International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 7, Issue 2, 117-125, 2026
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V7I2P118

Original Article

RetentionOS: A Closed-Loop Prescriptive Machine Learning Framework for Real-Time Customer Retention

Smeet Patel

Smeet H Patel
Director, Enterprise BI & AI/ML Engineering.

Received On: 08/03/2026 ***Revised On: 02/04/2026*** ***Accepted On: 09/04/2026*** ***Published On: 20/04/2026***

Abstract - *Existing customer retention systems in subscription-based industries rely on open-loop churn prediction that never systematically measures whether interventions caused customers to stay. This paper proposes RetentionOS, a closed-loop prescriptive machine learning framework that integrates churn propensity, pricing sensitivity, and customer lifetime value (CLV) profitability models with a low-latency feature store (AWS DynamoDB, Redis) and a live scoring pipeline, delivering top-3 curated retention offers via API within strict sub-second latency requirements. A champion-challenger traffic split with formal statistical promotion criteria drives continuous evidence-based model evolution. Outcome measurement via daily order processing tables, combined with holdout-based causal attribution, closes the learning loop. The framework contributes mathematical formulation of the decision objective and CLV-weighted retention metric, formal algorithms for arbitration and champion-challenger promotion, a capability comparison against four baseline approaches, and a reproducible ablation methodology. RetentionOS is proposed as a reference framework directly applicable to any high-CLV customer lifecycle requiring real-time intervention and causal learning - telecommunications retention, financial services attrition, subscription software cancellation, and healthcare patient engagement are immediate target domains.*

Keywords *- Closed-Loop Machine Learning, Prescriptive Analytics, Uplift Modeling, Real-Time Decisioning, Customer Retention, Causal Attribution, Champion-Challenger Model Selection, Feature Store Architecture, Heterogeneous Treatment Effects, Machine Learning Operations Mlops.*

1. Introduction

Subscriber churn is among the costliest operational challenges in telecommunications and similar subscription-based industries. In saturated markets where acquisition costs routinely exceed several months of average revenue per user, retaining an existing customer is substantially more economical than replacing one. Despite sustained industry investment in predictive analytics, most production retention systems share a critical structural weakness: they are open-loop. A model scores customers, offers are dispatched, and no systematic mechanism measures whether the intervention caused retention or whether the customer would have stayed regardless.

This paper proposes RetentionOS, a closed-loop prescriptive ML framework designed to address this limitation. Three compounding problems motivate the design. First, existing systems conflate churn propensity with treatability - a high-risk customer is not necessarily responsive to any given offer, and poorly targeted interventions can actively accelerate churn by reminding customers of their option to leave. Second, batch-scored models cannot respond to the highest-value retention moment in the customer lifecycle: the live inbound cancellation call, where agent-mediated intervention is possible within seconds. Third, absent outcome feedback, models continue training on static observational correlations with no signal from what actually happened after recommendations were executed.

RetentionOS addresses all three problems through a five-layer closed-loop architecture: (1) multi-model intelligence extraction, (2) low-latency feature retrieval and live scoring, (3) configurable decisioning engine arbitration, (4) sub-second API delivery to call center agents, and (5) outcome measurement with causal attribution and champion-challenger model governance. This paper proposes RetentionOS as a reference framework for practitioners building or modernizing enterprise retention systems at scale. The framework is grounded in the operational realities of large-scale telecommunications retention, where the author has direct industry experience, and is designed to be directly implementable on commodity cloud infrastructure. Although the worked examples, attribution windows, and intervention types throughout this paper draw on the telecommunications domain, the framework itself is industry-agnostic and is directly applicable to any high-CLV customer lifecycle where real-time intervention and causal learning are required - financial services attrition, subscription software cancellation, and healthcare patient engagement are immediate target domains.

The contributions of this work are: (i) a formally specified closed-loop prescriptive ML framework with

explicit mathematical formulation of the decision objective, the CLV-weighted retention metric, and the champion-challenger promotion criterion; (ii) a low-latency feature store design that separates feature engineering from model scoring, enabling live context-aware inference at sub-second latency; (iii) a daily order processing table methodology for high-fidelity causal outcome measurement without survey dependence; (iv) a statistically governed champion-challenger traffic split mechanism for continuous model evolution; and (v) a reproducible evaluation framework including a capability comparison against four baseline approaches and an ablation methodology with characteristic degradation signatures that practitioners can apply to validate equivalent systems.

The remainder of this paper is organized as follows. Section 2 reviews related work and positions RetentionOS against prior contributions. Section 3 presents the system architecture. Sections 4 through 7 describe the multi-model foundation, low-latency feature store, decisioning engine, and sub-second API serving layer, respectively. Section 8 details the closed-loop learning cycle including outcome measurement, causal attribution, and champion-challenger governance. Section 9 presents the proposed evaluation framework including capability comparison and ablation methodology. Section 10 discusses design considerations, limitations, and future work. Sections 11 and 12 address ethical considerations and reproducibility. Section 13 concludes. A nomenclature table summarizing all mathematical symbols appears at the end of Section 2.

2. Related Work

Customer churn prediction in telecommunications has an extensive literature. Classical approaches applied logistic regression and decision trees to usage and billing features [1]. Subsequent work incorporated gradient boosting ensembles and deep learning for behavioral sequence modeling [2]. A comprehensive review of predictive churn modeling in the telecommunications sector is provided by Zhao et al. [3]. A persistent limitation, however, is that the predominant evaluation framing optimizes prediction accuracy on held-out test sets - a metric that does not translate directly to retention effectiveness, because a high-AUC model can still produce economically wasteful or counterproductive interventions when treatability is not explicitly modeled.

Uplift modeling addresses this gap by estimating individual-level causal treatment effects rather than outcome probabilities. Radcliffe and Surry established the foundational net lift framework [4], and Kunzel et al. systematically compared meta-learner approaches (T-learner, S-learner, X-learner) for heterogeneous treatment effect estimation [5]. Gutierrez and Gerardy provided a broader survey of uplift modeling in practice [6]. The statistical foundations of the counterfactual approach trace to Rosenbaum and Rubin's propensity score methodology [7]. Causal machine learning has been extended by Athey and Imbens through recursive partitioning approaches [8].

Infrastructure for real-time ML serving was formalized in industry by Uber's Michelangelo platform [9], which established the feature store as a core pattern for training-serving consistency. Olson et al. subsequently characterized the architectural space of feature stores across organizations [10]. The broader data management challenges of production ML systems were surveyed by Polyzotis et al. [11]. Online controlled experimentation at scale was characterized by Kohavi et al. [12] and further extended into contextual bandit frameworks by Chapelle and Li [13]. Concept drift in streaming production systems was surveyed by Gama et al. [14]. MLOps patterns for production reliability have been consolidated in recent practitioner literature [15].

RetentionOS synthesizes these threads into a unified framework targeting enterprise-scale retention contexts. It extends prior work in three specific dimensions: (a) the integration of multi-model intelligence (churn, pricing sensitivity, CLV profitability) into a unified decisioning engine rather than relying on a single uplift model; (b) the daily order processing table methodology for outcome measurement, which has not been documented in the academic literature; and (c) a formal champion-challenger governance layer with statistical promotion criteria applied to retention decisioning. Table 1 summarizes this positioning against representative prior work.

Table 1: Positioning Of Retentionos against Representative Prior Work across Five Architectural Dimensions.

Prior Work	Prediction	Prescription	Real-Time	Causal Loop	Multi-Model
Huang et al. [1]	✓	—	—	—	—
De Caigny et al. [2]	✓	—	—	—	—
Radcliffe & Surry [4]	✓	✓	—	partial	—
Kunzel et al. [5]	✓	✓	—	partial	—
Hermann & Del Balso [9]	✓	—	✓	—	—
RetentionOS (this work)	✓	✓	✓	✓	✓

2.1. Nomenclature

Table 2 summarizes the mathematical symbols used throughout the paper for reader reference.

Table 2: Nomenclature of Mathematical Symbols.

Symbol	Meaning
c	Customer identifier
o	Candidate offer
f(c)	Feature vector for customer c retrieved from the Feature Store
O_c	Eligible offer set for customer c
O*(c)	Top-3 prescribed offers for customer c
τ(c, o)	Estimated individual treatment effect (uplift) of offer o on customer c
w_CLV(c)	CLV weight; w_CLV(c) = CLV(c) / CLV_median
e(c, o)	Binary eligibility function; 1 if customer c qualifies for offer o, else 0
R_CLV	CLV-weighted 90-day retention metric (Equation 3)
𝟙[·]	Indicator function; returns 1 if condition holds, 0 otherwise
Σ	Summation operator
ATE	Average Treatment Effect via difference-in-differences (Equation 4)
ρ	Champion-challenger traffic split ratio
σ	Split configuration containing ρ and routing parameters
μ_champ, μ_chall	Mean CLV-weighted retention of champion and challenger cohorts
Δ	Difference between challenger and champion means (μ_chall − μ_champ)
n_min, δ_min	Minimum sample size and minimum effect size for challenger promotion
α	Statistical significance level (0.05)

3. System Architecture

Figure 1 illustrates the five-layer RetentionOS architecture. The design is governed by four principles: (i) prescription over prediction - the primary output is a ranked curated offer set, not an abstract risk score; (ii) real-time over batch - every component that can operate at call time does; (iii) causal measurement over simple outcome observation - every reported metric has a counterfactual baseline; and (iv) continuous learning - every interaction enriches future model training through a formal feedback loop. The closed-loop learning cycle that distinguishes RetentionOS from open-loop retention systems is detailed in Section 8 and illustrated in Figure 3; open-loop systems terminate at the intervention phase without systematic outcome feedback.

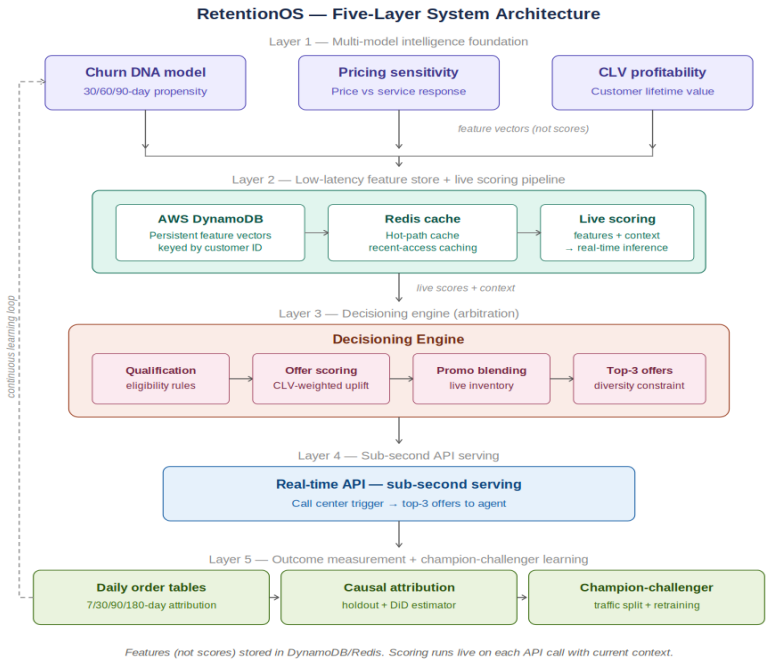

Fig 1: Retentionos Five-Layer System Architecture with Dynamodb/Redis Feature Store and Live Scoring Pipeline.

4. Multi-Model Intelligence Foundation

The decisioning engine draws from three specialized ML models, each contributing a distinct dimension of customer intelligence. All models are trained on curated historical datasets and refreshed on appropriate cadences: behavioral models refresh daily, service quality signals refresh in near-real-time, and tenure features refresh weekly.

The Churn DNA model ingests behavioral event sequences - service contacts, payment patterns, usage trajectories, equipment interactions, digital engagement - and computes churn probability at 30-, 60-, and 90-day horizons. A multi-model ensemble maintains separate sub-models for residential versus commercial segments and tenure-based cohorts, each optimized for its segment's churn dynamics. Competitive market signals (local competitive offer intensity, regional promotional activity) are incorporated as additional features.

The Pricing Sensitivity model estimates each customer's true marginal response to price-based interventions. This distinction matters because customers citing price as a cancellation reason are not uniformly price-motivated; many use price as a proxy for dissatisfaction with service quality or competitive attraction. The Customer Lifetime Value (CLV) Profitability model scores current and projected CLV, stratifying customers into tiers that set the economic ceiling on retention investment - ensuring the system maximizes CLV-weighted retention value rather than merely retention rate. Supplementary attribute models provide product propensity scores, service quality flags (recent disruptions, unresolved tickets), and payment risk estimates.

5. Low-Latency Feature Store and Live Scoring Pipeline

A common misconception is that pre-computing final model scores is the only path to sub-second API latency. RetentionOS takes a fundamentally different approach: the Enterprise Feature Store holds pre-engineered feature vectors - not scores - keyed by customer identifier in low-latency cloud stores. AWS DynamoDB provides the persistent feature vector layer, and Redis provides a hot-path cache for recently accessed customers. When an API call arrives, the scoring pipeline retrieves the pre-computed feature vector within the feature-retrieval latency budget (Table 3), combines it with live call-context inputs (call reason code, agent identifier, current promotional inventory, concurrent account events), and executes model inference in real time.

This live scoring approach produces more current and contextually accurate predictions than serving stale pre-scored batches, because the model can incorporate signals from events that occurred since the last batch run - a new service ticket, a recent payment, a competitor offer received that morning. The feature store architecture maintains training-serving consistency by enforcing that feature computation logic at training time is identical to that at serving time, eliminating the training-serving skew that silently degrades production model performance. Feature vectors are versioned and lineage-tracked, enabling retrospective audit of what inputs drove each decisioning event - a prerequisite for the causal attribution methodology described in Section 8.

6. Decisioning Engine Arbitration

The decisioning engine transforms the intelligence package into the top-3 curated offer prescriptions through a formalized four-step arbitration. Let c denote a customer with feature vector f(c), O_c the eligible offer set, $\tau(c, o)$ the estimated individual treatment effect of offer o on customer c (uplift), $w\_CLV(c)$ the CLV weight, and $e(c, o) \in \{0, 1\}$ the binary eligibility function encoding business rules. The decision objective selects the top-k offers maximizing expected CLV-weighted uplift:

$$O^*(c) = \text{top-}k \{ \tau(c, o) \cdot w\_CLV(c) : o \in O_c, e(c, o) = 1 \} \quad (1)$$

subject to a diversity constraint ensuring the selected offers span distinct value-proposition categories (price-based, service-enhancement, loyalty recognition). The CLV weight is defined as:

$$w\_CLV(c) = CLV(c) / CLV\_median \quad (2)$$

normalizing retention value contributions relative to the median customer CLV. Algorithm 1 formalizes the arbitration procedure.

Algorithm 1: Decisioning Engine Arbitration

Input: *customer_id c, call_context κ, promo_inventory P*
Output: *ranked top-3 offer prescriptions O*(c)*

```
f(c) ← FeatureStore.retrieve(c)            // DynamoDB + Redis
scores ← ScoringPipeline.run(f(c), κ)      // live inference
O_c ← { o : e(c, o) = 1 }                  // qualification
for each o in O_c do
    score(o) ← τ(c, o) · w_CLV(c) · promo_weight(o, P)
O_ranked ← sort(O_c by score descending)
O*(c) ← apply_diversity_constraint(O_ranked, k=3)
return O*(c)
```

Step 1 (Customer Qualification) applies eligibility rules configurable by business stakeholders without engineering changes - minimum tenure thresholds for loyalty offers, outstanding balance checks, recency constraints preventing duplicate interventions, and commercial-versus-residential routing. Step 2 (Offer Scoring) computes expected CLV-weighted uplift for each qualified offer. Step 3 (Promotional Blending) adjusts scores by promo_weight(o, P), which boosts offers with remaining budget and expiring time windows. Step 4 (Top-3 Selection) applies the diversity constraint, ensuring the agent receives genuinely distinct options rather than three variations of the same intervention.

7. Sub-Second Api Serving

Figure 2 illustrates the complete API flow from customer cancellation intent to agent offer receipt. The hard latency requirement is under one second; a live call center interaction cannot tolerate visible processing delays without

degrading customer experience and intervention effectiveness. Three architectural decisions combine to meet this requirement: (i) feature retrieval from DynamoDB/Redis rather than on-the-fly computation; (ii) parallel feature dimension extraction executing all lookups simultaneously; and (iii) stateless decisioning engine deployment enabling horizontal scaling without shared-state bottlenecks.

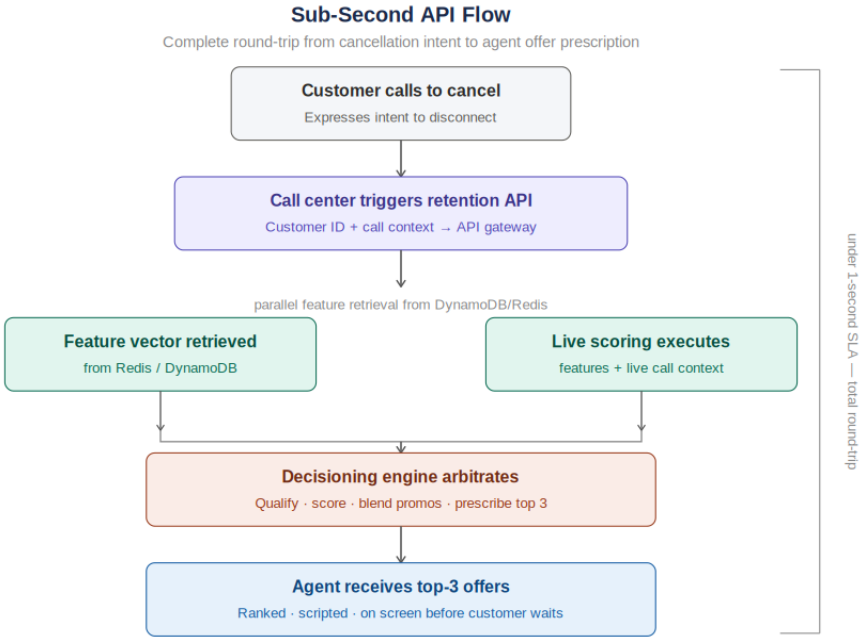

Fig 2: Retentionos API Flow - Cancellation Intent to Agent Offer in Under One Second.

To meet the one-second end-to-end SLA, the system's latency budget is allocated across the pipeline stages as shown in Table 3. Each stage has an engineering target that allows comfortable headroom under the envelope. Auto-scaling cloud infrastructure handles call-volume variability, provisioning additional capacity within seconds of traffic spikes. Circuit breaker patterns provide graceful degradation - if any component exceeds its latency budget, the system falls back to a pre-ranked default offer set, ensuring the agent experience is never blocked.

Table 3: Latency Budget Allocation across Pipeline Stages to Meet the One-Second End-to-End SLA.

Pipeline Stage	Latency Budget	Design Notes
Feature retrieval (Redis hot-path)	≤ 20 ms	Recent-access caching
Feature retrieval (DynamoDB)	≤ 30 ms	Persistent vector store
Live scoring pipeline	≤ 200 ms	Model inference + context fusion
Decisioning engine arbitration	≤ 100 ms	Qualify · score · blend · select
Network and serialization overhead	≤ 150 ms	API gateway + payload transport
Reserved headroom	≤ 500 ms	Burst capacity + circuit fallback
End-to-end SLA	≤ 1000 ms	Hard operational ceiling

8. Closing the Loop: Outcome Measurement and Continuous Learning

The closed-loop learning cycle distinguishes RetentionOS from open-loop retention systems. Figure 3 visualizes the five-phase cycle: Predict → Intervene → Measure → Attribute → Retrain. Each complete cycle enriches the training dataset with causally grounded records, producing models that progressively improve at identifying not just who will churn, but which interventions will causally produce retention for which customer profiles.

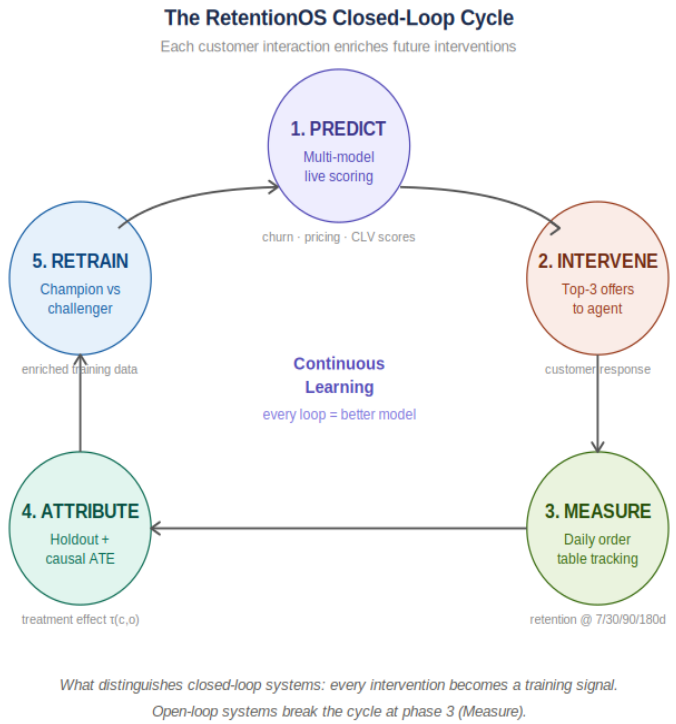

Fig 3: The Retentionos Closed-Loop Learning Cycle. Open-Loop Systems Break at Phase 3 (Measure).

8.1. A Worked Example

To illustrate the flow concretely, consider a hypothetical customer C-7829: tenure 34 months, CLV tier High (w_CLV = 1.6), churn score 0.81 at 30-day horizon, pricing sensitivity 0.32 (low - not primarily price-driven). The decisioning engine qualifies C-7829 for three offer categories. After scoring: Offer A (loyalty equipment upgrade, τ = 0.19, weighted score 0.304), Offer B (bundle discount, τ = 0.14, weighted score 0.224), Offer C (bottom-of-bill discount, τ = 0.09, weighted score 0.144). The diversity constraint is satisfied. The API returns the ranked triple to the agent well within the one-second SLA. The agent presents Offer A. C-7829 accepts. (All values in this example are illustrative and chosen to demonstrate the algorithm mechanics.)

At the 30-day attribution checkpoint, the daily order processing table confirms: account retained, equipment upgrade applied, no downgrade. The interaction is logged as a positive treatment record. At the 90-day checkpoint, retention is sustained. The enriched record - feature vector at decision time, offer presented, offer accepted, 90-day retention outcome, estimated counterfactual from holdout analysis - is appended to the retraining dataset, adding causal signal that loyalty recognition outperforms price reduction for high-CLV low-price-sensitivity customers in the 30-40 month tenure band. This signal strengthens τ estimates for comparable future customers.

8.2. Daily Order Processing Table Monitoring

Outcome measurement relies on operational transaction records - the daily order processing tables that capture every account state change. Four attribution windows are evaluated per interaction: 7 days (immediate retention), 30 days (short-term stability), 90 days (primary outcome metric), and 180 days (durability). For each window the system records retention status, offer application confirmation, service downgrade flag (partial churn), and realized revenue. This approach provides complete, auditable ground truth without the coverage gaps and response biases of survey-based measurement.

8.3. Causal Attribution

Raw retention rates are not valid learning signals without causal correction. The primary CLV-weighted retention metric is defined as:

$$R\_CLV = (1/N) \Sigma \; CLV(c_i) \cdot \mathbb{1}[retained_{i,90d}] \quad (3)$$

A statistically designed holdout group withholds AI-guided offer recommendations from a random sample of at-risk inbound callers, providing a direct counterfactual baseline. Agents interact with holdout customers using standard protocols. The average treatment effect is estimated via difference-in-differences (DiD):

$$ATE = (R\_CLV,treated,post - R\_CLV,treated,pre) - (R\_CLV,control,post - R\_CLV,control,pre) \quad (4)$$

This formulation controls for concurrent market dynamics affecting treated and control populations equally. Propensity score matching is additionally used for offer-type and segment-level analyses where random assignment is infeasible, constructing synthetic control groups matched on churn score, CLV tier, and call context.

8.4. Champion-Challenger Model Governance

To ensure continuous model evolution and guard against performance degradation, RetentionOS specifies a traffic

split mechanism for champion-challenger evaluation. Inbound retention API calls are routed according to a configurable split ρ (for example, 90% champion / 10% challenger). Both models serve real customers through the full decisioning pipeline. Outcome records are tagged by model version, enabling direct head-to-head comparison on the primary CLV-weighted retention metric. Promotion of challenger to champion follows a formal statistical criterion:

$$Promote\ M_{challenger} \rightarrow M_{champion}\ iff\ n \geq n_{min} \wedge t\text{-}statistic > t_{\{\alpha/2\}} \wedge \Delta > \delta_{min} \quad (5)$$

where n is the challenger sample size, n_min is a minimum-exposure threshold, α = 0.05 is the significance level, and δ_min is the minimum practically meaningful improvement in R_CLV. Algorithms 2 and 3 formalize the routing and promotion procedures.

Algorithm 2: Champion-Challenger Traffic Split Routing

Input: *customer_id c, split_config σ*
Output: *offer prescription O*

```
r ← random_uniform(0, 1)
if r < σ.challenger_pct then
    model_version ← CHALLENGER
else
    model_version ← CHAMPION
O ← score_and_arbitrate(c, model_version)
log_interaction(c, O, model_version)
return O
```

Algorithm 3: Nightly Promotion Evaluation

Input: *observation_window_days w, min_samples n_min, min_effect δ_min*
Output: *promotion decision*

```
champ_outcomes ← get_outcomes(CHAMPION, window=w)
chall_outcomes ← get_outcomes(CHALLENGER, window=w)
if len(chall_outcomes) < n_min then
    return NO_PROMOTION                    // insufficient data
μ_champ ← mean(champ_outcomes.R_CLV)
μ_chall ← mean(chall_outcomes.R_CLV)
Δ ← μ_chall − μ_champ
p ← welch_ttest(chall_outcomes, champ_outcomes)
if p < 0.05 and Δ > δ_min then
    promote(CHALLENGER → CHAMPION)
    return PROMOTED
return NO_PROMOTION
```

8.5. Training Data Enrichment

Each interaction enriches the training dataset with: the feature vector at decision time, the treatment applied (as determined by Algorithm 2's routing decision), the ordered outcome sequence across all attribution windows, and the estimated counterfactual from holdout and propensity-matched controls. This causally grounded dataset drives scheduled retraining cycles supplemented by drift detection - when churn score distributions, treatment assignment rates, or outcome rates deviate beyond defined thresholds, off-cycle retraining is triggered, and the next invocation of Algorithm 3 evaluates whether the retrained challenger should be promoted. Each successive model generation learns not only which features correlate with churn but which treatments causally produce retention for which customer profiles - a compounding improvement fundamentally unavailable to open-loop systems.

9. Proposed Evaluation Framework

This section presents an evaluation framework that practitioners can apply when implementing RetentionOS or comparable closed-loop retention architectures. The framework consists of two components: a capability comparison against representative baseline approaches, and an ablation methodology with expected degradation signatures. Both are proposed as reproducible reference tools rather than reported outcomes - enabling each adopting organization to benchmark its own implementation with its own production data.

9.1. Baseline Comparison Framework

Four baseline approaches are proposed as points of comparison, ordered by increasing architectural sophistication. The Human-Only baseline reflects agent-driven retention without any AI assistance. The Rule-Based baseline uses hand-specified eligibility and offer-assignment rules without predictive modeling. The Open-Loop Churn Prediction baseline adds a batch-scored churn model feeding a campaign management system but includes no uplift, no outcome feedback, and no CLV-weighted optimization. The Pure Uplift Single-Model baseline adds individual-level treatment effect estimation but still lacks multi-model intelligence, live scoring, CLV weighting, and champion-challenger governance. Table 4 summarizes the qualitative capability profile of each approach across seven architectural dimensions.

Table 4: Capability Comparison of Baseline Approaches versus Retentionos across Seven Architectural Dimensions.

Capability	Human-Only	Rule-Based	Open-Loop	Uplift	RetentionOS
Predictive scoring	—	—	✓	✓	✓
Individual treatment effects	—	—	—	✓	✓
Real-time live scoring	—	—	—	partial	✓
Multi-model intelligence	—	—	—	—	✓

CLV-weighted optimization	—	partial	—	—	✓
Causal outcome attribution	—	—	—	partial	✓
Closed-loop retraining	—	—	—	—	✓

Each progression from a weaker to a stronger baseline contributes a distinct capability that the subsequent approach adds. RetentionOS is the first framework we are aware of to unify all seven capabilities in a single closed-loop system. For organizations adopting RetentionOS, the recommended evaluation protocol is: measure the Human-Only baseline via a statistically designed holdout group as described in Section 8.3; compare Rule-Based, Open-Loop, and Uplift baselines via historical analysis of prior retention systems in the organization, or via controlled side-by-side arm comparison where feasible.

9.2. Ablation Methodology

To quantify the contribution of each architectural component, we propose an ablation methodology that disables one component at a time and measures the resulting change in the primary CLV-weighted retention metric R_CLV against an unablated reference window. Table 5 enumerates the ablation cases along with their expected degradation signatures derived from the functional role each component plays in the framework. Practitioners can use this as a diagnostic reference: during production monitoring, when aggregate metrics shift, the shape of the shift across dimensions often identifies which component is underperforming.

Table 5: Ablation Cases and Their Expected Degradation Signatures.

Ablation Case	Component Removed	Expected Degradation Pattern
A1	Champion-challenger governance	Gradual performance drift over time
A2	CLV weighting	Retention rate preserved; value drops sharply
A3	Uplift modeling	Major retention drop; false-positive targeting
A4	Promotional blending	Offer availability mismatches; acceptance drop
A5	Pricing sensitivity model	Systematic over-discounting
A6	Diversity constraint	Offer fatigue; reduced acceptance

Each component's removal produces a characteristic failure mode that distinguishes it from other components: CLV weighting loss produces a revenue collapse without a retention rate drop; uplift modeling loss produces the opposite; promotional blending loss manifests as inventory mismatches; and so on. This diagnostic structure is itself a contribution of the framework - enabling practitioners to reason about not just whether the system is underperforming but which specific component is responsible.

10. Discussion, Limitations, and Future Work

Three design considerations deserve emphasis for practitioners implementing this framework. First, the configurable qualification layer is more consequential than it may initially appear; without separating business eligibility rules from ML scoring, the system can produce technically optimal offers that violate contractual or policy constraints and harm operational uptake. The qualification layer should be owned and governed by business stakeholders, not by the ML engineering team. Second, the holdout group should be designed as AI-augmented versus standard agent interaction (not as treated versus untreated), ensuring no customer is denied retention assistance and resolving ethical concerns about experimentation on customers in distress. Third, circuit-breaker fallbacks to pre-ranked default offer sets are essential during infrastructure events - the cost of an occasional slightly suboptimal recommendation is vastly lower than the cost of a blocked agent interface.

Three limitations are acknowledged. Cold-start performance for newly acquired customers remains weaker than for established customers, as behavioral sequence features are undefined or sparse; a dedicated cold-start sub-model or behavioral proxy from early interactions would improve this. The attribution window choice is inherently a tradeoff between statistical power (longer windows) and responsiveness of the learning loop (shorter windows); the selected 90-day primary window reflects a practical compromise but is not optimal for all offer types. As the offer catalog grows, the combinatorial complexity of the diversity constraint increases; a learned offer taxonomy rather than a hand-specified category structure is an open direction.

Future work includes four promising extensions: (i) online learning integration that updates uplift estimates continuously rather than in batched retraining cycles; (ii) multi-agent reinforcement learning for sequential retention decisions across multiple customer contacts; (iii) cross-product transfer learning to accelerate model bootstrapping in adjacent business lines; and (iv) interpretability layers that surface the specific features driving each offer recommendation to support agent judgment and regulatory compliance.

11. Ethical Considerations

Closed-loop prescriptive systems like RetentionOS can influence decisions affecting large customer populations, warranting explicit ethical treatment. Periodic bias audits should compare offer distributions across customer segments defined by geography and tenure as proxies for demographic characteristics; statistically significant skew should trigger investigation. The CLV profitability tier mechanism - which

constrains offer generosity for lower-value customers - is a deliberate economic policy rather than an incidental outcome, and should be documented and visible to business governance stakeholders. Agent interfaces should explicitly flag recommendations as AI-generated, preserving agent override authority and professional judgment. The holdout group design should ensure all customers receive human retention assistance; only the AI-optimization of that assistance varies across experimental arms.

12. Reproducibility

The framework can be implemented on a commodity cloud stack. A reference implementation stack is: Python 3.14 for model training and scoring pipelines; Apache Spark 4.x for distributed feature engineering; AWS DynamoDB for persistent feature vector storage; Redis for hot-path caching; containerized stateless decisioning engine services on Kubernetes; API gateway with auto-scaling. Churn propensity, pricing sensitivity, and CLV models are recommended as gradient boosting ensembles (XGBoost or LightGBM) trained on historical transaction and interaction data, though any calibrated probabilistic classifier is suitable. Uplift modeling can use the X-learner meta-learner framework [5]. The architectural patterns described in this paper are stack-agnostic; equivalent implementations using Google Cloud Bigtable/Memorystore or Azure Cosmos DB/Cache are expected to achieve comparable latency characteristics.

13. Conclusion

This paper presented RetentionOS, a closed-loop prescriptive ML framework for real-time customer retention. The core contributions are fourfold: (1) a formal mathematical framework integrating multi-model intelligence with CLV-weighted decisioning and a statistically governed champion-challenger promotion criterion; (2) a low-latency feature store design that separates feature engineering from scoring, enabling sub-second live inference without sacrificing prediction currency; (3) a daily order processing table methodology for high-fidelity causal outcome measurement without survey dependence; and (4) a reproducible evaluation framework including capability comparison against four baseline approaches and an ablation methodology with characteristic degradation signatures. The framework is directly applicable to any high-CLV customer lifecycle requiring real-time intervention and causal learning - telecommunications retention, financial services attrition, subscription cancellation flows, and healthcare patient engagement are immediate target domains. RetentionOS is offered as a reference framework that practitioners can adopt, adapt, and extend in their own production environments.

References

[1] Huang, M. T. Kechadi, and B. Buckley, "Customer churn prediction in telecommunications," *Expert Syst. Appl.,* vol. 39, no. 1, pp. 1414-1425, Jan. 2012.

[2] A. De Caigny, K. Coussement, and K. W. De Bock, "A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees," *Eur. J. Oper. Res.,* vol. 269, no. 2, pp. 760-772, Sep. 2018.

[3] H. Zhao, X. Yao, and R. P. W. Duin, "Predicting customer churn in telecommunications: A review," *Int. J. Inf. Manage.,* vol. 55, Dec. 2020, Art. no. 102185.

[4] N. J. Radcliffe and P. D. Surry, "Real-world uplift modelling with significance-based uplift trees," Stochastic Solutions, Edinburgh, UK, White Paper TR-2011-1, 2011.

[5] S. R. Künzel, J. S. Sekhon, P. J. Bickel, and B. Yu, "Metalearners for estimating heterogeneous treatment effects using machine learning," *Proc. Nat. Acad. Sci.,* vol. 116, no. 10, pp. 4156-4165, Mar. 2019.

[6] P. Gutierrez and J.-Y. Gerardy, "Causal inference and uplift modelling: A review of the literature," in *Proc. 3rd Int. Conf. Predictive Appl. APIs,* vol. 67, pp. 1-13, 2017.

[7] P. R. Rosenbaum and D. B. Rubin, "The central role of the propensity score in observational studies for causal effects," *Biometrika,* vol. 70, no. 1, pp. 41-55, Apr. 1983.

[8] S. Athey and G. Imbens, "Recursive partitioning for heterogeneous causal effects," *Proc. Nat. Acad. Sci.,* vol. 113, no. 27, pp. 7353-7360, Jul. 2016.

[9] J. Hermann and M. Del Balso, "Meet Michelangelo: Uber's machine learning platform," Uber Engineering Blog, Sep. 2017. Accessed: Apr. 17, 2026. [Online]. Available: https://eng.uber.com/michelangelo-machine-learning-platform/

[10] M. Olson, M. Armstrong, A. Gundlach, and J. Stoica, "The feature store: Architectural patterns for machine learning," in *Proc. IEEE Int. Conf. Big Data,* pp. 3420-3427, Dec. 2020.

[11] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data management challenges in production machine learning," in *Proc. ACM SIGMOD Int. Conf. Manage. Data,* pp. 1723-1726, May 2017.

[12] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann, "Online controlled experiments at large scale," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining,* pp. 1168-1176, Aug. 2013.

[13] O. Chapelle and L. Li, "An empirical evaluation of Thompson sampling," in *Advances in Neural Information Processing Systems 24,* pp. 2249-2257, Dec. 2011.

[14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.,* vol. 46, no. 4, pp. 1-37, Apr. 2014.

[15] M. Treveil, N. Omont, C. Stenac, K. Lefevre, D. Phan, J. Zentici, A. Lavoillotte, M. Miyazaki, and L. Heidmann, *Introducing MLOps: How to Scale Machine Learning in the Enterprise.* Sebastopol, CA, USA: O'Reilly Media, 2020.