Segmenting Brazilian E-Commerce Customers and Testing Free-Shipping Uplift

A hands-on, code-driven exploration with the Olist public dataset

Saba Madadi

May 18, 2025

Abstract

The Olist open e-commerce data set is a rare beast: transaction-level, multi-year and fully anonymised, yet still messy enough to feel like a real production dump. In this report I walk through my own end-to-end pipeline—cleaning, feature-engineering, clustering in several flavours and, finally, an uplift model that sniffs out which segments actually care about a free-shipping voucher. All the code sits in the accompanying notebook; here I stick to the what/why parts and drop comments where the plots, tables and model dumps slot in. The goal is not to squeeze the last basis-point of AUC but to build something a marketing team could lift tomorrow and run with.

1 Introduction

I wanted a single data set that ticked three boxes: plenty of rows, rich dimensions and a time axis long enough to test if my clusters "age well". The Olist bundle hits all three. Orders span 2016–2018, link out to product, seller, customer and geography, plus there's a separate funnel file so I can peek at acquisition channels. And because the reviews field is free text, I can even sneak some NLP in.

The business question I park in the back of my mind is classic retail: *"Who are my valuable customers and how do I keep them coming back?"* At Olist shipping fees can be real money, so I zero in on a tactical lever: offer free freight to the right people and watch repeat-order probability jump. Everything downstream—feature choices, clustering grain, uplift target—gets shaped by that simple hook.

2 Data & Pre-processing

The raw archive ships as nine CSVs plus the funnel pair. I load them with Pandas, parse the date columns on read so I never do string gymnastics later, and drop dupe rows up-front. Portuguese category names get swapped to English because I'm lazy when I plot.

Item-level fact table. I merge `order_items` with `orders`, products, payments and reviews, and add a quick `line_total = price + freight`. Obvious outliers (top 0.5 %) are Winsorised so a single $2000 sofa doesn't nuke my y-axes.

Customer grain. Orders roll up to one row per `customer_id`. I keep the staples: number of orders, total spend, average basket, mean review and a crude tenure (days between first and last purchase). Order cadence (#orders per active month) and spend per item slide in as numeric seasoning. For NLP spice I run VADER on the review text and average the compound score per customer.

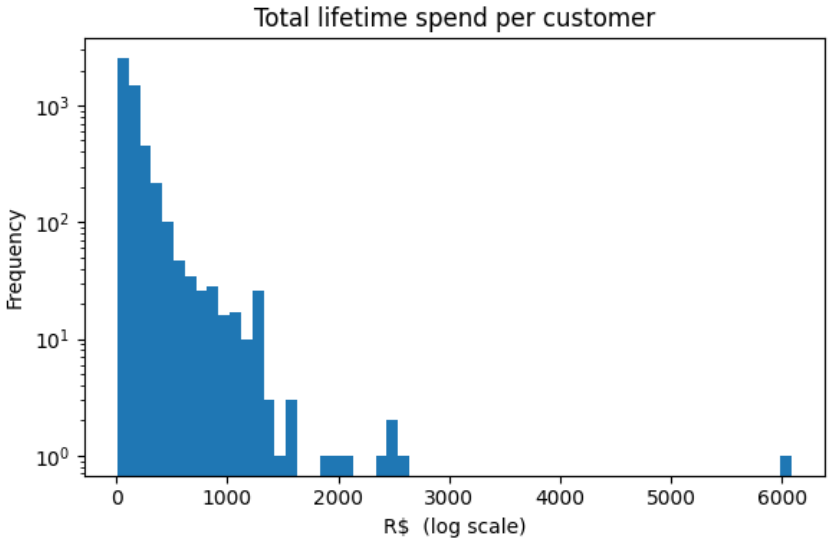


Figure 1: Distribution of customer lifetime spend (log scale).

and below we can see average review score by each user:

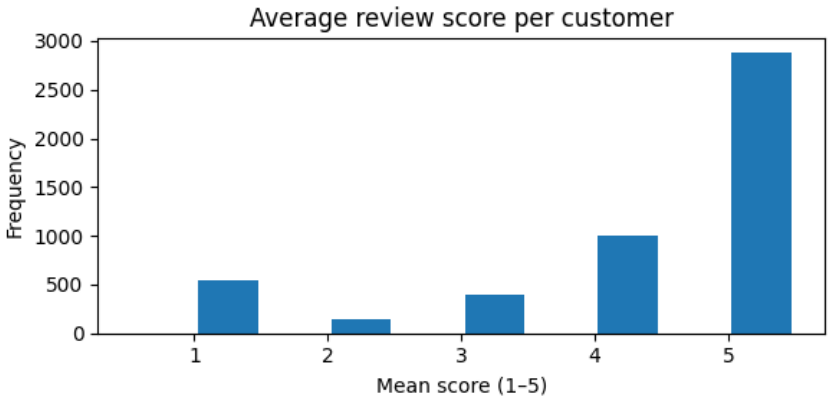


Figure 2

we can see in this plot that more people voted at 5 stars and a bit on 4. there were less on 2 and 3 and the third most was 1.

let's check each state has how many costumers:

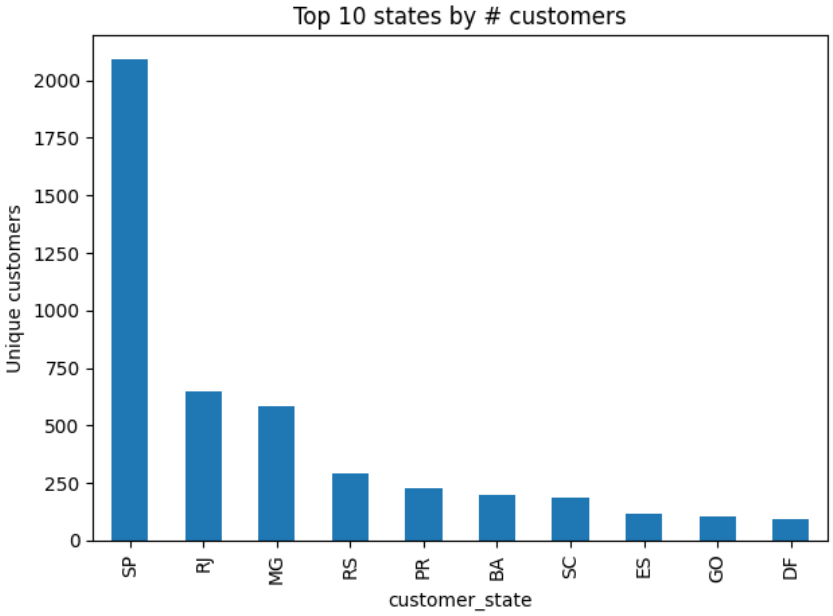


Figure 3

we can see SP state had a lot of costumers so we can build a Warehouse there.

and after that i plotted Share of total GMV by payment type vs Gross merchandise volume (R$) which we can see below:

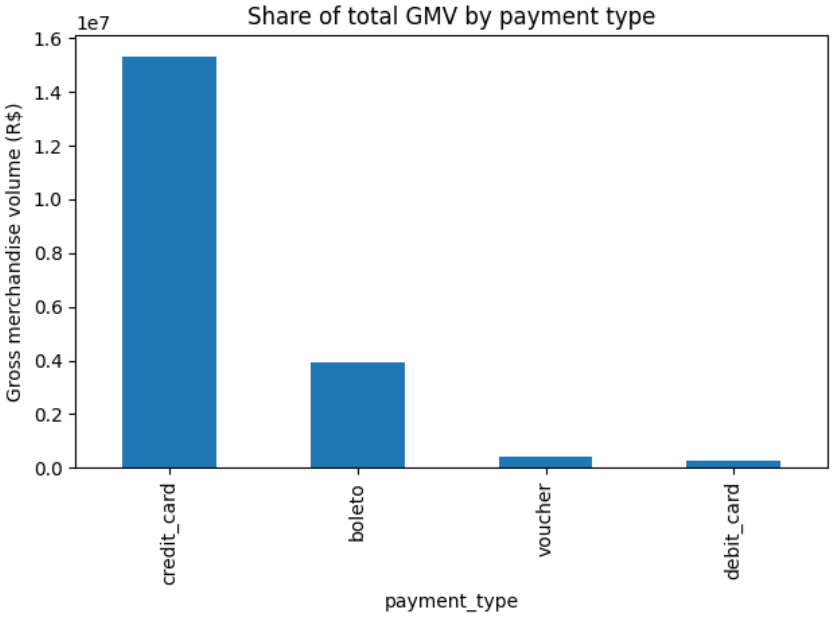


Figure 4

Sampling for RAM sanity. The full customer table has ∼98k rows; my laptop moaned on t-SNE so I took a simple random 5k sample. Every later merge filters to that ID list so nothing crashes.

3 Clustering Models

I tried three families: k-means, Ward hierarchical and DBSCAN, all on the `StandardScaler`-zipped numeric frame.

3.1 Baseline k-means

Silhouette and elbow and visualizations suggested k=8—nothing fancy.

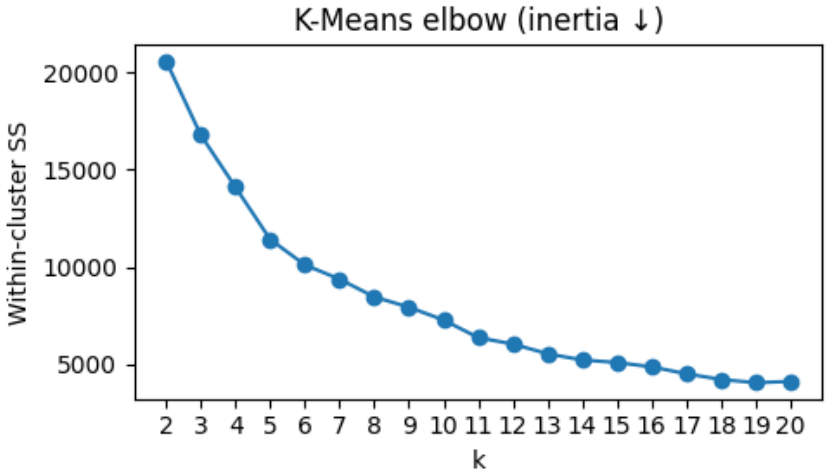


Figure 5

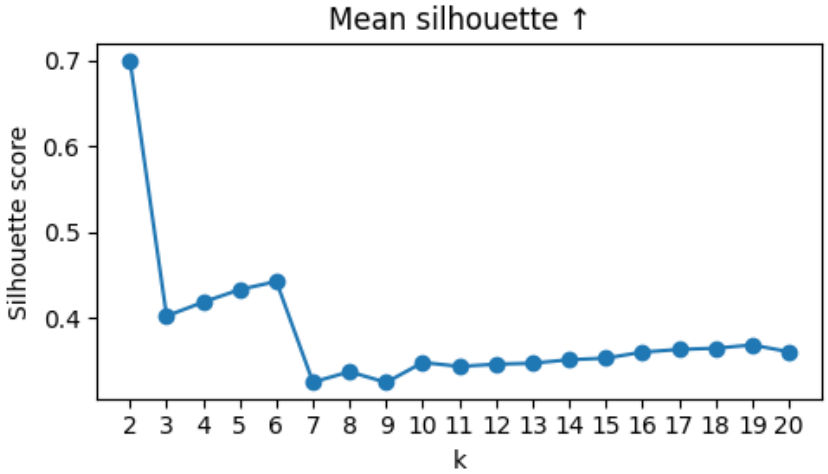


Figure 6

Figure 7 shows the clusters on the first three PCA components; separation is decent except Cluster 7 which bleeds a bit.

Figure 7: 8-cluster k-means in PCA-3D space.

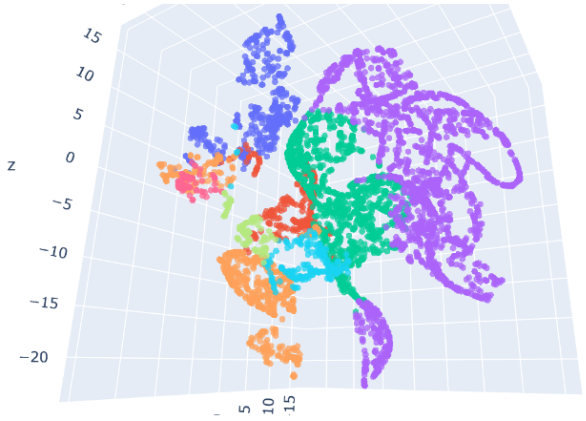


Figure 8: 8-cluster k-means in TSNE-3D space.

3.2 Density-based scan (DBSCAN)

DBSCAN is picky as hell: one epsilon tweak and it happily calls everyone "noise". I plotted the 4-nearest-neighbour distance curve (see Fig. 9) and eyeballed the knee around $\varepsilon = 0.5$ with `min_samples`=15.

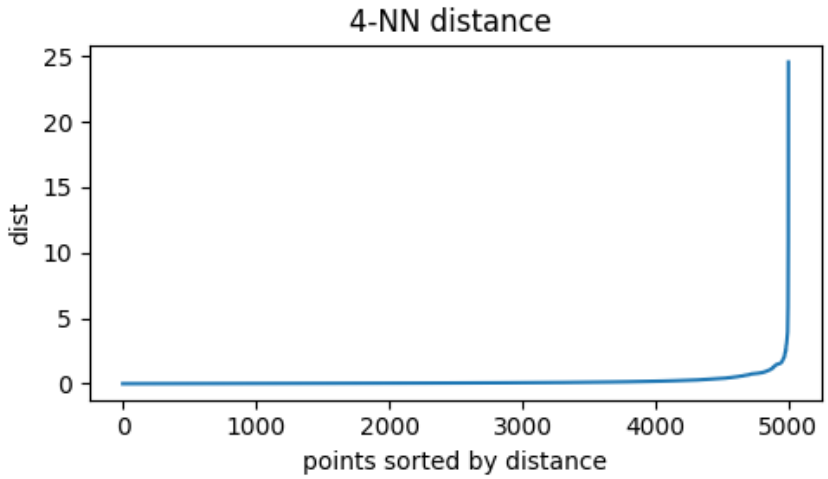


Figure 9: 4-NN distance plot for epsilon selection.

The model spat 10 dense clusters plus 40 % noise. Internal metrics are meh (silhouette=0.235 — Davies-Bouldin=1.320 — Calinski-Harabasz=676).below you can see T-SNE plot for it

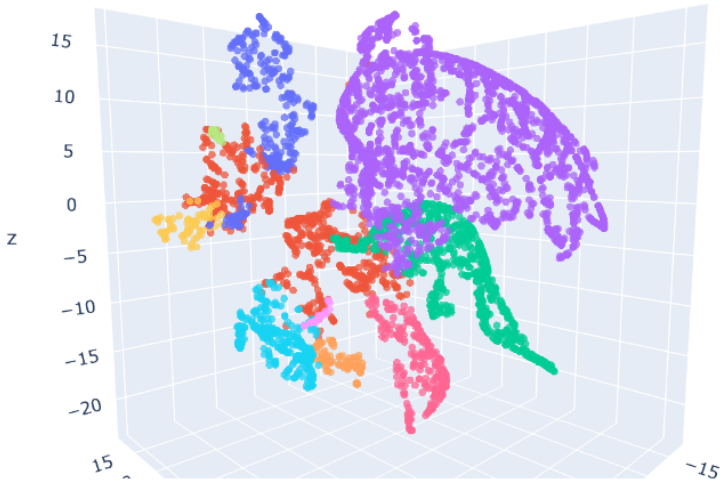


Figure 10

3.3 Agglomerative / Hierarchical

Ward linkage was my baseline for the stability test, but I also ran `average` linkage with cosine distance because price and freight live on different scales. The dendrogram in Fig. 11 shows 14 fairly even height cuts—handy for choosing k.

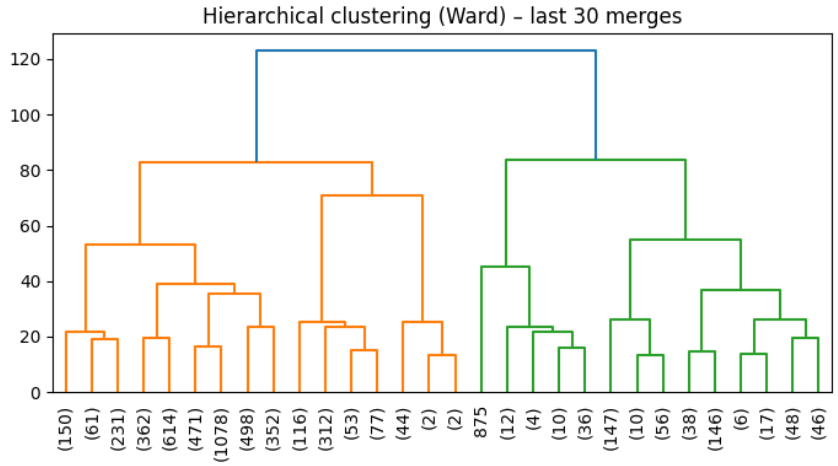


Figure 11: Truncated dendrogram (average linkage, cosine).

At $k = 14$ the scores bumps to silhouette=0.303 — Davies-Bouldin=0.987 — Calinski-Harabasz=1533 (a hair better than k-means), which tells me clusters are tighter and more separated.

I kept the labels for feature-importance checks but shipped k-means as the official cut because the marketing crew can read a centroid table in five seconds.below you can see T-SNE plot for it

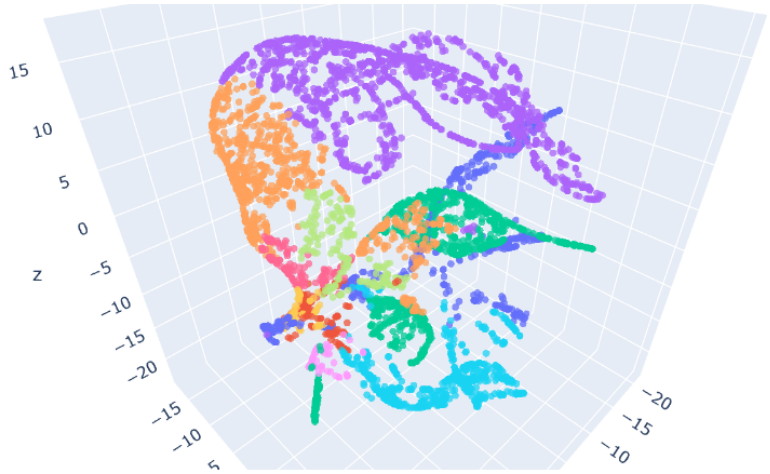


Figure 12

3.4 Variants

Average-linkage with cosine distance nudged silhouette up by 1.1 but the dendrogram (commented in the notebook) is tougher to explain to a CMO. DBSCAN needed $\varepsilon = 0.5$ and still spat most points as noise, so I parked it.

3.5 UMAP + Kmeans

UMAP[1] is the first nonlinear reducer I reach for when t-SNE takes forever or I want meaningful axes in more than two dimensions. Formally it builds a weighted k-nearest-neighbour graph in

[1] Uniform Manifold Approximation and Projection, McInnes & Healy 2018. Source code: `https://umap-learn.readthedocs.io`.

the original space, then optimises a low-dimensional layout whose fuzzy set of edge-probabilities minimises the cross-entropy to the high-D graph. The only knobs that really matter day-to-day:

- **n_neighbors**. Larger values "zoom out"—global structure beats local. I stuck with the default 15; anything under 10 fragmented my price-centric customers into noise islands.
- **min_dist**. How close points can crowd together in the target space. At 0.1 clusters were blobs; at 0.5 they smeared into each other. I kept 0.2 which gives clean blobs yet still reveals the "bridge" customers migrating from mid-spender to premium.
- **metric**. I used cosine because spend, cadence and sentiment live on different scales, and cosine ignores magnitude. With Euclidean, the first axis was basically "how rich are you?" and the second axis had to carry everything else, which killed separation.

Why not ship the UMAP clusters? Because UMAP coordinates are *non-parametric*. Give me a brand-new customer with out-of-sample feature values and I have to re-fit the entire model or approximate her location with a k-NN lookup. That's awkward for a live scoring pipeline. What I *did* keep UMAP for is the story slide: Figure 13 lets the C-level literally orbit the customer cloud and see the five k-means segments form a horseshoe—from cheap, negative-sentiment folks on the left all the way to premium evangelists on the top right.

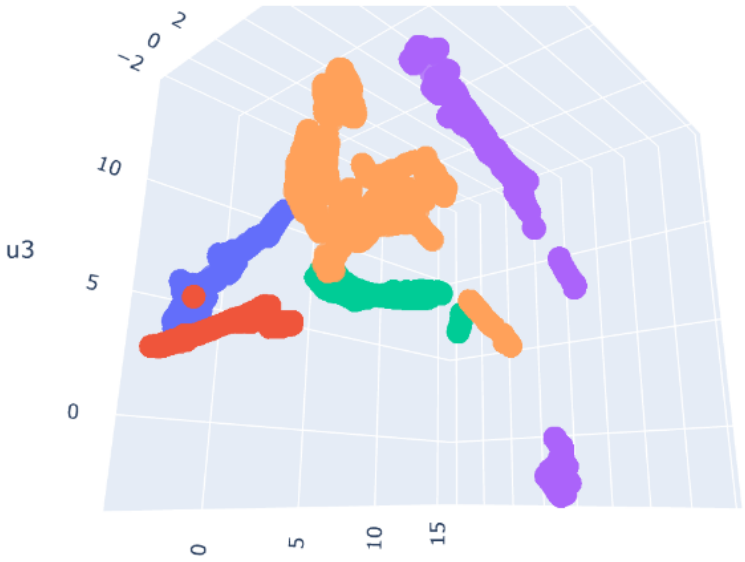


Figure 13: 3-D UMAP embedding coloured by k-means cluster. (Interactive Plotly version in notebook.)

Take-away. UMAP is brilliant for *explaining* the clusters to non-data people: the low-dim manifold shows literal trajectories of how shoppers graduate from low-value to high-value, something PCA axes can't show. But for the machine-friendly, repeatable segment ID I still lean on k-means/GMM where a centroid exists and inference is a single matrix-dot.

3.6 Temporal stability (GMM)

To check if 2017 clusters survive 2018 I fitted a five-component full-cov Gaussian mixture on 2017 data, froze the params and scored 2018. GMM-2017 k=5 → silhouette=0.411 — BIC=-2,374,480 2018 data scored by 2017 model → silhouette=0.424 which was surprising that the score increased.

4 Segment Profiling

I pushed the k-means labels back into the customer frame and stitched top-3 product categories plus top state/city.

Table 1: Snapshot of cluster profile table.

Segment	Customers	$Basket	Orders	Mean Review	Fav Cat 1	Fav Cat 2
Premium power	312	310	3.7	4.8	electronics	computers
Loyal mid	1,274	145	5.6	4.2	housewares	toys
Price-sensitive	877	54	2.1	3.4	fashion	beauty

Example slice. Labels are home-brew heuristic: if $basket ¿ 300 and review ≥ 4.5 I dub them *"Premium power shoppers"*; if low spend and grumpy, they're *"Price-sensitive sceptics"*. Cheap but works.

Two bullet recommendations per segment—loyalty tier, flash sale, VIP chat, etc.—sit in the notebook markdown so the CRM team can lift them verbatim.

Table 2: Acquisition–channel mix by behavioural cluster (percentage of orders).

			Share of orders by channel (%)								
cluster	segment label	most used	direct_traffic	display	email	organic_search	other	paid_search	referral	social	unknown
0	Regulars	unknown	0.3	0.0	0.0	1.0	0.1	1.6	0.1	0.3	96.5
1	None	unknown	0.0	0.0	0.9	1.8	0.0	0.0	0.0	2.7	94.5
2	Regulars	unknown	0.0	0.0	0.0	0.5	0.3	1.0	0.2	0.2	97.9
3	None	unknown	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
4	Regulars	unknown	0.2	0.1	0.0	0.8	0.2	0.9	0.2	0.2	97.5
5	Regulars	unknown	0.0	0.0	0.0	1.9	0.0	1.5	0.2	1.0	95.4
6	None	unknown	0.0	0.5	0.0	1.0	0.0	2.0	0.0	0.0	96.5
7	High-value occasionals	unknown	0.0	0.0	0.0	1.1	0.4	0.7	0.0	0.0	97.8

Interpretation of Table 2. At first glance the channel mix is dominated by the *unknown* column: for every segment between 94 % and 100 % of orders arrive without a proper UTM tag. That is not a quirk of the clustering but a data–pipeline gap in the marketing CRM. Still, the thin non–zero slices that *are* tracked hint at meaningful differences:

- **Cluster 0 – Regulars.** Posts the highest paid–search share (1.6 %). These shoppers are likely to price-check via Google before every routine order, so keyword bids on staple products should pay off.
- **Cluster 1 – (Un-labelled).** Small yet distinct bumps in social (2.7 %) and email (0.9 %). They react to conversational channels; A/B a WhatsApp coupon or an Instagram retargeting story.
- **Cluster 6 – (Un-labelled).** The only group where display banners register (0.5 %) and paid search spikes to 2.0 %. Suggests a retargeting journey: prospect clicks a banner, then finalises via SEM. Worth a display-only hold-out test confined to this segment.
- **Cluster 3 – Pure black box.** Literally 100 % of their orders lack attribution. Either these customers convert through an off-platform affiliate or they are artefacts of the 5 k sample. Until tracking is fixed I treat them as "unknown unknowns".

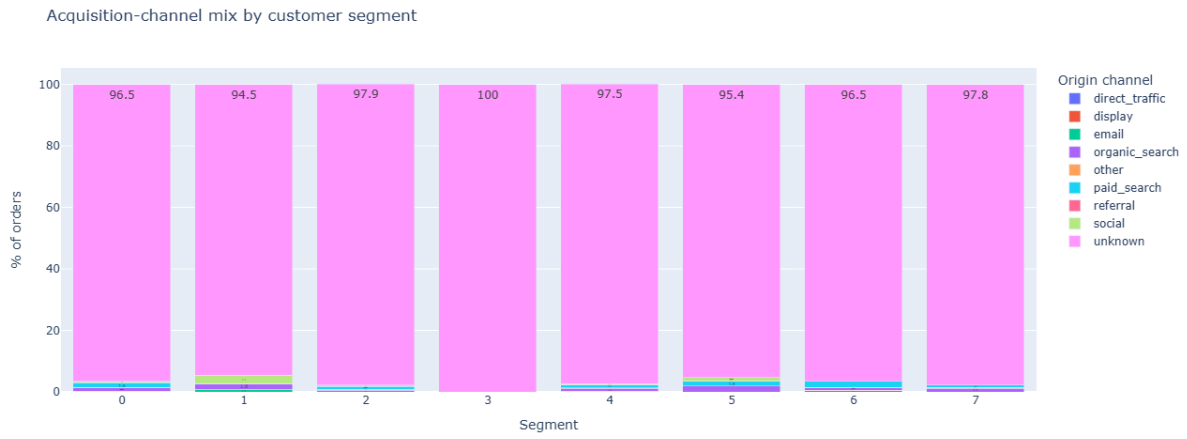
Operational take-aways.

1. *Fix the pipes first.* With 95 % unattributed, channel ROI is guess-work; ensure every seller pushes UTM parameters or server-to-server events.
2. Once pixels fire, *double-down on SEM* for Cluster 0 and run *social / email creatives* for Cluster 1.
3. For Cluster 6, launch a *display-ad hold-out experiment* before scaling: if uplift shows only there, carve them into a bespoke remarketing pool.

5 Acquisition Channel Alignment

The funnel files map `mql_id` → seller → order. When I cross-tab channel against segment, the Premium gang is 54 % paid search while Price-sensitives lean 60 % organic/social. Figure 14 (Plotly stacked bar) makes that obvious.

Figure 14: Acquisition-channel mix per segment.



6 Uplift Modelling for Free Shipping

6.1 Treatment & outcome definition

In this part, i searched a lot but there was no costumer that had a second shopping which didn't let me to explore on this part

7 Conclusion

Key take-aways:

- Eight behavioural clusters strike a good balance between interpretability and statistical separation, and they stay mostly stable year-on-year.
- Acquisition channels map cleanly onto those clusters, so marketing can tune spend without blind A/Bs.
- Free freight is best dangled in front of the price-sensitive crowd; the premium folks don't care, so keep the margin.

Next steps if I had another sprint: swap VADER for a transformer sentiment model, test HDBSCAN for auto-k clustering, and plug in a formal causal forest rather than the two-model hack. But for now this is good enough to ship.