CASE STUDY

Building Segmentation AI from Zero to Enterprise Launch

How Architecture Redesign, Cost Discipline, and an Intent-Driven Product Model Scaled Audience Discovery to Billions of Profiles at Adobe

Anjan Goswami | Head of AI & Data Science, Adobe Experience Platform | 2019–2023

Executive Summary

When I joined Adobe's Experience Platform team in 2019, I inherited a team whose manager had left weeks earlier and a prototype for Segmentation AI—an audience discovery product that would automatically surface customer segments from Adobe's unified profile data. The product had a hard delivery deadline, three enterprise customers waiting, and a pricing team that required 75% gross margin on any shipped AI capability.

The Alpha prototype used a deep learning pipeline for dimensionality reduction and clustering. It worked on 2 million profiles at $4K/month per customer. It could not scale to production data volumes (100M–5B profiles), and at projected costs, it would never meet margin targets.

Over the next nine months, I directly managed the team, redesigned the architecture from deep learning to distributed Spark-native clustering, built a driver-based cost model with the pricing team, and shipped a Beta that scaled to 5 billion stitched profiles at a per-customer cost reduction exceeding 70%.

But the most impactful change was a product insight, not an infrastructure one. Customers didn't want to browse 30–40 clusters—they already knew what audience they were looking for. I proposed a search interface where customers described their campaign intent, built a Bayesian ranking formulation with a statistician to score clusters against those keywords, and added Approximate Nearest Neighbor expansion to find similar profiles across all clusters. This search-rank-expand workflow transformed Segmentation AI from a tool data scientists appreciated into a platform marketers could use directly. We deployed with Verizon, Home Depot, and CBSi—all actively using the product in production.

Outcome:	2M → 5B profile capacity; 70%+ cost reduction; 75% margin; intent-driven audience discovery
Customers:	Verizon Wireless, Home Depot, CBSi (CBS Interactive)
Timeline:	Alpha (Dec 2019 – Apr 2020) → Beta (May – Nov 2020)
Role:	Direct management of the team; architecture, cost modeling, product design, Bayesian ranking formulation, customer delivery

1 The Situation I Walked Into

Adobe's Experience Platform had invested heavily in the **Unified Profile Service (UPS)**—a system that stitched customer identities across Adobe Audience Manager (AAM) and Adobe Analytics (AA) into a single profile graph. The platform processed billions of profile events for the world's largest enterprises.

Segmentation AI was meant to be the intelligence layer on top of this data: given a customer's stitched profiles, automatically discover meaningful audience segments without requiring manual rule writing. For enterprise marketers at scale, this was a significant capability gap—rules couldn't keep up with the combinatorial complexity of multi-channel behavioral data.

1.1 What I Found

- **No manager.** The team's manager had departed before I started. The engineers had been working semi-autonomously, building what they understood the product to be.
- **A working but unscalable prototype.** The Alpha pipeline used a deep learning model for dimensionality reduction, followed by clustering. Training was sequential on a single NC24 GPU machine (5 hours), and the full pipeline took 8 hours. It could process approximately 2 million profiles per run.
- **Real customers waiting.** CBSi was already in Alpha. Verizon Wireless and Adobe.com were queued. These were not hypothetical—they had expectations and timelines.
- **A pricing team with hard margin requirements.** Adobe's Digital Experience business operated under strict OpEx discipline. Any AI feature shipped to customers needed to demonstrate 75% gross margin. This was non-negotiable.
- **Cost that didn't add up.** At Alpha cost levels, the projected yearly cost per customer was $4,316 (small), $9,938 (medium), and $19,674 (large). At 75% margin, the implied price to customers would make the feature uncompetitive.

1.2 The Core Tension

The team had made an understandable architectural bet: deep learning for dimensionality reduction would capture richer structure than classical methods. The problem was that this bet optimized for *model expressiveness* at the expense of *operational economics*. In enterprise SaaS, a capability that works but can't be priced sustainably doesn't ship.

The question wasn't "can we make it work?" It already worked. The question was: *can we make it work at 5 billion profiles while keeping per-customer cost low enough to hit 75% margin?*

2 Diagnosis: Separating the Architecture Problem from the Model Problem

Before changing anything, I spent two weeks understanding the Alpha system end-to-end. This meant running the pipeline myself, reading every notebook, profiling every cost driver, and sitting with each engineer to understand their reasoning.

2.1 The Alpha Architecture

The pipeline had five stages:

1. **Data Input:** UPS profile exports (AAM attributes)
2. **Dimensionality Reduction:** Deep learning autoencoder on a single NC24 GPU
3. **Clustering:** TDA-inspired (Topological Data Analysis) method—mathematically elegant but computationally expensive and not horizontally scalable
4. **Ranking:** Probabilistic scheme to rank discovered clusters by quality
5. **Visualization:** Results displayed in a card view UI

2.2 Where the Cost Was

Using data from December 2019 through April 2020, the actual spend pattern was revealing:

Month	Databricks	BatchAI	Customers	Avg/Customer
December	$2,954	$1,937	1 (CBSi)	$4,891
January	$869	$1,077	1 (CBSi)	$1,946
February	$8,653	$1,103	2 (CBSi, Verizon)	$4,878
March	$13,256	$1,084	3 (CBSi, Verizon, Adobe.com)	$4,780
April (partial)	$9,377	$345	3	$3,240
Total				**$36,900**

Table 1: Alpha period actual costs (Azure subscriptions AZR2724 + AZR0203)

Average cost per customer per month: approximately $4,000. But this was on only 2 million sampled profiles. The actual production volumes for these customers ranged from 100 million to 5 billion stitched profiles. Extrapolating the Alpha cost structure to production volumes was untenable.

2.3 Root Cause: Two Architectural Bottlenecks

The cost wasn't distributed uniformly. Two components accounted for the majority:

1. **The deep learning model was sequential and GPU-bound.** Training on a single NC24 machine could not be parallelized across the Spark cluster. Every additional customer or data volume increase required more GPU hours at fixed throughput. The 10 DBUs consumed per NC24 training run were small in isolation but accumulated with the required 10 runs/month/customer.
2. **The TDA-inspired clustering was not horizontally scalable.** It produced beautiful topological structure but its computational complexity grew superlinearly with profile count. At 2M profiles it was feasible; at 100M it was not.

The Databricks cluster configuration told the story: 1 E8 driver + 2–20 E16 workers, with 40 DBUs average per pipeline run. The E16 fleet scaled linearly with data, but the NC24 GPU training was a fixed bottleneck that couldn't be distributed.

3 The Redesign: Distributed-First, Cost-Aware Architecture

With the diagnosis clear, the redesign focused on three principles:

1. **Every component must be distributed.** No single-machine bottlenecks. If a stage can't run on Spark, replace it.
2. **Replace model expressiveness with operational scalability.** A Spark K-Means model that processes 5B profiles is more valuable than a deep learning model that captures richer structure on 2M profiles.
3. **Build cost into the architecture, not around it.** The cost model should be a first-class design input, not a post-hoc constraint.

3.1 Architecture Changes: Alpha to Beta

Component	Alpha	Beta
Data Input	AAM PQL with attributes	UPS exports (AAM + AA unified)
Dim. Reduction	Deep learning autoencoder (NC24 GPU, sequential)	Random search trees (Spark-distributed)
Clustering	TDA-inspired (expensive, not scalable)	Spark K-Means + density-based methods
Discovery UX	Browse 30–40 clusters manually	Intent search interface with Bayesian cluster ranking
Audience Expansion	Manual selection only	Approximate Nearest Neighbor across all clusters
Ranking	Probabilistic scheme	Bayesian likelihood scoring given campaign keywords
Configuration	No user config	Configuration UI for customer tuning
Observability	None	Quality, throughput, and latency metrics at every pipeline stage

Table 2: Architecture changes from Alpha to Beta

The single most impactful decision was removing the deep learning autoencoder. It was the team's most sophisticated component—and the one that had to go. This required a direct, honest conversation with the engineers who had built it. I framed the decision not as a judgment on their work but as a change in the constraint set: *the model was good; the economics were wrong.*

3.2 Dual Data Source Integration

The Alpha only ingested Audience Manager (AAM) data—behavioral traits and audience segments. The Beta added Adobe Analytics (AA) event data through the Unified Profile Service, enabling clustering on actual user behavior (page views, conversions, journeys), not just declared audience attributes.

This was both a product improvement and a cost efficiency: richer input data meant the simpler models could achieve comparable or better segmentation quality, because the signal was in the data, not the model complexity.

3.3 Cost Model: Working with the Pricing Team

Rather than building the system and then asking finance to price it, I worked with the pricing team from the start to build a **driver-based cost model** that connected architecture decisions directly to margin targets.

The cost drivers were:

- Number of stitched profiles (determines data volume)
- Number of models per customer per run (AAM + AA = 2 parallel)
- Number of runs per month (exploratory + production)
- Processing time per run (determines DBU consumption)
- Cluster configuration (E8 driver + E16 workers)
- Storage (CosmosDB, shared with DSW, 2-week purge policy—negligible)

Cost Driver	Small	Medium	Large
Total profiles stitched	100M	1B	5B
Profiles processed per run	10M	30M	60M
AAM pipeline time	<2 hrs	<2 hrs	<3 hrs
AA pipeline time	<2 hrs	<2 hrs	<2 hrs
Parallel run time	<3 hrs	<3 hrs	<4 hrs
Cluster config	1 E8, 2–4 E16	1 E8, 2–12 E16	1 E8, 2–20 E16
DBUs per run	17	52	84
Runs/month/model	10	10	10
Adobe cost/month	**$80**	**$210**	**$520**
Adobe cost/year (both sources)	**$960**	**$2,520**	**$6,420**

Table 3: Beta projected cost model (driver-based)

Compare the Alpha projections: $4,316 / $9,938 / $19,674 per year at the small/medium/large tiers. The Beta cost model showed:

- **Small tier:** $4,316 → $960/year (**78% reduction**)
- **Medium tier:** $9,938 → $2,520/year (**75% reduction**)
- **Large tier:** $19,674 → $6,420/year (**67% reduction**)

At these cost levels, the pricing team could set customer-facing prices that comfortably exceeded the 75% margin threshold—making Segmentation AI a viable, sustainable enterprise feature rather than a margin-destroying demo.

3.4 Pipeline Observability

The Alpha had no systematic measurement of pipeline quality. In the Beta, we instrumented metrics at every stage—clustering quality scores, throughput (profiles/second), and end-to-end latency. This served two purposes: it gave us confidence that the cheaper Spark-based pipeline (K-Means clustering, random search trees for dimensionality reduction) was producing segments of comparable or better quality than the Alpha's deep learning approach, and it gave the pricing team hard numbers on resource consumption per customer tier.

The results were clear: the simpler models on richer data (AAM + AA combined) matched or exceeded the Alpha's segmentation quality, at a fraction of the compute cost.

4 The Product Insight: From Browsing Clusters to Finding Audiences

The architecture redesign solved the cost and scale problems. But working directly with customers revealed a more fundamental issue: *the product's interaction model was wrong.*

4.1 The Problem with "Browse and Select"

The Alpha workflow assumed a marketer would: (1) run the clustering pipeline, (2) browse the resulting 30–40 clusters, (3) visually inspect each cluster's attributes, (4) manually select profiles from multiple clusters, and (5) package them into an audience for Adobe Target campaigns.

In practice, this fell apart. A marketer at Verizon planning a Father's Day promotion for a new iPhone with a subscription bundle doesn't want to browse 40 clusters. They already know what they're looking for: fathers, likely in a certain age range, with signals of interest in mobile upgrades or Apple products. They need the system to *find the right clusters for them*—not present all clusters and hope they find the right ones.

This was a product design problem masquerading as a data science deliverable. The researchers on the team saw the clusters as the output. The customers saw the clusters as an intermediate step toward an audience they already had in mind.

4.2 The Search Interface: Letting Customers Describe Intent

I proposed adding a search layer to the product and brought this to the product team. Instead of browsing, customers would describe their campaign intent: "Father's Day iPhone promotion," "lapsed subscribers likely to re-engage," "high-value customers interested in home improvement." These keywords became the input to a ranking system that surfaced the most relevant clusters.

The researchers initially pushed back: clusters are unsupervised groupings, they argued—there is no natural ranking. I disagreed. The clusters have attributes. The customer has intent. We can compute a match.

4.3 Bayesian Cluster Ranking

I brought in a statistician and together we formulated the ranking as a Bayesian inference problem.

Each cluster C_k has a set of defining attributes: profile demographic attributes from AAM, behavioral event data from AA (including product click strings and category codes from Adobe's taxonomy services). The customer provides campaign keywords $\mathbf{q}$. We want to compute:

$$P(C_k \mid \mathbf{q}) \propto P(\mathbf{q} \mid C_k) \cdot P(C_k)$$

The likelihood $P(\mathbf{q} \mid C_k)$ measures how well the customer's campaign keywords match the attribute vocabulary of cluster k. We computed this using keyword overlap between the

query terms and the cluster's attribute strings—profile attributes, event descriptions, product interaction codes, and Adobe service category labels. The prior $P(C_k)$ could incorporate cluster size, recency, or engagement intensity.

The result was a ranked list of clusters, ordered by their posterior probability of containing the profiles the marketer was looking for. Instead of browsing 40 clusters, a Verizon marketer could type "Father's Day iPhone upgrade" and immediately see the 3–5 clusters most likely to contain their target audience, with an explanation of *why* each cluster ranked highly (which attributes matched).

This was a qualitative change in the product experience. Customers went from "I don't know where to start with 40 clusters" to "show me the ones that matter for this campaign."

4.4 Approximate Nearest Neighbor Expansion

Once a customer selected profiles from the ranked clusters, the next question was always: "are there more people like this in the other clusters?"

We implemented an Approximate Nearest Neighbor (ANN) search that, given the selected profile set, could propagate across *all* clusters to find similar profiles that the initial ranking might not have surfaced. This was critical because a Father's Day campaign audience might span profiles in a "mobile upgraders" cluster, a "suburban families" cluster, and a "high-spend electronics" cluster—the Bayesian ranking would surface the most obvious matches, but ANN expansion captured the long tail.

The workflow became: **search** (describe intent) $\rightarrow$ **rank** (Bayesian scoring surfaces top clusters) $\rightarrow$ **select** (customer picks profiles) $\rightarrow$ **expand** (ANN finds similar profiles across all clusters) $\rightarrow$ **package** (audience exported to Adobe Target for campaign execution).

This search-rank-select-expand pipeline became the feature that customers talked about. It transformed Segmentation AI from a clustering tool that data scientists appreciated into an audience discovery platform that marketers could use directly.

5 Execution: Managing the Team Through the Transition

5.1 The Human Side of the Architecture Change

Rearchitecting a system is a technical exercise. Rearchitecting a system that a leaderless team has been building is a leadership exercise. Three things mattered:

First, I took direct ownership of the team immediately. With no manager and a deadline, there was no time for a search or a placeholder. I managed the team myself, ran standups, did code reviews, and worked directly with each engineer on their component. This wasn't a temporary arrangement—I did this through the full Beta delivery.

Second, I was transparent about the constraint change. The Alpha prototype demonstrated real engineering skill. The deep learning autoencoder, the TDA-inspired clustering—these were sophisticated. Telling the team we needed to replace them required honesty: the work was good, the business economics were the binding constraint, and the architecture needed to change to meet them.

Third, I preserved the team's investment where possible. The preprocessing pipeline, the ranking logic, the visualization layer—these survived into Beta. The changes were surgical: replace the two components that couldn't scale, add the new data source integration, and build the cost model. The team saw continuity, not a rewrite.

5.2 Customer Engagement

I worked directly with the customer teams at Verizon, Home Depot, and CBSi. The conversations were different at each:

- **CBSi** had been in Alpha longest and had the most feedback on segmentation quality. Their input directly influenced the ranking improvements in Beta.
- **Verizon Wireless** had massive data volumes (billions of profiles) and was the primary stress test for the distributed architecture. Making Verizon happy meant the system could handle anything.
- **Home Depot** had specific use cases around in-store vs. online customer behavior that validated the dual-source (AAM + AA) architecture.

All three customers were actively using the Beta in production—not just running pilots.

6 Results

6.1 Technical Outcomes

- **Scale:** 2M profiles (Alpha) $\rightarrow$ 5B stitched profiles (Beta), a 2,500$\times$ increase in capacity
- **Processing time:** 8+ hours (Alpha, sequential) $\rightarrow$ <4 hours (Beta, fully parallel) for the largest tier
- **Dual data sources:** AAM behavioral attributes + AA event data, processed in parallel through UPS
- **Dimensionality reduction:** Random search trees on Spark replaced GPU-bound deep learning autoencoder
- **Cost per customer:** 67–78% reduction across all tiers
- **Margin target:** 75% gross margin achieved at all three tiers
- **Pipeline observability:** Quality, throughput, and latency metrics instrumented at every stage

6.2 Product Outcomes

- **Intent-driven discovery:** Search interface replaced manual cluster browsing—customers found relevant audiences in seconds instead of hours
- **Bayesian cluster ranking:** Surfaced the 3–5 most relevant clusters from 30–40, with attribute-level explanations of why each cluster matched
- **ANN audience expansion:** After initial selection, Approximate Nearest Neighbor search found similar profiles across all clusters, capturing the long tail

- **End-to-end workflow:** Search → Rank → Select → Expand → Package for Adobe Target campaigns

6.3 Business Outcomes

- **Three enterprise customers in production:** Verizon Wireless, Home Depot, CBSi—all actively using the system, not running pilots
- **Pricing team sign-off:** Feature approved for general availability pricing
- **Platform integration:** Segmentation AI became a first-class capability in Adobe Experience Platform's intelligence layer
- **Customer satisfaction:** The search-rank-expand workflow was the feature customers consistently highlighted as the reason Segmentation AI was useful to their marketing teams

7 Lessons Learned

7.1 Your Customers Already Know What They Want—Build for That

The most impactful change in Segmentation AI was not the architecture redesign. It was recognizing that marketers don't want to explore clusters—they want to find audiences for campaigns they've already planned. The Alpha treated cluster browsing as the product. The Beta treated campaign intent as the input and audience delivery as the output.

This required challenging the team's framing. The researchers saw "clusters cannot be ranked" as a technical fact. But it was an assumption about the product, not about the math. Given customer intent keywords and cluster attributes, Bayesian inference gives you a perfectly natural ranking. The lesson: when domain experts say something is impossible, check whether they're making a statement about their model or about the problem.

7.2 Cost Is an Architecture Decision, Not a Finance Conversation

The most common pattern I see in enterprise AI teams is: build the system, then ask the pricing team to make the numbers work. This always fails. If the architecture's cost structure doesn't support the margin target, no amount of pricing creativity fixes it.

In this case, building the driver-based cost model *before* finalizing the Beta architecture meant every technical decision had an immediate cost implication visible to the team. "Should we use 4 or 12 E16 workers?" was simultaneously a performance question and a margin question.

7.3 Model Sophistication Is Not Product Value

The deep learning autoencoder was the most technically impressive component in the Alpha. It was also the component that had to be removed. In applied AI, the mapping from model sophistication to product value is not monotonic—a simpler model that runs on distributed infrastructure at production scale delivers more value than a complex model that works on 0.1% of the data.

This is not an argument against deep learning. It is an argument for matching model complexity to operational constraints. At 2019-era compute costs, a Spark-native pipeline was the right

answer for this product, at this scale, at this margin target.

7.4 Stepping In as a Manager Is Sometimes the Right Move

When I joined, the conventional path would have been to immediately hire or transfer a manager for the team and focus on "strategic" work. But the team had a delivery deadline, no manager, and a prototype that needed fundamental rearchitecting. Delegating management to someone who didn't yet understand the system would have added latency the project couldn't afford.

Direct management also gave me something irreplaceable: firsthand understanding of every component, every engineer's strengths, and every cost driver. That understanding is what made the redesign possible in the timeframe we had.

About the Author

Anjan Goswami, Ph.D., is the founder of SmartInfer, an AI research and product company. Previously, he served as Head of AI & Data at Microsoft (PowerPoint Copilot) and Head of AI & Data Science at Adobe (Experience Platform & Document Cloud). He has 20+ years of experience building ML systems at scale across Amazon, eBay, Walmart, Salesforce, Adobe, and Microsoft, with 10+ patents and publications at NeurIPS and IEEE.