Price as Structure, Not Feature:

Three Problems in Commerce Search Where Getting Price Wrong Breaks Everything Else

Anjan Goswami, Ph.D.

Abstract

Machine learning systems in e-commerce—search ranking, faceted navigation, recommendation—encode implicit assumptions about price. These assumptions are usually wrong, and the failures they produce appear to be model failures (bad rankings, low conversion, revenue leakage) but are actually economic failures. This paper argues that price is not one signal among many in a feature vector. It is a **structural variable** that determines what the customer is looking for, what "relevant" means for a given query, and how marketplace value should be allocated across the catalog. Drawing on systems deployed at two major e-commerce platforms, we present three problems—each producing >2% revenue lift in controlled experiments—that illustrate the argument at increasing levels of sophistication: (1) price defines the consideration set, and imposing uniform price ranges in faceted navigation misaligns navigation with customer intent; (2) price conditions relevance, and treating "sort by price" as a simple ordering operation ignores intent-conditional price expectations; (3) price determines the optimal marketplace value allocation, and ranking systems that optimize for conversion implicitly misallocate impressions away from the revenue-generating catalog. Each problem was initially misdiagnosed as a model quality issue. Each was solved only after recognizing price as the structural cause. Together, they form a hierarchy: you cannot get impression allocation right if your training target is price-corrupted, you cannot get targets right if relevance is price-agnostic, and you cannot get relevance right if the consideration sets are wrong. The cumulative revenue impact suggests that commerce ML systems that have not addressed price structurally at every layer are leaving significant value on the table.

1 Introduction: Why ML Systems Get Price Wrong

Commerce search and recommendation systems are, at their core, resource allocation engines. They allocate a scarce resource—user attention, operationalized as impressions on a search results page—across a large catalog of items. The allocation is governed by a ranking model that assigns scores to items for a given query, and the items with the highest scores receive the most impressions.

Price enters these systems in one of two ways, both inadequate.

Price as a feature. The ranking model includes price as one input among hundreds—alongside textual relevance, behavioral signals, item attributes, seller quality. The model learns some function of price that improves prediction. This treats price as information, equivalent in kind to title match or click-through rate.

Price as a filter. The navigation system offers price-range facets ($0–$25, $25–$50, $50–$100) or a "sort by price" operation. This treats price as an orthogonal axis that the customer controls independently of relevance.

Both treatments are wrong. Price is neither a feature nor a filter. It is a **structural variable** that shapes the optimization landscape of the entire system:

- Price defines **what the customer is looking for**. A customer browsing HDTVs at $200 and one browsing at $1,000 are in different markets with different consideration sets, different quality

expectations, and different competitive alternatives. The price tier is not a filter applied after the customer forms intent—it is constitutive of the intent itself.

- Price conditions **what "relevant" means**. A query for "iPhone" has an expected price range. An item at $5 is not relevant to that query regardless of its textual match, and an item at $1,200 may be relevant even with weaker lexical overlap. Relevance is not price-independent.
- Price determines **what the model learns to optimize**. When ranking models use behavioral signals (clicks, sales, cart adds) as training targets, cheap items generate disproportionate signal because they convert at higher rates. The model learns that cheap items are "good" items—not because customers prefer them, but because the target is confounded by price. The model is optimizing for the wrong thing, and no amount of feature engineering or architectural improvement can fix it because the corruption is in the objective, not the model.
- Price determines **the marketplace's value allocation**. Even with a perfectly trained model, the impression distribution may be revenue-suboptimal. A conversion-optimized ranking overexposes cheap high-converting items and under-exposes moderately priced items that generate more revenue per transaction. The ranking system is maximizing the wrong objective for the marketplace.

These four roles form a hierarchy, and the remainder of this paper walks through three problems—each discovered in production, each initially misdiagnosed as a model problem, each solved by addressing price structurally—that correspond to the first three levels. Each intervention produced >2% revenue lift in controlled A/B experiments at scale. At the transaction volumes of major e-commerce platforms, 2% revenue lift represents hundreds of millions of dollars annually.

2 Problem 1: Price Defines the Consideration Set

2.1 The Standard Approach and Its Failure

Faceted navigation—the left-panel filters on an e-commerce search page—typically presents price ranges as uniform buckets: $0–$25, $25–$50, $50–$100, $100–$200, $200+. These ranges are set manually, often globally or at coarse category granularity, and remain static.

This approach imposes a structure on the catalog that rarely matches the actual distribution of products or customer intent. Consider high-definition televisions. The price distribution of HDTVs is multimodal: a cluster of entry-level models around $200, a cluster of mid-range models around $500, and a cluster of premium models around $1,000. Uniform $100-wide buckets split the entry-level cluster across two facets ($100–$200 and $200–$300), merge entry-level and mid-range in a $200–$500 bucket, and present a sparse $500–$1,000 bucket that mixes the tail of mid-range with the body of premium.

The customer using these facets is trying to express a product-tier preference: "show me entry-level TVs" or "show me premium TVs." The uniform buckets **misalign the navigation with the intent**. The customer clicks a bucket that partially matches their intended tier, sees a mix of products from different tiers, and either abandons the facet or refines further—friction that reduces engagement and conversion.

2.2 Kernel Density Estimation for Natural Price Clusters

The solution is to let the data define the facet boundaries. For each product category, we estimate the price density function using **kernel density estimation** (KDE) and place facet boundaries at the **valleys** (local minima) between density peaks.

Given a set of item prices $\{p_1, p_2, \ldots, p_n\}$ in a category, the kernel density estimate is:

$$\hat{f}(p) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{p - p_i}{h}\right)$$

where K is a kernel function (we used Gaussian) and h is the bandwidth parameter controlling the smoothness of the density estimate. The facet boundaries are the local minima of $\hat{f}(p)$—the price points where the fewest products exist, which are the natural "gaps" between product tiers.

Why KDE over alternatives:

- **K-means clustering** requires specifying the number of clusters *a priori* and assumes roughly equal-sized, roughly spherical clusters. Price distributions are highly skewed—a dense mass of items at low price points, a sparse tail of premium items. K-means splits the dense mass into artificial clusters and merges sparse high-end tiers. It imposes structure rather than discovering it.
- **Quantile-based bins** (equal number of items per bin) break natural product tiers. If 60% of HDTVs are \$200–\$300, quantile bins split this cluster in half to equalize bin sizes. The resulting facets separate products that belong together.
- **KDE** imposes no assumption on the number of tiers, their sizes, or their spacing. The number of facets emerges from the data—determined by the number of density peaks, which correspond to real product tiers defined by manufacturing price points, feature tiers, and brand positioning. The bandwidth h is the only tuning parameter, and it controls resolution (how fine-grained the tier structure is) rather than cluster count.

The resulting facets are **category-specific**: HDTVs might have 3 tiers, laptop accessories might have 5, furniture might have 4. The number of facets, their widths, and their boundaries all vary because they reflect the actual product landscape, not an imposed grid.

2.3 Price Clusters as Consideration Set Boundaries

The economic interpretation is that price tiers correspond to **consideration sets** in the discrete choice sense. A customer in the market for an HDTV forms a consideration set—the subset of products they will evaluate before making a purchase decision. This consideration set is defined, in significant part, by price tier: the customer shopping for an entry-level TV at \$200 does not consider \$1,000 premium models, and vice versa. The products within a tier are substitutes; products across tiers are not.

KDE-derived facets align the navigation with these consideration sets. When a customer clicks the \$180–\$350 facet (a boundary derived from the density valley between the entry and mid-range peaks), they see exactly the products they would consider—no mixing of tiers, no splitting of natural clusters. The facet *is* the consideration set boundary, estimated from the data.

2.4 Result

Replacing uniform price facets with KDE-derived facets produced >2% revenue lift in controlled experiments. The improvement came primarily from reduced navigation friction: customers found the products matching their tier preference faster, with fewer facet clicks and fewer abandoned searches. Engagement with price facets increased, indicating that the facets were now expressing distinctions customers actually cared about.

3 Problem 2: Price Conditions Relevance

3.1 The "Sort by Price" Problem

E-commerce platforms universally offer a "sort by price: low to high" option. The implementation is typically a simple sort: rank items by price ascending. This is what the customer asks for, and it appears to be a trivially correct operation.

It is not. Consider the query "iPhone." A simple price sort surfaces \$3 screen protectors, \$5 cable adapters, and \$10 cases before any actual iPhone. The items are cheap, and they have

lexical overlap with the query (they contain the word "iPhone" in their titles), so they survive any pre-filtering. The customer who asked for "sort by price" wanted the cheapest *iPhone*, not the cheapest item containing the word "iPhone."

The problem is that "sort by price" implicitly assumes that all items in the result set are equally relevant, and the customer's only preference dimension is price. This assumption is false. Relevance varies dramatically across the result set, and a $5 accessory is not relevant to the query "iPhone" regardless of its price.

3.2 Relevance-Gated Price Sorting

The solution is a two-stage architecture that **gates on relevance first, then sorts by price within relevance bands**.

Stage 1: ML relevance ranking. An ML ranking model scores all candidate items by query-item relevance. This model captures semantic match, category alignment, behavioral signals, and all the standard relevance features. The output is a relevance score s_i for each item i given the query q.

Stage 2: Banded price sort. The items displayed on a results page (e.g., 48 items) have relevance scores spanning some range $[s_{\min}, s_{\max}]$. This range is divided uniformly into k bands:

$$B_j = \left[s_{\min} + (j-1) \cdot \frac{s_{\max} - s_{\min}}{k}, \quad s_{\min} + j \cdot \frac{s_{\max} - s_{\min}}{k} \right], \quad j = 1, \ldots, k$$

Within each band B_j, items are sorted by price ascending. The bands are then presented in order from highest relevance (B_k) to lowest (B_1).

The result: the customer sees cheap items first—which is what they asked for—but the *most relevant* cheap items come before *less relevant* cheap items. The banding is invisible to the customer; the page looks like a price-sorted result. But the $5 iPhone cable no longer outranks the $699 iPhone, because the iPhone has a much higher relevance score and sits in a higher band.

The parameter k controls the tradeoff:

- Large k (many narrow bands) $\rightarrow$ approaches pure relevance ranking, with price serving only as a tiebreaker within near-identical relevance scores
- Small k (few wide bands) $\rightarrow$ approaches pure price sort, with relevance serving only as a coarse filter
- $k = 1 \rightarrow$ pure price sort (the original behavior)
- $k = n$ (where n is the number of items) $\rightarrow$ pure relevance ranking

The optimal k was determined experimentally. The value that maximized revenue was neither extreme—it preserved the customer's expectation of a price-influenced ordering while eliminating the most egregious relevance violations.

3.3 Price Expectations as Revealed Willingness-to-Pay

The deeper insight is that **every query has an implicit price expectation**, and this expectation is a revealed willingness-to-pay signal. The distribution of purchase prices for the query "iPhone" is concentrated in the $700–$1,200 range. The distribution for "iPhone case" is concentrated in the $10–$40 range. These distributions are not features of the items—they are features of the *query intent*.

The relevance model implicitly learns these distributions through behavioral training data: items purchased after the query "iPhone" are disproportionately actual iPhones, so the model assigns high relevance to items in the iPhone price range. The banded price sort leverages this learned price-intent association without requiring an explicit price expectation model.

This connection between query-level price distributions and customer intent generalizes beyond "sort by price." It implies that **any** ranking operation should be conditioned on the price expectations associated with the query. A relevance model that ignores price expectations will systematically misrank items whose price is inconsistent with the query intent—not because the items are textually irrelevant, but because they are *economically* irrelevant to the customer's implied consideration set.

3.4 Result

Relevance-gated price sorting produced >2% revenue lift in controlled experiments. The gain came from two sources: eliminating the accessory problem (relevant products no longer displaced by cheap irrelevant items in price-sorted results) and surfacing better price-intent matches that increased conversion for customers using the price sort option.

4 Problem 3: Price Determines the Marketplace Value Allocation

This section describes two connected interventions—an empirical diagnosis and its resolution—that together address the deepest layer of the price problem: how the ranking system's optimization objective interacts with price to determine the marketplace's value allocation.

4.1 The Observation: Sales Peak and Revenue Peak Diverge

In any e-commerce catalog, the relationship between price, sales volume, and revenue is non-trivial. We analyzed this relationship through the lens of **impression allocation**: how the ranking system distributes visibility across the catalog, and what marketplace outcomes that distribution produces.

We binned catalog items by their impression frequency—how often each item appeared in search results—and for each bin computed three quantities: average sales volume, average item price, and average revenue (sales $\times$ price). The resulting distributions revealed a structural misalignment:

- **High-impression items** (items the ranking system showed most frequently) had **high sales volume** but **low average price**. These were cheap, high-converting items that the sales-optimized ranking model favored because they maximized the predicted target (conversion probability or sales count).
- **Low-impression items** (items the ranking system rarely showed) had **low sales volume** but **high average price**. These were moderately to highly priced items that converted less frequently per impression but generated substantially more revenue per transaction when they did convert.
- **Revenue** (sales $\times$ price) **peaked at a different point in the impression distribution than sales volume**. The revenue peak was shifted toward lower-impression, higher-priced items.

The geometry is clear: the ranking system was allocating the most impressions to items that generated the most sales but not the most revenue. The system was implicitly solving the wrong optimization problem for the marketplace.

4.2 Why Conversion-Optimized Ranking Misallocates Impressions

The root cause is that ML ranking models in commerce are typically trained on behavioral targets—click-through rate, add-to-cart rate, purchase rate—that are **confounded by price**. Cheap items have systematically higher conversion rates, not because they are better products but because price-sensitive customers (who comprise a large fraction of traffic) are more likely to purchase

cheap items. The model learns this association and ranks cheap items higher, which gives them more impressions, which generates more sales data, which reinforces the model's belief that cheap items are "better."

This creates a feedback loop:

Cheap items rank higher → more impressions → more sales → higher conversion signal → model ranks them hi

The loop converges on a **sales-optimal impression allocation** that is **revenue-suboptimal**. The system has no mechanism to discover that shifting some impressions from cheap high-converting items to moderately priced lower-converting items would increase total marketplace revenue—because the moderately priced items never receive enough impressions to generate the behavioral signal that would cause the model to promote them.

This is a special case of the exploration-exploitation tradeoff, but the key insight is that the tradeoff here is **driven by price confounding in the training target**, not by insufficient data per se. Even with infinite data, a conversion-optimized model will converge on the sales-optimal allocation, not the revenue-optimal one, because the target itself is biased toward cheap items.

4.3 Phase 1: Heuristic Impression Redistribution

The first intervention was a direct, heuristic correction applied within the existing Best Match ranking system. We identified items in the top revenue tier—items with high revenue-per-transaction but low impression share—and applied a **multiplicative rank boost** to increase their visibility.

The boost was intentionally conservative. The goal was not to promote expensive items aggressively (which would crater conversion and damage customer experience) but to **nudge the impression distribution** toward the revenue peak—to give revenue-generating items a fair opportunity to be seen, not to guarantee they would be purchased.

This heuristic intervention served as a proof of concept. Revenue increased because items that generated the most value per sale were finally receiving enough impressions to produce sales. The ranking system had been starving them of visibility, not because they were bad products, but because their conversion rates—depressed by higher prices—signaled "low quality" to the conversion-optimized model.

However, the heuristic had limitations. The boost parameters were manually tuned, category-specific adjustments were labor-intensive, and the intervention operated *on top of* a fundamentally misaligned ranking model. The model was still learning the wrong objective; the boost was compensating for that error rather than fixing it.

4.4 Phase 2: Price-Normalized Training Targets

The principled solution was to fix the training target itself. Instead of training the ranking model on raw behavioral signals, we **price-normalized the target variable**:

$$y_i = \frac{\text{sales}_i}{\sqrt{\min(p_i,\, p_{\text{cap}})}}$$

where p_i is the item's price and p_{cap} is a category-specific price ceiling that prevents extreme prices from dominating the normalization.

Why square root, not linear? This was discovered empirically, not derived theoretically. We evaluated three normalization functions:

Normalization	Target	Effect
None	$y_i = \text{sales}_i$	Over-promotes cheap items (status quo)
Linear	$y_i = \text{sales}_i / p_i$	Over-promotes expensive items
Square root	$y_i = \text{sales}_i / \sqrt{p_i}$	Balances conversion and revenue

Table 1: Effect of price normalization on ranking behavior.

No normalization produced the observed problem: the model ranked cheap items highest because they had the highest raw sales signal per impression. Linear normalization ($1/p$) overcorrected: it divided by price so aggressively that expensive items with even modest sales were promoted above popular items, damaging conversion rates and user experience. Square root normalization hit the empirical sweet spot—it reduced the unfair advantage of cheap items without creating an unfair advantage for expensive ones.

The economic interpretation of the normalization function is that it implicitly encodes a **demand curve**. The square root normalization says: "expected conversion decays with the square root of price." This is an elasticity assumption. If actual demand elasticity were linear (conversion $\propto 1/p$), then linear normalization would be correct. If demand were perfectly inelastic (conversion independent of p), no normalization would be correct. The empirical finding that $\sqrt{p}$ was optimal implies that the true demand elasticity lies between these extremes—which is consistent with standard economic models of consumer demand.

The cap p_{cap} handles the tail: for very expensive items (luxury goods, high-end electronics), the square root normalization alone would produce extreme target values. The cap ensures that above a category-specific threshold, additional price does not further boost the normalized target.

4.5 Why the Two Phases Tell One Story

The impression distribution analysis (Phase 1) was the **diagnosis**: it revealed that the ranking system's implicit optimization objective was misaligned with marketplace value. The heuristic boost demonstrated that the diagnosis was correct—redistributing impressions toward the revenue peak increased total revenue.

The price-normalized target (Phase 2) was the **cure**: it encoded the sales-revenue tradeoff directly into the model's learning objective. The model itself learned to allocate impressions in a way that balanced conversion and revenue, eliminating the need for external heuristic corrections.

The connection between the two phases is critical: **the impression distribution analysis explains why price normalization works**. Without the empirical observation that sales and revenue peak at different impression levels, price normalization appears to be an ad hoc correction to a feature engineering problem. With the observation, it is revealed as the solution to a structural misalignment between the ranking system's optimization objective and the marketplace's value function.

This same price normalization was independently discovered and validated at a second major e-commerce platform, where it also produced >2% revenue lift—suggesting that the underlying economic structure (conversion-optimal $\neq$ revenue-optimal) is general to commerce ranking, not specific to a particular platform.

4.6 Result

The combined intervention—heuristic boost followed by price-normalized targets—produced >2% revenue lift in controlled experiments at both platforms where it was deployed. The improvement came from redistributing impressions from the lowest-revenue transactions (cheap items with high conversion but low revenue per sale) to moderate-revenue transactions (moderately priced items with lower conversion but higher revenue per sale), without significantly reducing total unit sales. The marketplace generated more revenue from approximately the same transaction volume by allocating its attention budget more efficiently.

5 The Hierarchy: Why Order Matters

The three problems form a hierarchy in which each level subsumes the previous:

Level	Problem	What Price Determines	Consequence of Error
1	Consideration sets	What the customer evaluates	Wrong products shown
2	Relevance conditioning	What "relevant" means	Irrelevant results survive
3	Value allocation	What "optimal" means	Revenue left on the table

Table 2: The price hierarchy in commerce search systems.

The hierarchy is not merely organizational—it is **logically dependent**. Level 3 (value allocation via price-normalized targets) assumes that the items being ranked are relevant to the query. If Level 2 (price-conditioned relevance) is not addressed, the ranking model will price-normalize irrelevant items, promoting expensive irrelevant items alongside expensive relevant ones. Level 2 assumes that the consideration sets presented to the customer are coherent. If Level 1 (price clusters) is not addressed, the customer navigates into a mixed set of products from different tiers, and relevance scoring operates on an incoherent candidate pool.

Each level, addressed in isolation, produces >2% revenue lift. Addressed together, the effects compound—not additively (they operate on different mechanisms) but multiplicatively in the sense that each level creates the conditions under which the next level's intervention is effective.

6 Implications for Commerce ML Systems

6.1 The Diagnostic Checklist

Any commerce ML system can be audited against the three levels:

1. **Are your price facets data-driven?** If price ranges are manually set or uniform, you are likely splitting natural product tiers and misaligning navigation with customer intent. Apply KDE to your category price distributions and compare the resulting facets to your current ones. If they differ substantially, your navigation is costing you revenue.

2. **Does your "sort by price" respect relevance?** If price sorting is a simple ascending sort on the full result set, you are almost certainly surfacing irrelevant items above relevant ones for any query with a natural price expectation. Test relevance-gated price sorting and measure the impact on conversion and revenue in the price-sort segment.

3. **Is your training target price-confounded?** Plot the relationship between item price and your ranking model's target variable (conversion, sales, CTR). If there is a strong negative correlation—cheap items systematically have higher target values—your model is learning to promote cheap items. Test price-normalized targets and measure the effect on the revenue distribution across the catalog.

6.2 Why Economists and ML Scientists Must Collaborate

Each of the three problems sits at the intersection of economics and machine learning, and each was misdiagnosed when approached from only one discipline.

ML scientists diagnosed the accessory problem (Problem 2) as a relevance model failure and attempted to fix it with better features and more training data. The fix required recognizing that **relevance is conditioned on price expectations**—an economic insight about consideration sets and willingness-to-pay.

ML scientists diagnosed the cheap-item bias (Problem 3) as a model calibration issue and attempted to fix it with sample weighting and loss function adjustments. The fix required recognizing that **the training target encodes an implicit demand curve**—an economic insight about price elasticity and revealed preference.

Economists, conversely, would recognize the demand curve and elasticity structure but might not identify the **training pipeline** as the source of the problem. The model is not estimating demand incorrectly because it lacks economic structure—it is estimating demand incorrectly because **the target variable it learns from is a biased sample of the true demand function**, biased by the ranking system's own impression allocation. This feedback loop between model training and marketplace outcomes is an ML systems problem that requires ML systems thinking to diagnose.

The most effective commerce science teams staff economists and ML scientists together, with a shared understanding that the ranking system is simultaneously a prediction engine (ML) and a resource allocation mechanism (economics). Price is the variable where this dual nature is most consequential.

6.3 Price Normalization as an Implicit Demand Curve

The square root normalization deserves further theoretical attention. The finding that $y/\sqrt{p}$ outperforms both y and y/p as a training target implies a specific functional form for the relationship between price and conversion:

$$\mathbb{E}[\text{conversion} \mid p] \propto \frac{1}{\sqrt{p}}$$

This corresponds to a demand elasticity of -0.5: a 1% increase in price is associated with a 0.5% decrease in conversion probability. This is a moderate elasticity, consistent with commerce settings where customers are price-sensitive but not exclusively price-driven.

The fact that this elasticity was estimated *implicitly*—by finding the normalization function that maximized revenue in a controlled experiment, rather than by estimating a demand model directly—raises an interesting methodological point. The ranking system's A/B test is, in effect, a **revealed preference experiment**: by testing different normalization functions, we are testing different assumptions about the demand curve, and the function that maximizes revenue reveals the true elasticity structure.

This suggests a general approach: when designing training targets for commerce ML systems, **the normalization function is a design variable that encodes economic assumptions about the demand structure**, and it should be tuned experimentally rather than set by convention. Different categories may have different optimal normalizations—reflecting different elasticity structures—and the optimal normalization may change over time as competitive dynamics and customer price sensitivity evolve.

7 Conclusion

Price is the most consequential variable in commerce search, and it is systematically undertreated. ML systems encode it as a feature; navigation systems encode it as a filter; neither treatment captures its structural role in defining customer intent, conditioning relevance, and determining marketplace value allocation.

The three problems presented here—each producing >2% revenue lift in controlled experiments at major e-commerce platforms—demonstrate that getting price right is not a marginal improvement. It is a foundational correction that changes what the system optimizes, what it shows customers, and how it allocates the marketplace's most scarce resource: attention.

The revenue impacts compound across levels. A commerce ML system that has not addressed price structurally at the consideration set level, the relevance level, and the training target level is

likely leaving 5–10% or more in cumulative revenue on the table—not from model quality issues, but from economic structure issues that no amount of model improvement will resolve.

Price is not a feature. It is the structure.