Developments: Some Personal Views: Statistical Theory: The Prequential Approach. *Journal of the Royal Statistical Society. Series A* (*General*), **147**, 278. https://doi.org/10.2307/2981683

[7] Rousseeuw, P.J. (1987) Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, **20**, 53-65. https://doi.org/10.1016/0377-0427(87)90125-7

[8] Hubert, L. and Arabie, P. (1985) Comparing Partitions. *Journal of Classification*, **2**, 193-218. https://doi.org/10.1007/bf01908075

[9] Aggarwal, C.C., Yu, P.S., Han, J. and Wang, J. (2003) A Framework for Clustering Evolving Data Streams. In: Freytag, J.C., et al., Eds., *Proceedings* 2003 *VLDB Conference*, Elsevier, 81-92. https://doi.org/10.1016/b978-012722442-8/50016-1

[10] Cao, F., Estert, M., Qian, W. and Zhou, A. (2006) Density-Based Clustering over an Evolving Data Stream with Noise. *Proceedings of the* 2006 *SIAM International Conference on Data Mining*, Bethesda, 20-22 April 2006, 328-339. https://doi.org/10.1137/1.9781611972764.29

[11] Chen, Y. and Tu, L. (2007) Density-Based Clustering for Real-Time Stream Data.

Proceedings of the 13*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, 12-15 August 2007, 133-142. https://doi.org/10.1145/1281192.1281210

[12] Kranen, P., Assent, I., Baldauf, C. and Seidl, T. (2010) The ClusTree: Indexing Micro-Clusters for Anytime Stream Mining. *Knowledge and Information Systems*, **29**, 249-272. https://doi.org/10.1007/s10115-010-0342-8

[13] Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C. and Sohler, C. (2012) StreamKM++: A Clustering Algorithm for Data Streams. *ACM Journal of Experimental Algorithmics*, **17**, 2.1-2.30. https://doi.org/10.1145/2133803.2184450

[14] Wang, X., Wang, Z., Wu, Z., Zhang, S., Shi, X. and Lu, L. (2023) Data Stream Clustering: An In-Depth Empirical Study. *Proceedings of the ACM on Management of Data*, **1**, 1-26. https://doi.org/10.1145/3589307

[15] Wang, Z., Wang, X. and Zhang, S. (2024) MOStream: A Modular and Self-Optimizing Data Stream Clustering Algorithm. 2024 *IEEE International Conference on Data Mining* (*ICDM*), Abu Dhabi, 9-12 December 2024,500-509. https://doi.org/10.1109/icdm59182.2024.00057

[16] Bifet, A. and Gavaldà, R. (2007) Learning from Time-Changing Data with Adaptive Windowing. *Proceedings of the* 2007 *SIAM International Conference on Data Mining*, Minneapolis, 26-28 April 2007, 443-448. https://doi.org/10.1137/1.9781611972771.42

[17] Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B. and Smola, A.J. (2012) A Kernel Two-Sample Test. *Journal of Machine Learning Research*, **13**, 723-773.

[18] Hennig, C. (2007) Cluster-Wise Assessment of Cluster Stability. *Computational Statistics & Data Analysis*, **52**, 258-271. https://doi.org/10.1016/j.csda.2006.11.025

[19] Bifet, A., Holmes, G., Kirkby, R. and Pfahringer, B. (2010) MOA: Massive Online Analysis. *Journal of Machine Learning Research*, **11**, 1601-1604.

[20] (2017) InstaCart Online Grocery Basket Analysis Dataset. Kaggle, Dataset 1889830 (Mirror of the 2017 Instacart Release). https://www.kaggle.com/datasets/yasserh/instacart-online-grocery-basket-analysis-dataset/data

[21] (2026) Ta Feng Grocery Dataset. Kaggle, Dataset 22987. https://www.kaggle.com/datasets/chiranjivdas09/ta-feng-grocery-dataset/data

[22] Olist (2026) Brazilian E-Commerce Public Dataset by Olist. Kaggle, Dataset 55151. https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

[23] Pedregosa, F., et al. (2011) Scikit-Learn: Machine Learning in Python. *The Journal of Machine Learning Research*, **12**, 2825-2830.

[24] Scikit-Learn Developers (2026) Birch. Scikit-Learn Documentation. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html

[25] Demšar, J. (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, **7**, 1-30.