© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

Agentic Artificial Intelligence for Personalized Customer Experience Optimization in Retail: Algorithms, Architectures, and Production Implementation

VENKATESH GUNDU
Senior Manager, Data Services & AI Platforms, Victoria's Secret

Abstract- ***This paper provides a technically grounded examination of Agentic AI architectures deployed for retail customer experience optimization. Unlike prior work that addresses this topic at a strategic level, this paper identifies the specific machine learning algorithms, model architectures, feature engineering pipelines, and system orchestration patterns that underpin production deployments. Core topics include contextual bandit algorithms — LinUCB, Thompson Sampling, and neural bandits — for real-time offer selection; two-tower neural retrieval networks with approximate nearest-neighbor search for large-scale product recommendation; gradient-boosted demand elasticity models and reinforcement-learning pricing agents for dynamic pricing; survival analysis and deep temporal models for customer churn prediction; and LLM-based multi-agent orchestration patterns for autonomous retail workflows. The paper further addresses real-time feature store architecture, bias detection and mitigation, and guardian-agent governance for safe autonomous action execution. Two detailed production scenarios — influencer-triggered dynamic pricing and subscription churn intervention — illustrate end-to-end system integration. The intended audience is data scientists, machine learning engineers, and technical architects building or evaluating agentic retail systems.***

Keywords: Agentic AI, Contextual Bandits, Two-Tower Networks, Survival Analysis, Dynamic Pricing, Reinforcement Learning, Feature Stores, Multi-Agent Orchestration, Retail Personalization, Algorithmic Fairness.

I. INTRODUCTION

The retail industry has undergone a fundamental transition from product-centric to experience-centric competition. Consumers now interact with retailers across websites, mobile applications, social media, in-store sensors, and voice interfaces simultaneously, generating unprecedented volumes of behavioral data. Organizations that convert this data into real-time, individualized responses at scale hold a structural competitive advantage.

Traditional machine learning systems deployed in retail — collaborative filtering engines, batch pricing models, rule-based promotions — were designed for a different operating environment. They consume historical data, produce a ranked list or decision, and wait for the next scheduled batch run. They cannot observe a sudden shift in consumer sentiment caused by a social media post and respond with a pricing adjustment and targeted campaign within minutes. They cannot detect, from a customer's browsing session, that she is three days from cancelling her subscription and route a personalized retention offer through her preferred channel before that happens.

Agentic AI systems close this gap. The term "agentic" describes systems capable of four behaviors that traditional ML pipelines lack: (1) continuous perception of real-time signals, not just historical logs; (2) multi-step planning toward a long-horizon objective rather than single-step prediction; (3) autonomous action execution — triggering downstream systems without per-action human approval; and (4) closed-loop learning, where outcomes of actions update model parameters or beliefs. Together these capabilities enable a qualitatively different mode of customer engagement.

This paper is addressed to the engineering and data science practitioners responsible for building or evaluating such systems. Sections 2 through 5 detail the core ML algorithms for recommendation, offer optimization, pricing, and churn prediction. Section 6 addresses system architecture including real-time feature stores and multi-agent orchestration. Section 7 covers bias detection and mitigation. Section 8 presents two complete production scenarios. Section 9

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

addresses governance and Section 10 concludes with an implementation roadmap.

II. FROM TRADITIONAL AI TO AGENTIC SYSTEMS: THE ARCHITECTURAL SHIFT

2.1 Limitations of Conventional Retail ML

Classical retail recommendation systems operate in one of two modes. Collaborative filtering learns latent user and item embeddings from a historical purchase-interaction matrix. Matrix factorization methods such as Alternating Least Squares (ALS) and SVD++ decompose the user-item interaction matrix $R \approx U \times V^T$, where U and V are low-rank embedding matrices trained to minimize reconstruction error on observed interactions. These perform well for popular items but fail for new products and users — the cold-start problem. Content-based filtering represents items using feature vectors and matches them to user preference profiles. It handles new items but cannot discover cross-category preferences.

Both approaches share a structural limitation: they are static. A model trained on last month's transaction log makes recommendations based on that snapshot. They cannot respond to inventory depletion mid-session, incorporate a viral social mention that emerged two hours ago, or adjust strategy when a customer transitions from exploration to purchase mode within a single session.

2.2 The Four Agentic Capabilities

An agentic system replaces the batch predict-and-wait cycle with a continuous perceive-plan-act-learn loop. Table 1 contrasts the two paradigms along operationally meaningful dimensions.

Table 1. Traditional ML vs. Agentic AI in Retail Contexts

Dimension	Traditional ML System	Agentic AI System
Data currency	Trained on historical batches; features computed hours or days prior	Continuous ingestion of real-time streams; features updated within seconds
Decision horizon	Single-step prediction: next best item or offer	Multi-step planning: sequences of actions optimized for long-horizon LTV
Action authority	Produces recommendations; humans approve and execute	Autonomously triggers pricing APIs, promotion systems, communication channels
Adaptation speed	Model updated in scheduled retraining runs (daily to weekly)	Online learning and bandit feedback update policies continuously
Scope	Single-task: recommendation OR pricing OR churn	Multi-task: orchestrates across recommendation, pricing, support, and fulfillment simultaneously
Failure mode	Stale recommendations; misses emerging trends	Autonomous error propagation across interconnected systems if ungoverned

III. REAL-TIME OFFER SELECTION: CONTEXTUAL BANDIT ALGORITHMS

3.1 Problem Formulation

The problem of choosing which promotion, recommendation slot, or content variant to show a specific user in a specific context is an exploration-exploitation problem. At decision time t, the system observes context x_t — a vector encoding user features, session features, item features, time of day, device type, and real-time signals. It selects action a_t from action space A and observes scalar reward r_t (click, add-to-cart, purchase, or revenue). The goal is to learn a policy $\pi(a|x)$ that maximizes expected cumulative reward. Three algorithms dominate production retail deployments, compared in Table 2.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

Table 2. Contextual Bandit Algorithms: Mechanisms, Use Cases, and Trade-offs

Algorithm	Core Mechanism	Retail Use Cases	Key Trade-off
LinUCB	Models $E[r\|x,a] = x^T\theta_a$ via ridge regression per arm. Selects arm with highest score + uncertainty bonus: $x^T\theta_a + \alpha\sqrt{(x^TA_a^{-1}x)}$. α controls exploration vs. exploitation.	Homepage banner selection, email subject line testing, search result ranking	Fast (<5ms inference), interpretable. Breaks for non-linear reward structures.
Thompson Sampling	Maintains posterior $P(\theta_a\|data)$ per arm. At each step samples θ_a from posterior and picks argmax_a $E[r\|x,\theta_a]$. Posterior updated analytically after each observation.	Real-time product recommendation, push notification timing, dynamic offer sequencing	Stronger empirical performance than UCB; naturally handles non-stationary demand. Harder to audit.
Neural Contextual Bandit	Deep neural network models reward non-linearly. Exploration via NeuralUCB (last-layer uncertainty) or ensemble disagreement across bootstrap-trained heads.	Cross-category recommendations, style matching in fashion, multi-objective offer optimization	Captures complex feature interactions. Requires GPU inference (50–200ms); more complex to maintain.

3.2 LinUCB: Upper Confidence Bound Formulation

LinUCB is the standard starting point for retail bandit deployments due to its deterministic inference and interpretability. Figure 1 presents the algorithm structure formally.

Figure 1. LinUCB Algorithm — Formal Structure

LinUCB (Linear Upper Confidence Bound)	
Reward model	$E[r \mid x, a] = x^T \theta_a$
Arm selection	$a_t = \text{argmax\_a}\ [\ x^T\theta_a + \alpha\ \sqrt{(x^T A_a^{-1} x)}\ ]$
Update rule	$A_a \leftarrow A_a + x_t x_t^T \quad b_a \leftarrow b_a + r_t x_t \quad \theta_a \leftarrow A_a^{-1} b_a$
Where	A_a = regularized feature covariance (initialized: I)
	b_a = reward-weighted feature vector (initialized: 0)
	α = exploration parameter (tuned on validation set)
Inference	< 5ms. Deterministic. Interpretable coefficients θ_a.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

3.3 Thompson Sampling Update Rule

Thompson Sampling consistently outperforms UCB methods empirically and naturally handles non-stationary demand shifts — an important property in retail, where seasonal and trend-driven demand changes are frequent. Figure 2 presents the update procedure for Bernoulli rewards (click/no-click), the most common retail reward signal.

Figure 2. Thompson Sampling — Bernoulli Reward Case

Thompson Sampling (Bernoulli Rewards)	
Prior	$\theta_a \sim \text{Beta}(\alpha_a, \beta_a)$ initialized: $\alpha_a = \beta_a = 1$ (uniform)
At each step	Sample $\tilde{\theta}_a \sim \text{Beta}(\alpha_a, \beta_a)$ for each arm a
Arm selection	$a_t = \text{argmax\_a } \tilde{\theta}_a$
Posterior update	If reward $r_t = 1$: $\alpha_a \leftarrow \alpha_a + 1$
	If reward $r_t = 0$: $\beta_a \leftarrow \beta_a + 1$
Property	Posterior concentrates around true θ^* as data accumulates.
Inference	~1ms. Naturally balances exploration as uncertainty decreases.

IV. LARGE-SCALE PRODUCT RECOMMENDATION: TWO-TOWER NEURAL NETWORKS

4.1 Architecture Overview

For catalogs with millions of items and tens of millions of users, real-time scoring of all item-user pairs is computationally infeasible. Modern retail recommendation systems address this through a two-stage pipeline: retrieval (identify a candidate set of hundreds of relevant items from the full catalog) followed by re-ranking (score candidates using a richer model incorporating real-time context). Two-tower neural networks are the dominant retrieval architecture. Figure 3 illustrates the complete architecture.

Figure 3. Two-Tower Neural Retrieval Architecture

Two-Tower Neural Network — Retrieval Architecture
USER TOWER — Input Features
User ID embedding | Age bucket | Gender | Device type | Purchase history embedding (mean of last-N item embeddings) | Brand affinity vector | Price sensitivity quintile | Current session category embedding | Recency score
▼
USER TOWER — Encoder (3-layer MLP, ReLU, batch normalization)
Layer 1: 512 units → Layer 2: 256 units → Layer 3: 128 units Output: user_embedding ∈ $\mathbb{R}^{128}$ (L2-normalized)
▼
ITEM TOWER — Input Features
Item ID embedding | Category embedding | Brand embedding | log(price) | Average rating | Review count | Inventory level | Description embedding (BERT / sentence-transformer) | Image embedding (ViT or ResNet-50)
▼
ITEM TOWER — Encoder (3-layer MLP, ReLU, batch normalization)
Layer 1: 512 units → Layer 2: 256 units → Layer 3: 128 units Output: item_embedding ∈ $\mathbb{R}^{128}$ (L2-normalized, pre-computed and indexed offline)
▼
RETRIEVAL — Approximate Nearest Neighbor Search
Relevance score = dot_product(user_embedding, item_embedding) Top-K candidates retrieved via

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

ANN index (FAISS, ScaNN, or Pinecone) Latency: < 20ms across catalogs of 10M+ items
▼
RE-RANKING — Real-Time Context Layer
Inputs: Top-K candidates + current inventory level + promotion status + session clickstream (last 5 items) + competitor price signals + inventory objective weight w(i,t) Model: LightGBM (< 10ms) or cross-attention transformer (50–100ms) Output: Final ranked list served to customer

4.2 Sampling Bias Correction

A critical training issue is sampling-bias correction. In a standard setup, the model trains on observed positive interactions with randomly sampled negatives. Because popular items dominate training data, the model learns to over-rank popular items — popularity bias. The IPS-corrected training loss reweights each negative sample by the inverse of its frequency in the training batch:

Figure 4. Sampling-Bias-Corrected Training Objective

Sampling-Bias Correction in Two-Tower Training	
Standard loss	$L(\theta) = (1/N)\ \Sigma_i\ \text{loss}(\ f\_\theta(x_i, a_i), r_i\ )$
IPS-corrected loss	$L\_IPS(\theta) = (1/N)\ \Sigma_i\ (\ r_i / p(a_i)\ ) \times \text{loss}(\ f\_\theta(x_i, a_i), r_i\ )$
Where	$p(a_i)$ = estimated probability item a_i was shown under the logging policy
Effect	Rare items shown infrequently but receiving positive responses receive high weight during training, correcting the exposure bias of the prior recommendation policy.

5. DYNAMIC PRICING AND CHURN PREDICTION

5.1 Dynamic Pricing Architecture

Dynamic pricing optimization requires estimating how demand responds to price changes — the price elasticity of demand — and optimizing prices subject to business constraints. Production retail pricing systems employ a two-layer architecture as illustrated in Figure 5.

Figure 5. Dynamic Pricing System Architecture

Dynamic Pricing — Two-Layer Architecture
LAYER 1: Demand Forecasting (LightGBM)
Estimates the price-demand function Q(P, X) where P is price and X includes: seasonality indicators | promotional calendar flags | competitor prices (from price intelligence APIs) | inventory age | historical sales velocity | category demand trends LightGBM is preferred over deep models for this layer: native categorical feature handling, missing data tolerance, and interpretable coefficients satisfy regulatory and commercial team requirements for explainable pricing decisions.
▼
LAYER 2: Pricing Optimization Agent
Constrained optimization problem solved given the demand curve from Layer 1: Maximize: Revenue(P) = P × Q(P, X) Subject to: P ≥ cost_floor (margin constraint) P ≤ suggested_retail_price (brand constraint) Q(P, X) ≤ current_inventory (stock constraint) |P − competitor_price| ≤ δ (competitive constraint) For single-item, single-period pricing: solved analytically from the demand curve. For multi-item pricing with cross-elasticity and multi-period inventory targets: Proximal Policy Optimization (PPO) or Soft Actor-Critic (SAC) RL agents trained in simulated environments before deployment.
▼
GUARDIAN REVIEW — Before Execution

All price changes reviewed by guardian agent against: margin floor check | MSRP ceiling check | anomaly detection (N-σ from historical distribution) | fairness constraint (no discriminatory pricing outcomes) Auto-approved if all checks pass. Queued for human review otherwise.

5.2 Churn Prediction: Survival Analysis vs. Binary Classification

Predicting customer churn is conventionally framed as binary classification: will this customer churn within 90 days, yes or no? This framing discards the most actionable information — when the customer is likely to churn. A model that predicts churn probability at a single fixed horizon cannot distinguish a customer who will churn in 3 days (urgent intervention required) from one who will churn in 85 days (intervention can be scheduled). Survival analysis addresses this directly.

Figure 6. Survival Analysis Models for Customer Churn

Survival Analysis — Churn Risk Modeling	
Hazard function	$h(t \mid x)$ = instantaneous rate of churn at time t given survival until t
Cox PH model	$h(t \mid x) = h_0(t) \times \exp(x^T\beta)$
Where	$h_0(t)$ = non-parametric baseline hazard (estimated from data)
	β = learned coefficients on customer feature vector x
Survival curve	$S(t \mid x) = \exp(-\int_0^t h(u\|x)\, du)$
Churn probability	$P(\text{churn before day } t) = 1 - S(t \mid x)$ for any horizon t
DeepSurv	Replaces linear $x^T\beta$ with neural network $f_\theta(x)$, capturing non-linear churn drivers
With sequences	GRU or Transformer encoder processes session history → context vector fed into survival model

In a production system, the survival model outputs a time-indexed risk curve for each customer daily. An agentic orchestrator monitors these curves and triggers escalating interventions as risk crosses defined thresholds: a personalized email when P(churn < 30 days) > 0.40, a push notification with offer when P(churn < 14 days) > 0.60, and a high-value retention offer when P(churn < 7 days) > 0.80. Counterfactual uplift — the marginal retention probability attributable to each intervention — is estimated using doubly-robust causal inference estimators trained on historical A/B test data.

VI. AGENTIC SYSTEM ARCHITECTURE

6.1 Real-Time Feature Store

Every component described in preceding sections depends on a feature store — a system that maintains up-to-date feature values for users, items, and context and serves them with low latency at model inference time. The feature store is the infrastructure prerequisite for real-time personalization; without it, even sophisticated ML models consume stale batch-computed features. Figure 7 illustrates the three-layer architecture used in production retail systems.

Figure 7. Real-Time Feature Store — Three-Layer Architecture

Real-Time Feature Store Architecture
ONLINE STORE — Target latency: < 10ms reads
Technology: Redis or DynamoDB Holds: current-session features updated within seconds of each event — last 5 items viewed, current session category distribution, recency/frequency scores, current item price, inventory level, and promotion status updated by operational system events in real time.
▼

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

STREAM PROCESSOR — Event-to-feature latency: 1–5 seconds
Technology: Apache Flink or Kafka Streams Consumes: raw event streams (clickstream, purchase events, inventory updates, social mention signals) Produces: derived real-time features materialized into online store — session engagement signals, real-time search query embedding, sudden demand spike indicators, in-session add-to-cart propensity score.
▼
OFFLINE STORE — High-throughput batch writes
Technology: Parquet on object storage (S3 / GCS) backed by BigQuery, Snowflake, or Spark Holds: long-horizon features computed in daily/weekly batch jobs — 12-month purchase embedding, cohort membership, price sensitivity quantile, lifetime value decile, brand affinity vector, historical churn risk trajectory.

6.2 Multi-Agent Orchestration

The orchestration layer sits above the ML models and decomposes high-level customer experience objectives into specific model queries and downstream system actions. Two architectural patterns are dominant in production. The ReAct pattern (Yao et al., 2023) uses a large language model as a reasoning engine interleaving Thought, Action, and Observation steps. This is appropriate for complex multi-step resolution workflows where context must be gathered incrementally. Figure 8 illustrates a ReAct trace for a shipping delay complaint from a high-value customer.

Figure 8. ReAct Orchestration — Shipping Delay Resolution Trace

ReAct Agent Trace: High-Value Customer Shipping Delay	
1	Thought: Retrieve order status before responding Action: query_order_api(order_id='ORD-88291') Observation: { status: 'delayed', new_eta: '2026-07-02', cause: 'weather_disruption' }
2	Thought: Order delayed 3 days. Customer LTV = $2,400 (Platinum tier). Retrieve retention policy. Action: lookup_retention_policy(tier='platinum', issue='shipping_delay') Observation: { eligible_goodwill_credit: '$20', max_discount_next_order: '15%' }
3	Thought: Apply proactive credit and communicate new ETA. No escalation needed. Action: apply_credit(customer_id='CID-441', amount=20, reason='shipping_delay_goodwill') Observation: { success: true, credit_ref: 'CRD-9921' }
4	Final Response: Generate personalized message Content: Apology acknowledging 3-day delay, confirmed new ETA of July 2, $20 credit applied with confirmation code, 15% discount offered on next order. Tone calibrated to Platinum tier relationship.

The LangGraph pattern constructs workflows as directed acyclic graphs where nodes are specialized agents (inventory agent, pricing agent, communication agent) and edges are conditional transitions. This provides explicit control flow, deterministic execution paths, and lower latency than ReAct — typically 50–150ms vs. 500ms–2 seconds for LLM-based ReAct. LangGraph is preferred for high-volume, low-latency decisions such as real-time recommendation API calls; ReAct is preferred for complex context-dependent resolution workflows.

6.3 Guardian Agents and Human-in-the-Loop Design

Guardian agents are specialized agents whose sole function is to evaluate proposed actions before execution and either approve, modify, or escalate them to a human review queue. They enforce three

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

categories of constraints: business rule constraints (pricing floors, discount maximums, budget limits) implemented as deterministic rule evaluations; statistical anomaly detection (actions deviating more than N standard deviations from historical distributions in the same context are flagged); and regulatory and fairness constraints (actions producing materially different outcomes for customers in protected demographic groups are blocked regardless of business rule compliance). New agent deployments begin with narrow autonomy, expanding as track records accumulate and action distributions are validated against expected behavior.

VII. BIAS DETECTION AND MITIGATION

7.1 Taxonomy of Bias Sources

Bias in retail ML systems has documented business consequences — systematic underperformance for certain customer segments — and regulatory exposure through discriminatory pricing or offer eligibility. Table 3 identifies the principal sources with detection methods.

Table 3. Principal Bias Sources in Retail Personalization Systems

Bias Type	Mechanism	Retail Manifestation	Detection Method
Popularity bias	CF models over-recommend popular items that dominate training data; long-tail items systematically underexposed.	Small brands fail to surface despite genuine relevance; popular items enter a self-reinforcing cycle.	Gini coefficient on recommendation distribution vs. catalog distribution; coverage@K metric.
Exposure bias	Models train only on interactions with items that were shown. Items never recommended generate no data, creating a false irrelevance signal.	Entire product categories or new collections disappear from recommendations, reinforcing prior promotional decisions indefinitely.	IPS during training; counterfactual evaluation using ε-greedy exploration holdout traffic.
Price sensitivity bias	If training data over-represents high-spending customers, models underserve price-sensitive segments.	Cold-start customers see systematically higher-priced recommendations; retention rates differ across income proxies.	Stratified performance analysis by price quintile; demographic parity testing on offer distribution.
Feedback loop bias	Agent actions determine future training data, reinforcing original biases over time.	Progressive narrowing of recommendation diversity; entire product categories gradually vanish from surfacing.	Monitor catalog coverage and category distribution in recommendations over rolling 30/60/90-day windows.

7.2 Mitigation Techniques

Three mitigation approaches with strong theoretical grounding are deployed in production retail systems. Figure 9 presents the IPS-corrected loss formulation and the Maximum Marginal Relevance diversity enforcement mechanism.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

Figure 9. Bias Mitigation — IPS Loss Correction and MMR Diversity Enforcement

Bias Mitigation Techniques	
IPS-corrected loss	$L\_IPS(\theta) = (1/N)\ \Sigma_i\ (r_i / p(a_i \mid x_i)) \times loss(f\_\theta(x_i, a_i), r_i)$
Effect	Underexposed items with positive responses receive high training weight, correcting exposure bias.
MMR score	$MMR(i) = \lambda \times relevance(i, user) - (1-\lambda) \times max\_\{j \in S\}\ similarity(i, j)$
Effect	Items selected greedily to maximize relevance while penalizing similarity to already-selected items. λ controls relevance-diversity tradeoff.
Fairness reg.	Training objective: $L\_total(\theta) = L\_task(\theta) + \gamma \times \Omega\_F(\theta)$
Where	$\Omega\_F(\theta)$ penalizes disparities in average recommendation probability across protected groups or brand tiers. γ is a tunable fairness weight.

VIII. PRODUCTION SCENARIOS: END-TO-END INTEGRATION

8.1 Scenario 1 — Influencer-Triggered Dynamic Price Adjustment

A fashion retailer's agentic system detects a spike in organic social media mention volume for a specific outerwear SKU and executes the following pipeline within approximately four minutes of signal onset, without human intervention.

Figure 10. Production Pipeline: Influencer-Triggered Dynamic Pricing

End-to-End Pipeline: Social Signal → Autonomous Price Adjustment	
1	Signal Detection Kafka consumer processes real-time social media API data. An Isolation Forest anomaly detection model — trained on 90-day rolling mention velocity distributions — flags the SKU when its per-hour mention rate exceeds three standard deviations above its 7-day rolling mean. Alert includes a fine-tuned BERT sentiment classifier score (positive: 0.89) and source breakdown (67% Instagram, 21% TikTok, 12% Twitter).
2	Context Assembly Inventory agent queries online feature store: current stock (847 units across 3 warehouses), current price ($148), cost floor ($62), active promotions (none), margin floor (42%). Demand forecasting agent retrieves historical price elasticity estimate: $\varepsilon = -1.3$ (a 10% price increase reduces demand by 13%).
3	Impact Modeling Causal uplift model — trained on historical social spike / demand-curve pairs using a difference-in-differences estimator — predicts +340 units incremental demand above baseline over 48 hours at current price. Pricing optimization agent solves the constrained LP: maximize $P \times (baseline + social\_uplift(P))$ subject to margin and inventory constraints. Optimal price: $161 (8.8% increase).
4	Guardian Review Business rule checks: price < MSRP $195 ✓, gross margin 61.5% > 42% floor ✓. Anomaly check: 8.8% increase is within the 12th percentile of historical pricing actions for this category — within normal range ✓. Auto-approved.
5	Action Execution Price updated via pricing API (propagated to website, app, and marketplace feeds within 30 seconds). Active 10%-off email campaign for this SKU paused to avoid cannibalization. Item's featured placement rank increased in content management system.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

6	Feedback Loop Conversion rate and revenue per visitor tracked in 6-hour windows for 48 hours. Actual demand lift compared to model prediction. Delta updates the social-uplift model's priors. Causal model retrained weekly on accumulated historical events. Full causal attribution chain logged: social_spike_id → demand_model_output → price_recommendation → guardian_approval → execution_timestamp.

8.2 Scenario 2 — Subscription Churn Intervention

A subscription beauty box retailer runs a daily batch scoring job that processes all active subscribers through a DeepSurv model. The model encodes 90 days of behavioral history — box opening rates, product ratings, website visits, customer service contacts, delivery satisfaction scores — through a two-layer GRU encoder and outputs a survival curve per customer. The following pipeline is triggered for a customer whose P(cancel within 14 days) = 0.72.

Figure 11. Production Pipeline: Subscription Churn Intervention

End-to-End Pipeline: Survival Model → Personalized Retention Intervention	
1	Causal Attribution (SHAP Decomposition) SHAP layer on the survival model decomposes the churn risk score into feature contributions. Primary drivers for this customer: three consecutive boxes with average rating below 3.5 stars (SHAP: +0.31), no website visit in 21 days (+0.24), support contact about incorrect item 12 days ago (+0.19). Attribution drives intervention selection, not just explanation.
2	Intervention Selection (Uplift Model) Multi-arm uplift model — trained on historical A/B test data across five retention intervention types — predicts expected lift in P(retention): standard email (+0.08), personalized SKU swap offer for next box (+0.14), 20% discount on next 3 months (+0.21), 30-day pause option (+0.17), outbound advisor call (+0.28). Expected value calculation: lift × predicted LTV × (1 − intervention cost / LTV). SKU swap offer selected as best risk-adjusted action given current budget constraints.
3	Content Personalization Email content constructed directly from SHAP attributions: messaging acknowledges the product mismatch experience, offers a preference update survey (results fed back into the recommendation model), and presents three specific alternative product options selected by the two-tower recommendation system based on her stated preferences and out-of-subscription purchase history. Subject line variant selected by contextual bandit based on her historical email engagement patterns.
4	Outcome Tracking and Model Update Email delivery, open, click, survey completion, and subscription status tracked for 30 days. Retained or churned outcome stored as a labeled example in the uplift model's training dataset. Survival model predictions compared to actual retention rates across the full scored cohort monthly to detect distributional drift. Model retrained quarterly or when PSI on key features exceeds 0.2.

IX. GOVERNANCE: ENGINEERING REQUIREMENTS

Governance of agentic retail systems is most effectively implemented as a set of engineering requirements embedded in the system architecture from the start, not as policy documents consulted after deployment. Three requirements are non-negotiable.

9.1 Comprehensive Action Logging

Every action taken by any agent must be logged with: (1) the complete context vector observed at decision time; (2) all candidate actions considered and their model scores; (3) the action selected and approval pathway (auto-approved, guardian-modified, or human-approved); (4) the outcome observed; and (5) the model version that made the decision. This log is

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719435

the foundation for debugging, compliance audits, bias investigations, and model improvement. Systems without this log cannot be audited after the fact.

9.2 Model Drift Monitoring

Model performance must be monitored continuously against business metrics (conversion rate, retention rate, revenue per visit) and statistical metrics (prediction distribution drift via KL divergence, feature distribution drift via Population Stability Index). Alert thresholds should be set at the level of business significance: a drift that moves conversion rate by 0.5% requires action; a statistically detectable but commercially insignificant drift does not.

9.3 A/B Testing for Delayed Rewards

Agentic systems present unique testing challenges: rewards are delayed (a retention intervention's effect is measured over 30 days) and actions have network effects (showing item A to User 1 affects inventory and demand signals for User 2). The solution is user-level randomization with full-window holdout groups — users assigned to treatment or control cohorts at login and maintained in that assignment for the full experiment window, with customer-level lifetime outcomes measured at the cohort level.

X. IMPLEMENTATION ROADMAP

Table 4 presents a phased implementation roadmap ordered by technical dependency, not business priority. Phase 1 is the prerequisite for all subsequent phases without exception.

Table 4. Phased Implementation Roadmap for Retail Agentic AI

Phase	Focus Area	Key Technical Deliverables	Duration
1 — Foundation	Data infrastructure	Real-time feature store (Redis + Flink + offline warehouse); event streaming (Kafka); feature pipeline monitoring; feature freshness SLAs defined and enforced	3–6 months
2 — Core Models	Retrieval and churn	Two-tower retrieval model; LightGBM re-ranker; survival analysis churn model; offline evaluation framework (NDCG, AUC, calibration, stratified subgroup analysis)	4–6 months
3 — Real-Time Personalization	Online learning	Contextual bandit layer (LinUCB or Thompson Sampling); A/B testing infrastructure with full-window holdout groups; conversion tracking pipeline; feature freshness monitoring	3–4 months
4 — Agentic Orchestration	Agent framework	LangGraph for structured flows; ReAct for open-ended resolution; action APIs for pricing, promotions, and communications; guardian agent with rule engine and anomaly detection; human review queue	4–6 months
5 — Governance	Ongoing operations	Bias audit framework (IPS, fairness metrics, disparity testing); comprehensive action logging; model drift	Ongoing

		monitoring with business-significance alerts; regulatory compliance documentation	

The most consistent failure mode in retail AI implementations is infrastructural, not algorithmic. Teams that deploy sophisticated neural recommendation models on top of batch feature pipelines with multi-hour latency will not achieve real-time personalization regardless of model quality. Investment in Phase 1 infrastructure is the prerequisite for everything that follows.

CONCLUSION

Agentic AI in retail represents a genuine architectural shift: continuous real-time perception, multi-step planning, autonomous action execution, and closed-loop learning enable a qualitatively different form of customer engagement. The technical foundations — contextual bandits for offer optimization, two-tower networks for retrieval, survival analysis for churn prediction, demand elasticity models for pricing, and multi-agent orchestration for complex workflows — are production-validated building blocks, not research prototypes.

What separates successful from unsuccessful implementations is not algorithm choice but infrastructure discipline. Real-time feature stores, bias-aware training, comprehensive action logging, and graduated autonomy governance are not optional enhancements. They are the conditions under which agentic retail systems perform reliably at scale and earn the organizational trust required for sustainable deployment. Practitioners who treat them as afterthoughts will build systems that are technically impressive in demonstration and operationally fragile in production.

REFERENCES

[1] Chapelle, O., & Li, L. (2011). An empirical evaluation of Thompson sampling. Advances in Neural Information Processing Systems, 24, 2249–2257.

[2] Chen, M., Beutel, A., Covington, P., Jain, S., Belletti, F., & Chi, E. H. (2019). Top-K off-policy correction for a REINFORCE recommender system. Proceedings of WSDM '19, 456–464.

[3] Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for YouTube recommendations. Proceedings of RecSys '16, 191–198.

[4] Davenport, T. H., Guha, A., Grewal, D., & Bressgott, T. (2020). How artificial intelligence will change the future of marketing. Journal of the Academy of Marketing Science, 48(1), 24–42.

[5] Forrester Research. (2025). State of AI in Retail 2025. Forrester Research Inc.

[6] IDC. (2025). IDC FutureScape: Worldwide Retail Predictions 2025. International Data Corporation.

[7] Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). DeepSurv: Personalized treatment recommender system using a Cox proportional hazards deep neural network. BMC Medical Research Methodology, 18(1), 24.

[8] Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. Proceedings of WWW '10, 661–670.

[9] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 4765–4774.

[10] McKinsey & Company. (2024). The State of AI: Global Survey on Artificial Intelligence. McKinsey Global Institute.

[11] Nucleus Research. (2024). Personalization and AI ROI in Retail: Industry Case Studies. Nucleus Research.

[12] Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

[13] Schnabel, T., Swaminathan, A., Singh, A., Chandak, N., & Joachims, T. (2016). Recommendations as treatments: Debiasing learning and evaluation. Proceedings of ICML '16, 1670–1679.

[14] Shankar, V., Kalyanam, K., Setia, P., Golmohammadi, A., Tirunillai, S., Douglass, T., Hennessey, J., Bull, J. S., & Waddoups, R. (2021). How technology is changing retail. Journal of Retailing, 97(1), 13–27.

[15] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. Proceedings of ICLR '23.

[16] Yi, X., Yang, J., Hong, L., Cheng, D., Heldt, L., Kumthekar, A., Zhao, Z., Wei, L., & Chi, E. (2019). Sampling-bias-corrected neural modeling for large corpus item recommendations. Proceedings of RecSys '19, 269–277.

[17] Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., Zhu, X., & Gai, K. (2018). Deep interest evolution network for click-through rate prediction. Proceedings of AAAI '19, 5941–5948.