6) Customer Affinities (Brand/Category) — Decayed shares (Dirichlet-smoothed)

Goal & decision

Score each customer's **brand/category affinity** for **targeting** (audiences/eligibility) and **slotting** (what to surface first).

Feature sources

- **events** (views/ATC/purchases): `customer_id` , `brand_id` , `category_id` , `ts`
- **customers** (store outputs): `users.affinities[]`

Compact formula

Time-decayed event counts per brand (half-life h days; event age a_e days):

$$n_{c,b} = \sum_{e \in \mathcal{E}_{c,b}} 2^{-a_e/h}$$

Dirichlet-smoothed affinity (typ. $\alpha = 1$):

$$\text{affinity}_{c,b} = \frac{n_{c,b} + \alpha}{\sum_{b \in B} n_{c,b} + \alpha\,|B|}$$

Full expansion (worked)

Customer's events and parameters: $h = 30\,\text{days},\ \alpha = 1,\ B = \{B1, B2, B3\}$.
B1 ages $= \{0, 5, 20\}$; B2 age $= 10$; B3 age $= 40$.

$$n_{c,B1} = 2^{-0/30} + 2^{-5/30} + 2^{-20/30} = 1 + 2^{-1/6} + 2^{-2/3} \approx 2.521$$

$$n_{c,B2} = 2^{-10/30} = 2^{-1/3} \approx 0.794$$

$$n_{c,B3} = 2^{-40/30} = 2^{-4/3} \approx 0.397$$

$$\text{denom} = \sum_{b \in B} n_{c,b} + \alpha|B| = 2.521 + 0.794 + 0.397 + 3 = \mathbf{6.712}$$

$$\text{affinity}_{c,B1} = \frac{2.521 + 1}{6.712} = \mathbf{0.525}$$

$$\text{affinity}_{c,B2} = \frac{0.794 + 1}{6.712} = \mathbf{0.267}$$

$$\text{affinity}_{c,B3} = \frac{0.397 + 1}{6.712} = \mathbf{0.208}$$

Result & steering

- Persist **top-K** `{ id, score, kind }` per customer for **audiences** and **slotting**.
- Tune **half-life** h (30–60 d commonly). Use **brand** and/or **category** tracks.

- If events are **sparse**, fall back to **recent-top** brand/category.

▼ **metrics: affinities_example (with feature_sources)** — click to expand


```
metrics:
  affinities_example:
    half_life_days: 30
    alpha_smoothing: 1.0
    brands: [B1, B2, B3]
    customer_events:
      # event ages in days (most recent = 0)
      B1: [0, 5, 20]
      B2: [10]
      B3: [40]
    derived:
      n_counts:
        B1: 2.521   # 1 + 2^(-5/30) + 2^(-20/30)
        B2: 0.794
        B3: 0.397
      denom: 6.712  # sum(n_counts) + alpha*|B|
      affinities:
        B1: 0.525
        B2: 0.267
        B3: 0.208
feature_sources:
  - mongo.events
  - mongo.customers

</details>
```


7) Product Assortment Similarity / Clustering — Co-buy graph + cosine (ANN serving)

Goal & decision

Provide **"similar products"** for **PDP/PLP** and **clusters** for discovery/bundling, especially for the **long tail**.

Feature sources

- **orders** → basket co-occurrence pairs
- **product_graph** (derived) → `co_buy_counts` , `neighbors`
- **products** → `category_id` , `brand_id` , `price_cents` , `attributes` (attribute fallback)
- **Outputs** → `products.cluster_id` , `product_features.similar[]`

Compact formula

Co-buy row vector for item i (counts to other items):
$\mathbf{c}_i = (c_{i,1}, \ldots, c_{i,n})$

Cosine similarity:
$\cos(i,j) = \dfrac{\mathbf{c}_i \cdot \mathbf{c}_j}{\|\mathbf{c}_i\| \, \|\mathbf{c}_j\|}$

(Optional) PMI edge weighting → PPMI
$\mathrm{PMI}_{ij} = \log\left(\frac{p(i,j)}{p(i)\,p(j)}\right), \quad \mathrm{PPMI}_{ij} = \max(0, \mathrm{PMI}_{ij})$

Serving: build **ANN index** over item embeddings or normalized co-buy vectors; return **top-K neighbors**; re-rank with **margin/stock** constraints.

Full expansion (worked)

Toy co-buy counts (selected pairs):
$c_{1,2} = 10,\ c_{1,3} = 12,\ c_{1,5} = 3; \ \ c_{3,1} = 12,\ c_{3,2} = 5,\ c_{3,4} = 6; \ \ c_{2,5} = 8$ (others $= 0$).

Construct co-buy rows over shared index set $\{2,3,4,5\}$:
$\mathbf{c}_1 = [10,\ 12,\ 0,\ 3], \quad \mathbf{c}_3 = [5,\ 0,\ 6,\ 0].$

Cosine pieces (dot and norms):

$$\mathbf{c}_1 \cdot \mathbf{c}_3 = 10 \cdot 5 + 12 \cdot 0 + 0 \cdot 6 + 3 \cdot 0 \ = \ \mathbf{50}$$
$$\|\mathbf{c}_1\| = \sqrt{10^2 + 12^2 + 0^2 + 3^2} \ = \ \sqrt{253} \ \approx \ \mathbf{15.90597}$$
$$\|\mathbf{c}_3\| = \sqrt{5^2 + 0^2 + 6^2 + 0^2} \ = \ \sqrt{61} \ \approx \ \mathbf{7.81025}$$

Cosine similarity (P1 vs P3):
$\cos(1,3) = \dfrac{50}{15.90597 \cdot 7.81025} = 0.40248 \approx \mathbf{0.4025}$

Neighbor selection & clustering: repeat vs other items → pick **top-K neighbors**; cluster the neighbor graph via **Louvain**/**k-means**; write `products.cluster_id`.

Result & steering

- **PDP neighbors:** return top-K by cosine; re-rank to **penalize low-stock** and **low-margin** items.
- **Assortment:** use `cluster_id` for browse pages, **markdown cross-effects**, and **bundle** candidates.
- **Cold start:** if counts are sparse, fall back to **attribute/semantic** similarity until interactions accrue.

▼ **metrics: product_similarity_example (with feature_sources)** — click to expand

```yaml
metrics:
  product_similarity_example:
    items: [P1, P2, P3, P4, P5]
    co_buy_counts:
      P1: { P2: 10, P3: 12, P4: 0,  P5: 3 }
      P3: { P1: 12, P2: 5,  P4: 6,  P5: 0 }
    vectors:
      P1: [10, 12, 0, 3]     # to [P2,P3,P4,P5]
      P3: [5,  0,  6, 0]
    cosine:
      dot: 50
      norm_P1: 15.90597
      norm_P3: 7.81025
      cos_P1_P3: 0.4025
feature_sources:
  - mongo.orders
  - mongo.product_graph
  - mongo.products

</details>
```

8) PLP/Search Ranking (LTR) — BM25/ANN retrieval → pairwise RankNet

Goal & decision

Order PLP/Search results to **maximize relevance and conversion (nDCG@K)**, honoring **stock/price/promo freshness**.

Feature sources

- **search_logs**: query, impressions, clicks, purchases
- **product_features**: `bm25`, `semantic_ann`, `margin_pct`, `stock_cover_days`, `promo_flag`, `price_norm`
- **products** (fresh price/stock)

Compact formula

Pairwise RankNet loss (clicked i, not-clicked j):

$$\mathcal{L}_{ij} = \log\big(1 + \exp\big(-(s_i - s_j)\big)\big)$$

with **score function**:

$$s_k = \mathbf{w}^\top \mathbf{x}_k$$

Gradients wrt scores:

$$\frac{\partial \mathcal{L}_{ij}}{\partial s_i} = \sigma(s_j - s_i), \qquad \frac{\partial \mathcal{L}_{ij}}{\partial s_j} = -\sigma(s_j - s_i), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}$$

Full expansion (worked)

One clicked, one not-clicked: $s_i = 1.20,\ s_j = 0.50 \Rightarrow s_i - s_j = 0.70.$

Loss:

$$\mathcal{L}_{ij} = \log\big(1 + e^{-(s_i - s_j)}\big) = \log\big(1 + e^{-0.70}\big) = \log(1 + 0.496585) \approx \mathbf{0.403}$$

Gradient signal:

$$\sigma(s_j - s_i) = \sigma(-0.70) = \frac{1}{1 + e^{0.70}} \approx \mathbf{0.3318}$$

Interpretation: push s_i **up** by $+0.3318$ (descent uses $-\partial\mathcal{L}/\partial s_i$); push s_j **down** by 0.3318.

Scoring (illustrative)

Let

$$s_k \ = \ w_0 \ + \ w_1\,\mathrm{bm25}_k \ + \ w_2\,\mathrm{sem}_k \ + \ w_3\,\mathrm{stockCover}_k \ + \ w_4\,\mathrm{promo}_k, \qquad \mathbf{w} = (0,\ 0.6,\ 0.8,\ 0.05,\ 0.2).$$

Inputs:
A: $\mathrm{bm25} = 1.4,\ \mathrm{sem} = 0.9,\ \mathrm{stockCover} = 9,\ \mathrm{promo} = 1$
B: $\mathrm{bm25} = 1.2,\ \mathrm{sem} = 0.7,\ \mathrm{stockCover} = 6,\ \mathrm{promo} = 0$
C: $\mathrm{bm25} = 0.9,\ \mathrm{sem} = 0.5,\ \mathrm{stockCover} = 4,\ \mathrm{promo} = 0$

Scores:

$$s_A = 0.6\cdot 1.4 \ + \ 0.8\cdot 0.9 \ + \ 0.05\cdot 9 \ + \ 0.2\cdot 1 \ = \ 0.84 + 0.72 + 0.45 + 0.20 \ = \ \mathbf{2.21}$$

$$s_B = 0.6\cdot 1.2 \ + \ 0.8\cdot 0.7 \ + \ 0.05\cdot 6 \ + \ 0.2\cdot 0 \ = \ 0.72 + 0.56 + 0.30 + 0 \ = \ \mathbf{1.58}$$

$$s_C = 0.6\cdot 0.9 \ + \ 0.8\cdot 0.5 \ + \ 0.05\cdot 4 \ + \ 0.2\cdot 0 \ = \ 0.54 + 0.40 + 0.20 + 0 \ = \ \mathbf{1.14}$$

Ranking: $A > B > C$. Apply **OOS/stock/price freshness gates** and **exposure caps after** scoring.

Result & steering

- Use **BM25 ∪ ANN** for retrieval; rank with **LTR (RankNet/LambdaMART)**.
- If **price/stock features** are stale (> 5 min), **fall back** to lexical + business rules.

▼ **metrics: plp_ltr_example (with feature_sources)** — click to expand

```yaml
metrics:
  plp_ltr_example:
    pairwise_loss:
      s_i: 1.20
      s_j: 0.50
      loss_logistic: 0.403
      grad_sigma_sj_minus_si: 0.3318
    scoring_weights:
      w0: 0.0
      w1_bm25: 0.6
      w2_sem: 0.8
      w3_stockCover: 0.05
      w4_promo: 0.2
    items:
      - { id: A, bm25: 1.4, sem: 0.9, stock_cover_days: 9,  promo: 1, score: 2.21 }
      - { id: B, bm25: 1.2, sem: 0.7, stock_cover_days: 6,  promo: 0, score: 1.58 }
      - { id: C, bm25: 0.9, sem: 0.5, stock_cover_days: 4,  promo: 0, score: 1.14 }

feature_sources:
  - mongo.search_logs
  - mongo.product_features
  - mongo.products

</details>
```

9) Markdown Optimizer — Stochastic ladder (risk-aware), DP/MILP

Goal & decision

Choose a **price ladder by week** to maximize **terminal gross margin** (sales GM + salvage of leftovers) by the deadline, **accounting for demand uncertainty**.

Feature sources

- **inventory**: `sku_id`, `stock_units`, `age_days`
- **pricing**: allowed ladder per week (e.g., `{P0,P1,P2}`)
- **forecast**: demand **quantiles** per (week, price)
- **products**: `cogs_cents`, `salvage_cents`

Compact formula

Given stock S_0, weeks $t = 1, \ldots, T$, ladder price $p_t \in \mathcal{P}_t$, margin $m_t = p_t - \mathrm{COGS}$, demand $D_t(p_t)$, and sales $s_t = \min\big(S_{t-1},\, D_t(p_t)\big)$:

$$\max_{p_1,\ldots,p_T} \mathbb{E}\left[\sum_{t=1}^{T} s_t\, m_t \;+\; S_T \cdot \mathrm{salvage}\right]$$

Subject to:

$$S_t = S_{t-1} - s_t, \qquad 0 \le s_t \le S_{t-1}, \qquad p_t \in \mathcal{P}_t \quad (t = 1, \ldots, T).$$

Risk-aware: replace $\mathbb{E}[D_t]$ by a **quantile** (e.g., Q_{50} with a Q_{10} guard) or use **CVaR**.

Full expansion (worked)

Setup: $T = 3$, $S_0 = 100$, $\text{COGS} = 6000 \text{ cents}$, $\text{salvage} = 3000 \text{ cents}$.
Ladder options: $P_0 = 10{,}000$, $P_1 = 9{,}000$, $P_2 = 8{,}000$ (cents) $\Rightarrow$
$m(P_0) = 4000$, $m(P_1) = 3000$, $m(P_2) = 2000$ (cents).

Median demand forecasts:

$$\begin{aligned}
&D_1(P_0) = 20, \quad D_1(P_1) = 30, \quad D_1(P_2) = 38 \\
&D_2(P_0) = 15, \quad D_2(P_1) = 25, \quad D_2(P_2) = 35 \\
&D_3(P_0) = 10, \quad D_3(P_1) = 20, \quad D_3(P_2) = 30
\end{aligned}$$

Plan A $[P_0, P_1, P_2]$

$$\begin{aligned}
s_1 &= \min(100, 20) = 20, & GM_1 &= 20 \cdot 4000 = 80{,}000, \quad S_1 = 100 - 20 = 80 \\
s_2 &= \min(80, 25) = 25, & GM_2 &= 25 \cdot 3000 = 75{,}000, \quad S_2 = 80 - 25 = 55 \\
s_3 &= \min(55, 30) = 30, & GM_3 &= 30 \cdot 2000 = 60{,}000, \quad S_3 = 55 - 30 = 25
\end{aligned}$$

$$\begin{aligned}
\text{Salvage} &= S_3 \cdot 3000 = 25 \cdot 3000 = 75{,}000 \\
\textbf{Total GM}_A &= 80{,}000 + 75{,}000 + 60{,}000 + 75{,}000 = \mathbf{290{,}000}
\end{aligned}$$

Plan B $[P_1, P_1, P_1]$

$$\begin{aligned}
s_1 &= \min(100, 30) = 30, & GM_1 &= 30 \cdot 3000 = 90{,}000, \quad S_1 = 100 - 30 = 70 \\
s_2 &= \min(70, 25) = 25, & GM_2 &= 25 \cdot 3000 = 75{,}000, \quad S_2 = 70 - 25 = 45 \\
s_3 &= \min(45, 20) = 20, & GM_3 &= 20 \cdot 3000 = 60{,}000, \quad S_3 = 45 - 20 = 25
\end{aligned}$$

$$\begin{aligned}
\text{Salvage} &= S_3 \cdot 3000 = 25 \cdot 3000 = 75{,}000 \\
\textbf{Total GM}_B &= 90{,}000 + 75{,}000 + 60{,}000 + 75{,}000 = \mathbf{300{,}000}
\end{aligned}$$

Pick: Plan B (higher terminal GM).
Risk-aware note: down-weight low-price paths if Q_{10} demand raises **stockout** risk or violates **min-margin** constraints.

▼ **metrics: markdown_example (with feature_sources)** — click to expand

```
metrics:
  markdown_example:
    sku_id: "SKU123"
    stock_units: 100
    cogs_cents: 6000
    salvage_cents: 3000
    ladder_prices_cents:
      W1: [10000, 9000, 8000]
      W2: [10000, 9000, 8000]
      W3: [10000, 9000, 8000]
    margins_cents: { 10000: 4000, 9000: 3000, 8000: 2000 }
    demand_med:
      W1: { 10000: 20, 9000: 30, 8000: 38 }
      W2: { 10000: 15, 9000: 25, 8000: 35 }
      W3: { 10000: 10, 9000: 20, 8000: 30 }
    plans:
      plan_A: [10000, 9000, 8000]
      plan_B: [9000, 9000, 9000]
    totals_gm_cents:
      plan_A: 290000
      plan_B: 300000

feature_sources:
  - mongo.inventory
  - mongo.pricing
  - mongo.forecast
  - mongo.products

</details>
```


10) Price Elasticity (GATED) — Log-log demand; Lerner rule (planning, not markdown)

Gate to enable

Seller controls `product.price` and has **≥ 5 distinct price points/SKU** (or a price-test plan). Otherwise keep **gated OFF**.

Goal & decision

Estimate **own-price elasticity** to inform **base price**. Use with guardrails (**min margin**, **MAP**, **competitor checks**).

Feature sources

- **price_history**: per-SKU weekly `price_cents`, `units`
- **promos**: promo flags (control for discount effects)
- **seasonality**: week/holiday dummies
- **products**: `cogs_cents` (margins)

Compact formula

Log–log demand (per SKU k):

$$\ln Q_t \;=\; \alpha \;+\; \varepsilon \,\ln P_t \;+\; \boldsymbol{\beta}^\top \mathbf{Z}_t \;+\; \epsilon_t$$

Elasticity estimate (simple OLS slope):

$$\hat{\varepsilon} \;=\; \mathrm{SLOPE}\big(\ln Q,\; \ln P\big)$$

Lerner pricing (planning heuristic, if marginal cost MC known):

$$\frac{P - MC}{P} \;=\; -\frac{1}{\hat{\varepsilon}} \quad\Longrightarrow\quad P^{\star} \;=\; \frac{MC}{1 + 1/\hat{\varepsilon}}$$

Use with caution: need $\hat{\varepsilon} < -1$ for an interior optimum; always respect MAP, min-margin, and competitor-band constraints.

Full expansion (worked)

Toy weekly data (one SKU): $(P, Q) = (100, 80), (110, 70), (90, 96), (95, 92), (105, 75), (115, 68).$

Log transform (Excel): `=LN(P)` , `=LN(Q)` for each row to obtain $\ln P_t,\ \ln Q_t$.

Elasticity estimate (OLS slope on logs):

$$\hat{\varepsilon} = \frac{\mathrm{Cov}(\ln P, \ln Q)}{\mathrm{Var}(\ln P)} \approx -\mathbf{1.20} \quad \text{(illustrative)}.$$

Unconstrained Lerner price (planning heuristic): let $MC = \mathrm{COGS} = 60$ (same units as price).

$$P^{\star} = \frac{MC}{1 + 1/\hat{\varepsilon}} = \frac{60}{1 + 1/(-1.20)} = \frac{60}{1 - 0.8333} = \frac{60}{0.1667} \approx \mathbf{360}.$$

Guardrails (retail): enforce **max markup band** (e.g., $\pm 10\%$–15%), **min-margin/MAP**, and **competitor match band** (e.g., $\pm 5\%$).
Use $\hat{\varepsilon}$ to simulate candidate prices P: predict $\widehat{Q}(P)$ from the log model, then compute **revenue** $R(P) = P\,\widehat{Q}(P)$ and **margin** $M(P) = (P - MC)\,\widehat{Q}(P)$; choose **within guardrails**.
If $|\hat{\varepsilon}|$ is **unstable** (few price points, promo confounding), keep **gated OFF** and gather more variation or run controlled price tests.

▼ **metrics: elasticity_example (with feature_sources)** — click to expand

```yaml
metrics:
  elasticity_example:
    sku_id: "SKU789"
    cogs: 60.0
    weekly_observations:
      - { price: 100.0, units: 80 }
      - { price: 110.0, units: 70 }
      - { price: 90.0,  units: 96 }
      - { price: 95.0,  units: 92 }
      - { price: 105.0, units: 75 }
      - { price: 115.0, units: 68 }
    ln_series_ready: true   # compute lnP, lnQ in Excel
    estimated_elasticity: -1.20   # from SLOPE(lnQ,lnP)
    lerner_unconstrained_price: 360.0
    guardrails:
      max_change_pct: 0.15
      min_margin_pct: 0.20
      map_respected: true
      competitor_match_band_pct: 0.05

feature_sources:
  - mongo.price_history
  - mongo.promos
  - mongo.products
```

</details>