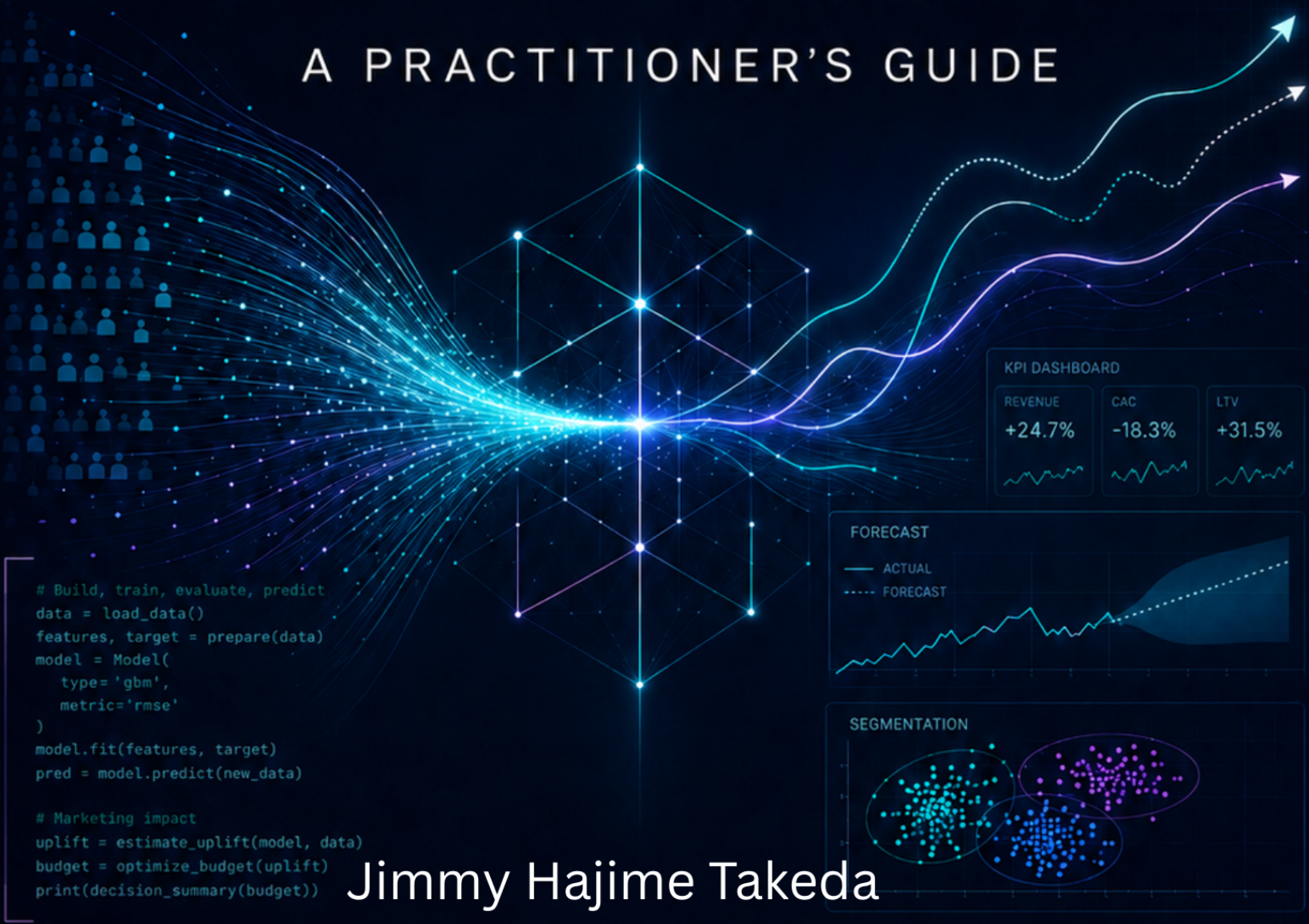
MARKETING DATA SCIENCE in PYTHON

A PRACTITIONER'S GUIDE



Jimmy Hajime Takeda

Table of contents

Preface

Draft version
Last updated: May 2026

Last month's campaign drove a spike in sales, and the team is celebrating. But before we take the credit, it's worth asking: was it really the campaign that made the difference?

Temperatures rose at the same time. A competitor ran out of stock. An influencer posted about our product on social media. Maybe one of these factors mattered more than we think, or maybe it was a mix of all of them. And even if sales went up, did the campaign actually bring in valuable new customers?

We handed out coupons and saw a bump in sales, but were we just pulling forward purchases that would have happened anyway? We spent millions on TV and CTV ads, but can we really call that a success?

Even in 2026, answering these questions directly is still hard. AI has made the creative side of marketing—generating ad copy, producing banners, building landing pages—much faster. But when it comes to measurement and decision making, questions like "Did it work?" and "What should we do next?" are still tough. This is where human judgment matters most.

This book is a practical guide for working on that frontier. I'll show you how to use tools like causal inference, Marketing Mix Modeling (MMM), pricing analysis, and Customer Lifetime Value (CLV) analysis to make marketing decisions more reliable, even when the data is messy. We'll cover both the theory and the code, always with an eye on what actually helps in practice.

Who This Book Is For

This book is mainly for data scientists, analysts, and engineers who know the basics—statistics, Python, some machine learning—but haven't yet applied those skills to real marketing problems.

Each chapter comes with Python code and notebooks. The idea is for you to get hands-on as you go, so you can apply these techniques to your own data right away.

I also wrote this for technically-minded business professionals—marketers, product managers, and executives. You can skip the formulas and code if you want. If you focus on the problem framing at the start of each chapter, you'll still get the key analytical thinking and decision-making frameworks.

Structure of This Book

Part	Theme	Contents
1	The Big Picture of Marketing Science	Definition, how it differs from general DS, industry characteristics
2	Analysis for Better Decisions	Analysis design, business questions, KPI trees, storytelling (SCR)
3	Causal Inference	A/B testing, quasi-experiments, causal machine learning, Uplift Modeling
4	Customer Analytics	Segmentation, CLV
5	Commercial Analytics	Price elasticity, assortment optimization, demand forecasting
6	Media Investment and Optimization	Attribution, MMM, budget optimization

About the Author

Jimmy Hajime Takeda is a marketing scientist and co-founder of Kuwalyst, where he works on practical ways to connect marketing measurement with business decisions. He is especially interested in MMM, incrementality, customer analytics, and how teams can make better decisions from imperfect data. If you are working on similar problems, he is always happy to exchange ideas. Connect with him on LinkedIn.

Part 1. Introduction

Photo by Carlos Muza on Unsplash

Marketing Science sits at the intersection of statistics, machine learning, and business judgment — and the way it plays out in practice looks very different from general data science. The definitions matter, the industry context matters, and the quirks of marketing data matter, because they all shape how you approach real business problems.

After this part, you will be able to:

- Define Marketing Science and explain how it differs from general Data Science
- Recognize how data availability and experimentation ease vary by industry
- Read marketing data through the lens of the funnel, and know which datasets and KPIs sit where
- Understand the three characteristics of marketing data and factor them into your analytical approach

1.1 What Is Marketing Science?

Context

In marketing, the gap between data science work and business value tends to show up in patterns like these:

- A coupon campaign is declared a success: recipients spent 20% more than non-recipients. Both the business and the data scientist agree the campaign worked — until it turns out the coupons were sent to customers who were already most likely to buy. The "lift" was selection, not effect.
- A data scientist delivers a fresh customer segmentation. Stakeholders nod, call it interesting, and keep using the segmentation they already had. The work never reaches a campaign.
- A demand forecast hits 2% MAPE at the brand level. But the business plans inventory at the SKU-by-region level, where the forecast is much noisier — leading to stockouts in some regions and overstock in others.

All three are different versions of the same problem. In the first, the analysis gave a clear number but missed the real cause. In the second, the model never reached the people who could use it. In the third, the analysis was done at a different level of detail than the business decision it was meant to support.

Definition

That gap is the starting point for the definition we use in this book:

Marketing Science is the applied discipline of using data, statistics, and machine learning to improve business decision-making in marketing.

The work is not done when the model is built. It is done when the analysis actually changes a decision, leads to a new action, and moves a business result.

In general data science, success is often measured by prediction accuracy. Marketing Science still cares about accuracy, but accuracy is a means to an end: the real measure of success is the business result the analysis enables.

Workflow

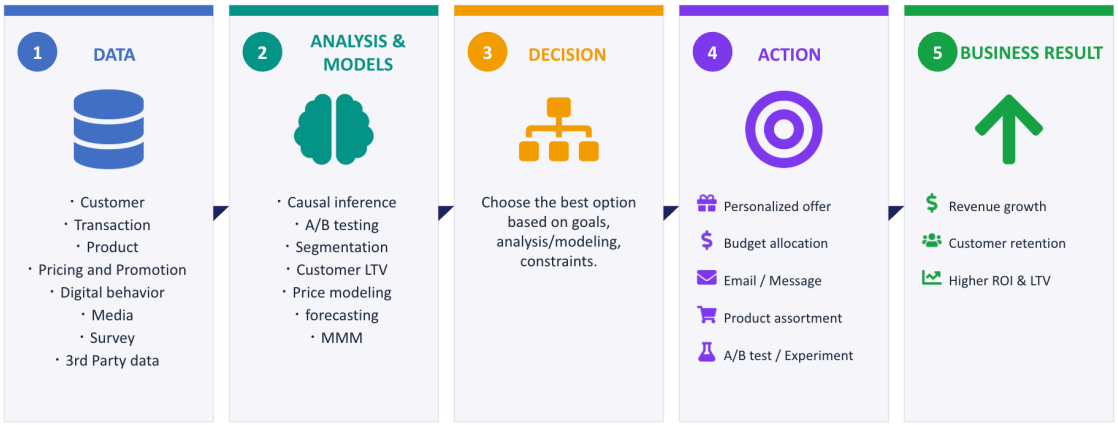


Figure 1: Five-stage marketing science pipeline: data → analysis → decision → action → result.

Figure 1 shows the central idea of this book: analysis creates value only when it changes a decision, an action, and ultimately a business result.

How Marketing Science Differs by Industry

The definition applies across industries, but *how* you do the work — what data you have, which methods you use, how easy it is to experiment — varies a lot.

Figure 2: Five industry contexts covered in the book: CPG, retail, e-commerce, consumer app, and SaaS.

Industry	Data Examples	Data Characteristics	Key Methods and Themes	Ease of Experimentation
CPG (Consumer Packaged Goods)	POS (via retailers), panel data (NielsenIQ, Circana), media spend data, retail media reports	Companies often do not own purchase data directly (obtained via retailers or panel providers). Mostly aggregate-level	MMM, price elasticity analysis, planogram optimization, trade promotion analysis	Low (hard to control because distribution is intermediated)
Retail (Offline)	POS, loyalty program data (Kroger Plus, Target Circle, etc.), promotion and planogram data	Owns individual-level purchase history. Can analyze at SKU x store x day granularity. The rise of retail media networks is also expanding data use for advertisers	Promotion optimization, assortment optimization, customer segmentation, retail media ROI analysis	Medium (store-level and region-level experiments are possible)
E-commerce	Clicks, page views, cart additions, purchases, media spend data	Rich behavioral logs. Full-funnel tracking is possible end to end	A/B testing, funnel analysis, recommendations, CLV prediction	High (randomization is easy online)

Industry	Data Examples	Data Characteristics	Key Methods and Themes	Ease of Experimentation
Consumer App	Downloads, in-app behavior, purchases, media spend data	Retention and engagement are the key metrics. Freemium models are common	Retention analysis, ROAS optimization, cohort analysis	High (in-app A/B testing is easy)
SaaS	Product usage logs, contract and billing data, support tickets	Detailed usage logs are readily available. Easy to link with contract and billing data	Churn prediction, PLG analysis, product analytics	High (feature-level A/B testing is routine)

You do not need to memorize the table. What matters is seeing that the same marketing question can call for very different methods, depending on who owns the data, how granular it is, and how easily the business can test ideas.

For example, MMM is a well-established method in CPG with decades of history, but it is rarely used in SaaS. Conversely, A/B testing is an everyday decision-making tool in app and e-commerce businesses, but running a rigorous A/B test in CPG is difficult.

In the U.S. market, the rapid growth of retail media networks (Amazon Ads, Walmart Connect, Instacart Ads, etc.) is blurring industry boundaries. CPG brands can now access near-user-level data through retail media. At the same time, this has created a new challenge for causal inference: "Is the retail media effect incremental, or are we just serving ads to customers who would have bought organically?"

This book focuses on concepts that work across industries, while touching on industry-specific considerations where relevant.

1.2 The Data Landscape of Marketing

Marketing data is never just one table. It is a stack of sources, each capturing a different part of the customer journey. In this section, I will map that stack using three lenses: the funnel, the KPIs at each stage, and the datasets behind them.

The Funnel as a Measurability Map

Customers rarely buy on first exposure. They move — sometimes over months, sometimes minutes — from "not knowing the brand exists" to "buying" to "buying again." The most common model of this journey is the funnel:

Awareness → Consideration → Conversion → Retention

The funnel is not useful because customers move through it in order. It is useful because it acts as a measurability map. As you move down the funnel, data gets more granular, measurement is more direct, signals come in faster, and experiments are easier to run.

Figure 1: The marketing funnel as a measurability map. As customers move down the stages — Awareness, Consideration, Conversion, Retention — data becomes more granular, measurement more direct, signals faster, and experiments easier to run.

This is why method choice depends on the business model. E-commerce and SaaS teams can often test behavior directly. CPG and brand advertisers have to infer upstream effects from signals that are noisier, slower, and more aggregated.

A common trap is the streetlight effect: focusing only on what is easy to measure. Performance channels can report ROAS every day, but brand campaigns cannot. The first step is to ask which part of the funnel a KPI, dataset, or method actually covers.

KPIs across the Funnel

Each funnel stage has its own set of KPIs. You do not need to memorize them, but you should be able to tell which stage a metric belongs to. That way, you can spot when a dashboard is mixing up different layers.

Funnel Stage	Typical KPIs	What They Measure
Awareness	Reach, Impressions, Aided / Unaided Brand Awareness, Share of Voice	Whether people know you exist
Consideration	CTR, Site Visits, Time on Site, Search Volume, Branded Search Lift	Whether people are evaluating you
Conversion	CVR, CAC, AOV, ROAS	Whether people buy, and how efficiently
Retention	Repeat Rate, Churn Rate, Retention Curve, LTV / CLV, NPS	Whether people come back

Typical Datasets in Marketing

The Data box in Figure 1 lists the main dataset categories that drive everything downstream. Here is what those categories look like in practice, and the pitfalls to watch for as a data scientist.

Category	Examples	What a DS Newcomer Should Know
Customer / User	CRM, loyalty data, demographics, account / contract data	Schemas drift; definitions of "active user" change with org restructures. Identity resolution across IDs is harder than it looks.
Transaction	Purchase history, subscription / billing, cart, returns	Owned by EC and SaaS; only partially seen in CPG (POS via retailers). Granularity (line item vs order vs daily) matters for every downstream model.
Product / Pricing	SKU master, price lists, inventory, planograms, promotion calendars	SKU hierarchies are messy. Prices move with promotions, competitor moves, and seasons — a "price" column is rarely as stable as it looks.
Digital behavior	Clicks, page views, sessions, in-app events, email opens / clicks	Volume is huge, but explains only a small fraction of actual purchase variance. Easy to over-trust.
Media	Spend, impressions, GRP, creative metadata, campaign tags, email sends, push logs	Every platform reports differently. Attribution windows, "conversion" definitions, and viewability standards diverge. Joining across platforms is a recurring tax.
Survey	Brand tracking, NPS / CSAT, ad recall, intent surveys	Pre-aggregated and sample-based. Useful for awareness-stage signals you cannot get from logs. Watch for sampling and response bias.

Category	Examples	What a DS Newcomer Should Know
Third-party	Panel data (NielsenIQ, Circana), syndicated category reports, search trends, weather, economic indicators, competitor data	Pre-aggregated and often delivered with a lag. Use for context, not for fine-grained causal claims.
Voice of customer	Reviews, support tickets, social mentions	Unstructured. Strong selection bias (only certain customers post). Useful for hypothesis generation, dangerous as a representative signal.

1.3 Three Characteristics of Marketing Data

The hard part of Marketing Science is not the complexity of the methods. It is the nature of the data itself.

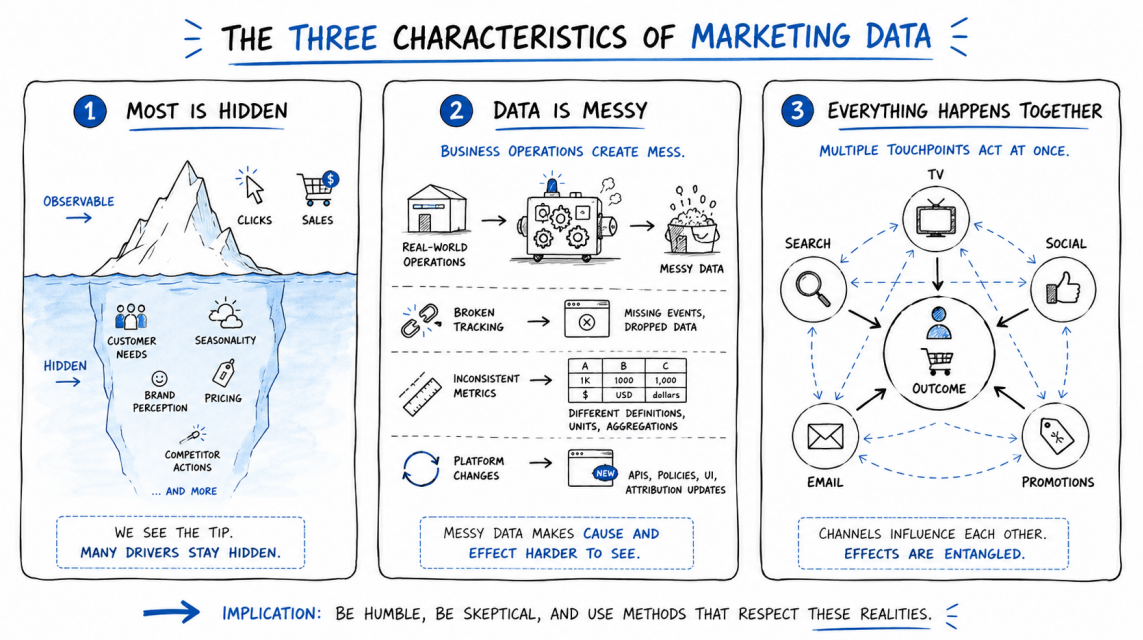


Figure 1: Three characteristics of marketing data: tip-of-the-iceberg, operational byproduct, multicollinearity.

Characteristic 1: What's measurable isn't all that's causal

Before a customer makes a purchase, they are influenced by countless touchpoints — your ads and website, yes, but also friends, competitors, the economy, the weather, even their mood. Most of this is invisible to your data.

What gets recorded is usually the easy-to-measure stuff: clicks, impressions, page views. The real drivers of a purchase, what statisticians call confounders, often are not captured

as data at all. This creates a gap: the variables that are easiest to measure are not always the ones that matter most.

Reality is messier than any dataset suggests. You need domain knowledge to fill in the gaps, and causal logic to untangle what actually drove what. That is where human judgment matters most.

Characteristic 2: Analysis runs on business time

Marketing data is not built for research. It is a byproduct of daily business operations that have to keep moving. Campaigns launch, budgets shift, tags break, attribution windows change, and platform definitions get updated while the business keeps running.

In academic research, you can wait until the data is enough and clean. In business, decisions cannot wait. An answer that arrives after the budget meeting is not useful.

Characteristic 3: Drivers rarely move independently

In a controlled experiment, you change one variable at a time and hold everything else constant. In real-world marketing, that almost never happens. The variables that drive sales tend to move together because something else is moving them all.

Consider a new product launch. TV ads, paid social, retail media, shelf-space expansion, and a price cut on the old product all happen at once because management planned them for the same launch window.

It is multicollinearity. The first step toward honest analysis is to acknowledge this constraint and be clear about what you can and cannot say.

1.4 Key Terms

A handful of marketing concepts will appear throughout the book without re-introduction. They are worth a quick glance before moving on.

- **CAC (Customer Acquisition Cost)** — what it costs to acquire one new customer.
- **LTV / CLV (Customer Lifetime Value)** — the expected revenue (or profit) from a customer over their relationship with the brand.
- **ROAS (Return on Ad Spend)** — revenue per dollar of ad spend, usually short-window and channel-specific.
- **Incrementality** — the causal lift attributable to an action, as opposed to the observed correlation. Most of Part 3 is about this.

Part 2. Analysis for Better Decisions

Photo by Headway on Unsplash

Most analyses do not fail because the numbers are wrong. They fail because they answer a question nobody needed answered.

Imagine spending two weeks preparing a deck: fifty slides, every KPI covered, clean charts, fresh data. The CMO listens for twenty minutes and then asks, "So what? What should we do?"

The problem is not the data quality. The problem is that the deck explains what happened, but not what decision should change. The CMO does not need a recap. She needs to decide where to move next quarter's budget.

That is what it means to design analysis for better decisions: start with the decision, then design the question, metrics, and story around it.

This part covers three practical skills: framing a vague request as a decision-oriented business question, designing KPIs that diagnose the problem rather than decorate a dashboard, and turning findings into a story that leads to action.

After this part, you will be able to:

- Start analysis from business decisions, not metrics
- Translate vague requests into clear business questions and hypotheses
- Design KPIs that diagnose business gaps and avoid vanity metrics
- Use KPI trees to connect outcomes, drivers, and actions
- Structure findings as a decision story using the SCR framework

2.1 Framing the Right Question

Before you even look at the data, get clear on the question your analysis needs to answer.

A useful question is not just interesting. It is tied to a real decision: who will do what differently if the analysis changes their mind?

In this chapter, I will walk through how to turn a vague request into a concrete business question, and then break that down into testable hypotheses.

Defining the Key Business Question

Before starting any analysis, ask yourself:

"As a result of this analysis, who in which department will take what action, and when? How much business impact can we expect from that action?"

If you can answer this concretely, the analysis is probably worth doing. If not, you are likely heading into a fishing expedition—wandering through data and hoping something useful turns up.

A useful business question passes three tests:

- Answerable — Can we answer it with available data before the decision deadline?
- Actionable — Can we name the owner and the decision it will affect?
- Worth it — Is the expected impact worth the analysis effort?

From Raw Request to Business Question

Most analysis requests do not arrive as clear business questions. They usually start out vague.

The analyst's job is not to take the request at face value. The real job is to translate that request into a decision.

Raw request	Better business question	Possible decision
Can you pull sales by channel?	Which channel explains the revenue gap?	Change next quarter's budget allocation
Why is revenue down?	Which customer group, product category, or channel is driving the decline?	Decide which owner should act this month
How did the campaign perform?	Should we increase, maintain, or reduce spend on this campaign?	Reallocate campaign budget
Can you check retention?	Which customer cohort is at risk?	Launch a reactivation campaign

The raw request tells you what data someone wants. The business question tells you what decision the analysis should change.

Hypothesis-Driven Design

Once you have a clear question, do not jump straight into the data. A question by itself is still too broad. Before you write any SQL, form specific hypotheses to narrow your search.

For example, instead of asking 'Why is revenue declining?', break it down: 'Is new customer acquisition falling?', 'Are repeat rates dropping?', or 'Is average order value down?'. Before you pull any data, sketch out what the answer would look like. Ask yourself: 'If this hypothesis is true, what chart or number would actually convince the decision-maker?'. Even a quick hand-drawn slide is enough. If you cannot sketch the output, your question or hypothesis is still too vague.

The key is not to think while you search. Lay out your hypotheses and possible conclusions first, then focus only on the validation you need. If the data contradicts your hypothesis, change course immediately. The moment you start cherry-picking data to fit your idea, you have left analysis and entered confirmation bias.

A simple template is enough:

```
Business question:
Hypothesis:
Metric to test:
```

```
Output needed:
Possible decision:
```

For example:

- Business question: Why did Q4 revenue miss plan?
- Hypothesis: Existing customers purchased less frequently.
- Metric to test: Purchase frequency by customer cohort versus last year and versus plan.
- Output needed: Cohort-level trend chart with the largest gap highlighted.
- Possible decision: Launch a reactivation campaign for the affected cohort.

This template keeps your analysis from drifting. If you cannot fill it in before pulling data, your question is probably still too vague.

The workflow is simple: Question → Hypothesis → Output design → Data → Analysis → Decision. The important point is the order — start with the question, not the data.

2.2 Designing KPIs for Diagnosis

Even with a clear question and hypothesis, you still need to decide what to measure. KPIs turn the business question into measurable drivers and into evidence the decision-maker can actually use.

What Makes a Useful KPI

A useful KPI has two requirements.

Diagnostic. It points to where the business problem is likely coming from. A KPI may not tell you the exact action immediately, but it tells you where to investigate next. If no team can respond to the metric, it is just an observation.

Aligned with business objectives. It has a credible link to the outcome the business cares about. Correlation isn't enough; you need to know that improving the KPI actually moves the end goal.

A common failure mode looks like this: an e-commerce team sets page views as its KPI. SEO drives up page views, but most of the new traffic does not buy, and revenue barely moves. Organic sessions to product and category pages would have been more useful, since that points the team toward traffic that is actually close to purchase.

KPI Tree: From Outcome to Drivers

A useful KPI hierarchy connects strategy to daily operations:

- KGI (Key Goal Indicator) — the ultimate business outcome (revenue, profit, market share).
- KPI (Key Performance Indicator) — the intermediate metrics that drive the KGI (new customer acquisition, repeat purchase rate, average order value).
- Action metrics — the operational levers teams control day-to-day (email send volume, ad impressions, page load speed).

This hierarchy forms a KPI tree. It breaks down the top-level outcome into the drivers that explain it. When the outcome moves, the tree tells you where to look first.

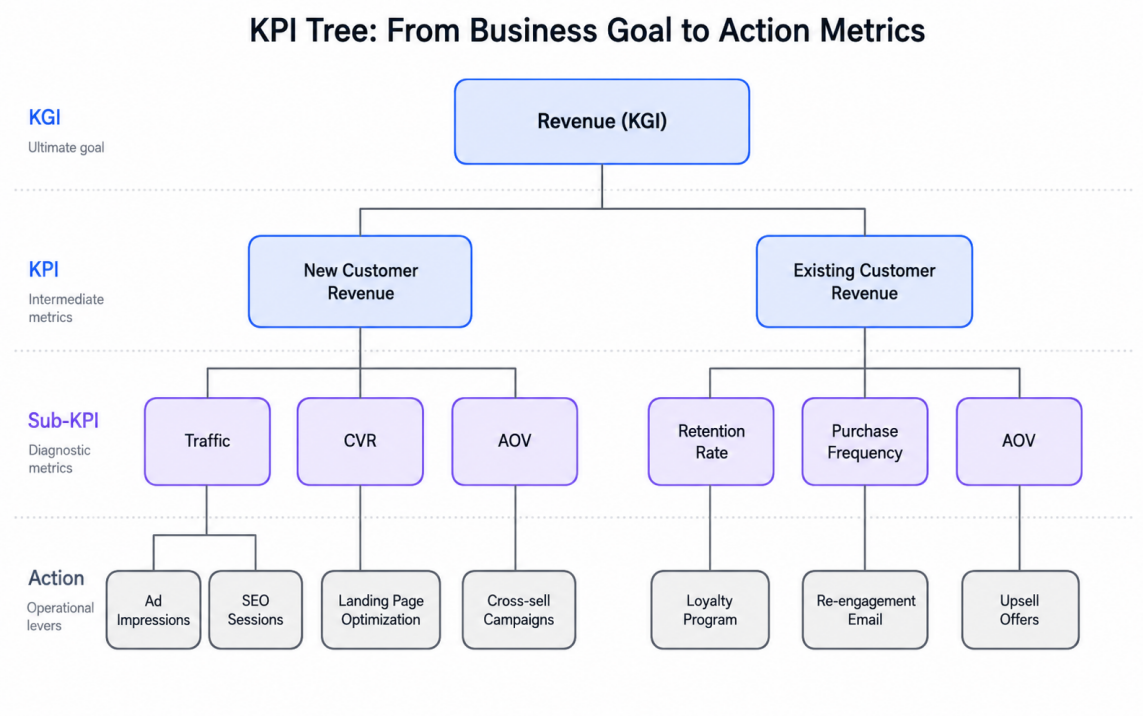


Figure 1: KPI tree: Revenue (KGI) → customer-cohort KPIs → Sub-KPIs → operational levers.

A good tree includes both lagging indicators that confirm results and leading indicators that give teams time to act.

Using the KPI Tree to Diagnose a Gap

"Can you just pull the numbers for our quarterly business review?"

This is the most common request data teams get. It sounds like a simple data pull, but it is actually a chance to add value. The difference between a report and a review is diagnosis. A report just lists numbers. A review explains why they moved and what to do next.

An effective business review follows four steps:

1. Start with the top outcome. Are we on track versus plan, forecast, and last year?

2. Decompose the gap. Which branch of the KPI tree explains the movement?
3. Diagnose the driver. Which hypothesis best explains the gap?
4. Recommend the next action. What should the owner do differently?

If you do not include diagnosis and a recommendation, a business review is just a report.

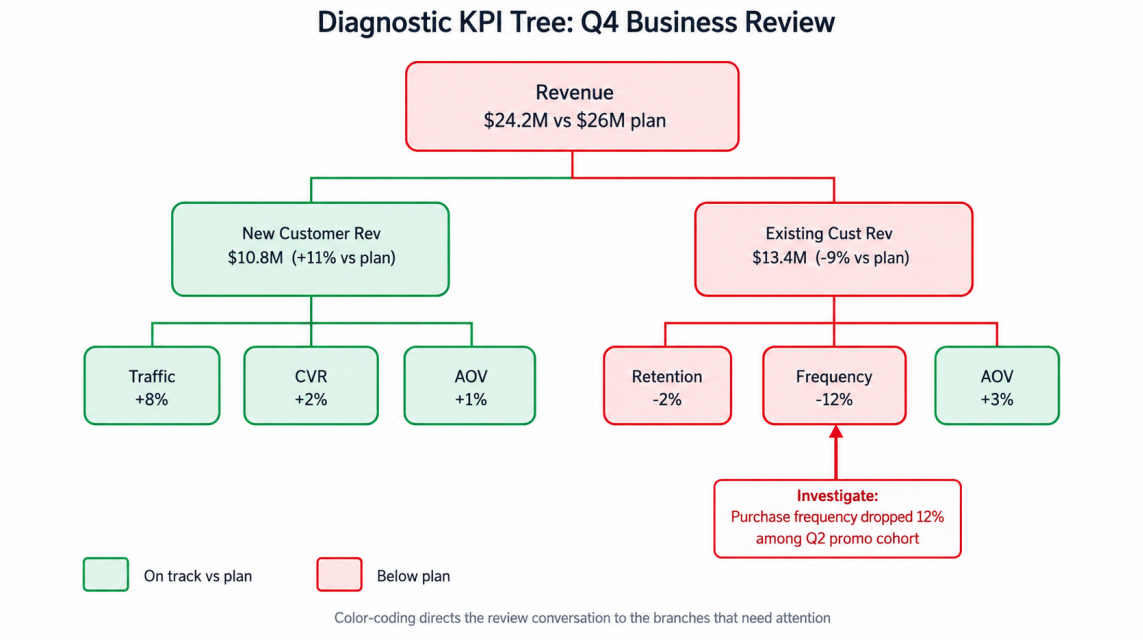


Figure 2: Diagnostic KPI tree for a Q4 review: $24.2M vs $26M plan, with the Existing Customer branch red.

Numbers only mean something when you compare them. A single number by itself does not tell you anything. Always use at least one benchmark:

- Year-over-year (YoY) — controls for seasonality. "Revenue is down 5%" in January might be fine if it is always down in Q1.
- Versus plan/forecast — measures execution against expectations.
- Versus market/competitors — separates company-specific issues from industry-wide trends.

So far, we have covered how to start from the right question and how to measure success. The next step is how to deliver findings as a story that actually drives action.

2.3 Turning Data into Story

At the end of any analysis, you can either present a list of findings or build a story that drives a decision. A decision story explains what changed, why it matters, and what to do next.

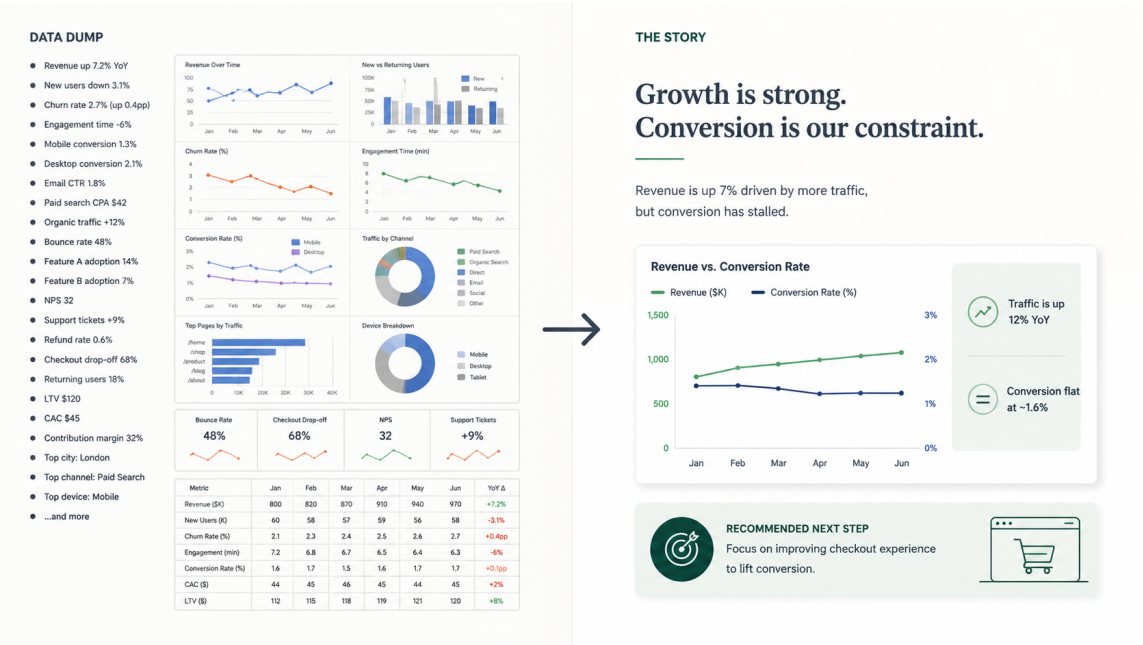


Figure 1

Audience Analysis

Before building the story, answer three questions:

1. Who makes the decision? Even if ten people are in the room, usually only one or two can actually act. Build your story for them.

2. What do they already believe? This becomes your Situation — the shared starting point your story builds on.
3. What action do you want them to take? This becomes your Resolution — the recommendation the analysis should produce.

The gap between #2 and #3 is what your data has to bridge.

The SCR Framework

The simplest and most effective story structure is Situation → Complication → Resolution (SCR).

Situation. What is the current state? What background does the audience already know? Start with facts they already accept.

Complication. What changed? What is the problem? What are the risks if nothing is done? This is where you show why action is needed.

Resolution. What does the data show, and what do you recommend? This is the ask.

Think of it this way: Situation is what the stakeholder believes now. Resolution is where you want them to end up. Complication, supported by your data, is the bridge. Before you build your presentation, write down these three parts clearly. If you cannot, the analysis is not ready.

Example: Explaining a Revenue Miss to the CMO

Recall the opening of this part: the CMO sat through fifty slides and asked, "So what? What should we do?" Here is what she actually needed.

The CMO has to decide where to shift next quarter's budget. That's the decision the analysis should change.

Situation. Revenue is growing, and new customer acquisition is on track.

Complication. Q4 finished below plan. The gap comes from existing customers: purchase frequency among the cohort acquired during Q2 promotions dropped sharply, pulling down existing-customer revenue.

Resolution. This is not a broad acquisition problem. Keep acquisition spend stable, but launch a reactivation campaign for the Q2 promotional cohort and tighten future promotion targeting.

The story is no longer just ‘here are the metrics.’ It is ‘the revenue gap is a retention problem, not an acquisition problem, and the next action is reactivation.’ That is the version the CMO can actually act on.

Visualizing the Story

Once the story is clear, pick only the charts needed to support it. Give each chart a title that states the message, not just the metric. The goal is not to show everything you analyzed. The goal is to make the decision easy to see.

Part 3. Causal Inference for Marketing

Photo by Artem Beliaikin on Unsplash

"Did the campaign actually cause the lift?" This section is about practical tools for answering that question in marketing. We'll cover A/B testing, quasi-experiments, meta-learners for understanding which customers respond differently, and uplift modeling to help you target the people who actually change their behavior.

After this part, you will be able to:

- Distinguish correlation from causation in marketing data
- Design, run, and analyze A/B tests correctly
- Estimate causal effects when randomization isn't possible
- Identify which customers benefit most from a treatment using uplift modeling

3.1 Causal Thinking

Imagine the following scenario. You're assisting a colleague from the marketing team, and they say:

"The marketing team sent coupons to some users, and their purchase rate was twice as high as those who didn't receive them. So, the coupons must have doubled the purchase rate!"

	Without Coupon	With Coupon
# of Customers	1,000	1,000
# of Purchasers	100	200
Purchase rate (%)	10%	20%

Figure 1: Coupon campaign results: 20% purchase rate with coupon vs. 10% without.

Would you agree with this conclusion? At first, it sounds reasonable. But what if you found out the marketing team sent coupons only to customers who had just browsed the product page, added items to their cart, or showed other signs they were ready to buy?

That should raise a red flag. These customers were already more likely to buy, with or without the coupon.

Selection Bias

The real question is: what did the coupons actually change? If we send coupons based on specific criteria, we can't fairly compare purchase rates. The people who got coupons might have bought anyway, since they were already more engaged.

> Tip
>
> Selection Bias

The distortion of a causal effect due to non-random assignment of participants, which leads to comparing groups that are not truly comparable.

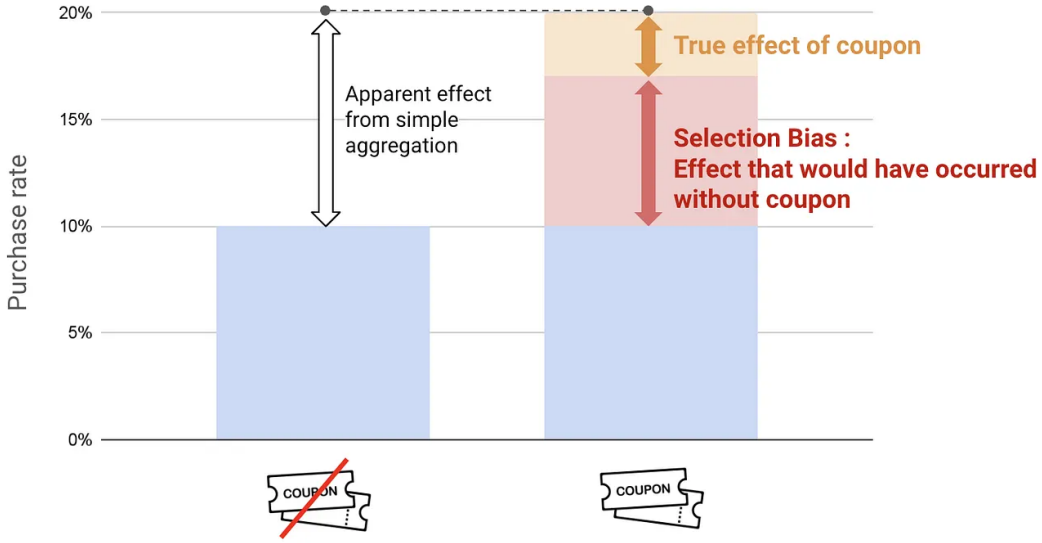


Figure 2: Most of the apparent effect is selection bias, not the true effect of the coupon.

The 10-point gap between groups looks impressive, but most of it was already there before the coupons went out. The real lift from the coupon—the part that wouldn't have happened otherwise—might be just a few points. The rest is selection bias: the coupon went to people who were likely to buy anyway.

You see this pattern all the time in marketing. Retargeting campaigns show high conversion rates, but that's because they reach people who already visited your site. Email campaigns look great on revenue per recipient, but those emails go to your most engaged subscribers. In both cases, the treatment and the outcome are linked because of customer intent, not because the treatment caused the outcome.

The Counterfactual

This leads to the key question: what would have happened if we hadn't done anything?

The main challenge in causal inference is that we only get to see one outcome for each person. If we had a time machine, we could give a customer a coupon, rewind, and see

what happens if we didn't. But in reality, we only see one version of the story for each customer.

> **Tip**
>
> Counterfactual
> The hypothetical outcome that we cannot observe for the same individual. The treatment effect is the difference between what would happen under treatment and what would happen under control.

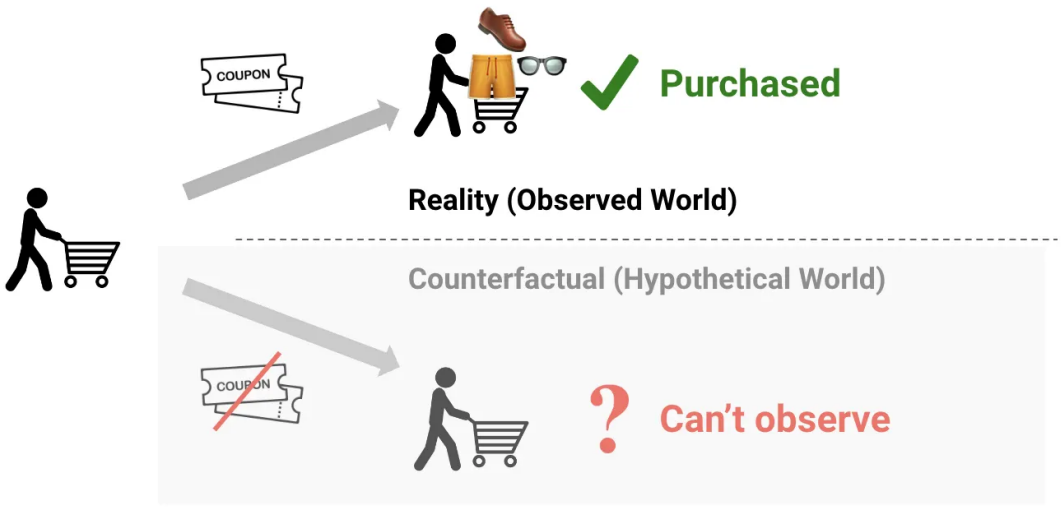


Figure 3: Fundamental problem of causal inference: we observe one outcome, never the counterfactual.

Formalizing this makes the problem precise. The treatment effect for a customer is:

$$\tau = Y(1) - Y(0)$$

Here, $Y(1)$ is what happens if someone gets the treatment, and $Y(0)$ is what happens if they don't. The problem is, we only ever see one of these for each person. If we just compare treated and untreated groups without accounting for this, we mix up the true effect with differences that were already there. That's selection bias.

Simpson's Paradox

You might think, "If confounders are the problem, I'll just look at the overall data more carefully." But confounders can produce wildly misleading results, even reversing the direction of an effect.

Consider a classic example from Judea Pearl's The Book of Why. Imagine a graph with exercise amount on the x-axis and cholesterol levels on the y-axis. At first glance, the data seems to show that people who exercise more actually have higher cholesterol levels. But that doesn't make sense!

What's missing? Age. If you break down the data by age group, you see a different story: within each group, people who exercise more usually have lower cholesterol. The overall trend reversed because older people both exercise more and have higher cholesterol. When you combine the groups, age acts as a confounder.

> **Tip**
>
> Simpson's Paradox
> A phenomenon where a trend appears in different groups of data but reverses or disappears when these groups are combined.

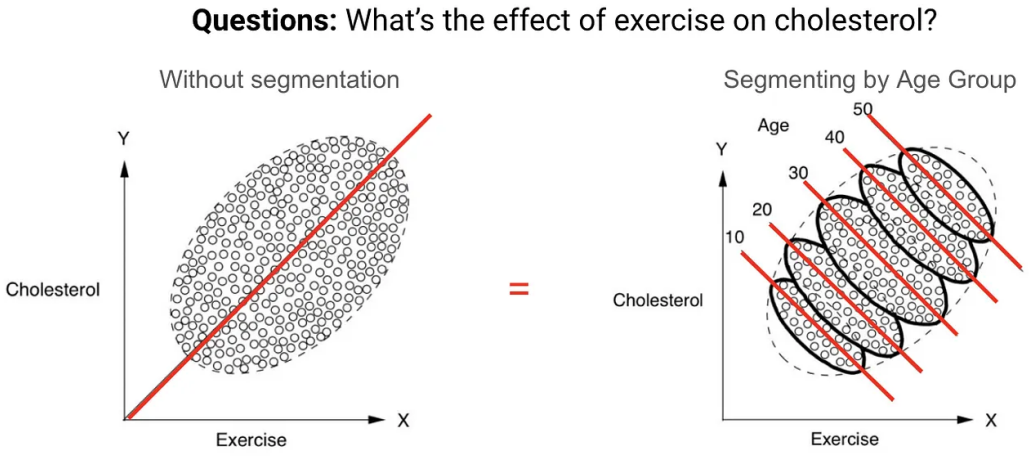


Figure 4: Simpson's Paradox: the overall trend reverses once you condition on age group.

This shows up frequently in marketing data. A channel might look unprofitable in aggregate but perform well within specific customer segments, or vice versa. The lesson: always ask whether there is a confounding variable hiding behind the numbers.

Why Not Just Run an A/B Test?

You might wonder, "Why not just run an A/B test?"

In causal inference, A/B testing is known as a Randomized Controlled Trial (RCT) and is often considered the "gold standard" for answering causal questions. By randomly assigning customers to treatment and control groups, we ensure that other factors (such as customer preferences or behavior) are balanced across both groups, which eliminates the bias that would otherwise distort the comparison.

> Tip
>
> Randomized Controlled Trial (RCT)
> An experimental method where participants are randomly assigned to groups to test the effect of an intervention. Because the assignment is random, any difference in outcomes can be attributed to the treatment rather than pre-existing differences.

So if we understand RCTs, does that mean they're always the perfect solution? Unfortunately, no. While RCTs are the strongest evidence available, there are many situations where they just aren't feasible:

1. Opportunity loss. If you run an RCT, half your customers might not get the new product or offer. That could mean missing out on sales, which isn't always acceptable for the business.
2. Time constraints. RCTs can be time-consuming. They often require weeks or even months to gather enough data to draw reliable conclusions. Time is money, after all!
3. Stakeholder complexity. RCTs become even more complex when multiple stakeholders are involved. Not every client or partner has the resources or ability to execute an RCT effectively.

That's why the rest of this section focuses on tools for measuring causal effects when you can't run an RCT, and for figuring out who actually benefits when you can.

Road Map for Part 3

Here's the plan. If you can randomize, A/B testing gives you the most reliable answers—but it's easy to make mistakes. If you can't randomize, quasi-experimental methods use natural variation to get as close as possible to what an experiment would show. Once you

have a causal estimate, meta-learners and uplift modeling help you see which customers actually respond, so you can avoid spending on people who would have converted anyway.

The next chapter starts with A/B testing, the gold standard. But even the gold standard has pitfalls that can lead you to the wrong answer.

3.2 A/B Test Design

Your team just finished a two-week A/B test on the new checkout page. Everyone is eager to see the results. Then the numbers come in:

Variant	Users	Conversions	CVR
Control	4,800	112	2.34%
Treatment	4,900	112	2.28%

The p-value is 0.42. The team says, "No significant difference. The redesign doesn't work. Let's move on."

But that conclusion isn't right—or at least, it's not backed up by the data. A p-value of 0.42 doesn't mean there's no effect. It just means the test couldn't tell a small effect from random noise. The difference between "no effect" and "we can't tell" comes down to whether your test had enough power.

The Power Problem

Statistical power is the chance your test will spot a real effect if it's there. Most teams aim for 80% power, which still means there's a 20% chance you'll miss a real effect even if it exists.

Here's the uncomfortable truth: for low-conversion events like purchases (say, a 2% conversion rate), you need tens of thousands of users per group to detect a 0.5 point lift. Most teams run tests with far fewer. This is the 1/10 problem: you often need about 10 times more samples than you think.

Figure 1: Statistical power: H0 and H1 distributions with alpha, beta, and (1−beta) regions.

Design Before You Run

The answer isn't to give up on A/B testing. It's to figure out the sample size you need before you start. The key inputs are:

- Baseline conversion rate (p_0): the current rate under the control condition.
- Minimum Detectable Effect (MDE): the smallest effect size you consider practically meaningful.
- Significance level (α, typically 0.05): the false positive rate you're willing to tolerate.
- Power ($1 - \beta$, typically 0.80): the probability of detecting the effect when it truly exists.

The intuition is simple: finding small effects in noisy data takes big samples. If your baseline conversion rate is 2% and you want to detect a 0.5 point lift, you'll need about 15,000 users per group. If you want to detect an even smaller lift, you'll need about four times as many.

Figure 2: Sample size sensitivity chart: halving MDE quadruples the required sample.

```
from statsmodels.stats.power import NormalIndPower
from statsmodels.stats.proportion import proportion_effectsize

# Example: CVR baseline 2%, MDE 0.5 pp, alpha=0.05, power=0.80
effect_size = proportion_effectsize(0.02, 0.025)
analysis = NormalIndPower()
n = analysis.solve_power(effect_size=effect_size, alpha=0.05, power=0.80)
print(f"Required sample size per group: {n:.0f}")
```

Do this calculation before you run the test, not after. Also, pre-register guardrail metrics—like revenue per user, page load time, or support tickets—that shouldn't get worse even if your main metric improves.

CUPED: Getting a Larger Sample for Free

You've done the sample size math, and the number is huge. Your product doesn't get enough traffic to hit it anytime soon. Is there a way to get the same precision with fewer users? CUPED (Controlled-experiment Using Pre-Experiment Data) helps by reducing

the variance of your metric using data you already have. It's like getting a bigger sample for free.

Take a customer who bought a lot in the month before the experiment. They'll probably keep buying a lot during the experiment, no matter which group they're in. Their high spending is mostly due to their past behavior, not the treatment. CUPED subtracts this predictable part from each user's outcome, so the treatment effect is easier to see. The stronger the link between pre-experiment and in-experiment behavior, the more variance CUPED removes, and the fewer users you need.

Formally, let Y be the outcome and X a pre-experiment covariate (e.g., purchases in the prior 30 days). The adjusted outcome is:

$$Y_{\text{CUPED}} = Y - \theta \cdot (X - \bar{X})$$

where $\theta = \text{Cov}(Y, X)/\text{Var}(X)$. The key result: the variance of the adjusted outcome reduces to $\text{Var}(Y_{\text{CUPED}}) = \text{Var}(Y)(1 - \rho^2)$, where ρ is the Pearson correlation between X and Y. If $\rho = 0.7$, CUPED removes roughly half the variance, equivalent to doubling your sample size. Netflix and Microsoft report 20–50% reductions in required sample sizes using this technique. The most common covariate choice is the same metric measured in the pre-experiment period, which typically yields the strongest correlation.

Warning

Peeking Inflates False Positives
Checking results repeatedly before the test reaches its planned sample size increases the false-positive rate. If you check daily for 14 days and stop as soon as you see significance, the effective alpha can be 2–3 times higher than your nominal 0.05. Either commit to a fixed horizon or use sequential testing methods (such as confidence sequences) that formally budget alpha across interim looks.

Figure 3: Peeking and alpha inflation: daily checking inflates the nominal 5% alpha to 20-30%.

Warning

Multiple Testing
When you test 20 metrics simultaneously, one of them will be "significant" at the 5% level by chance alone. Pre-register your primary metric and analyze it at nominal alpha. Treat secondary and exploratory metrics with appropriate skepticism; apply

Bonferroni or Benjamini-Hochberg corrections to control false discoveries.

How to Interpret Null Results

A non-significant result doesn't mean the treatment had no effect. It just means the test didn't find an effect at your chosen significance level. Before you say "no effect," check:

- Was the test adequately powered for the MDE you're interested in?
- Is the confidence interval narrow enough to rule out meaningful effects?
- Could the test period have been atypical (holidays, outages, competing campaigns)?

Absence of evidence is not evidence of absence. If the 95% confidence interval for the treatment effect is [-1%, +2%], you cannot claim "no effect." You can only say the data are consistent with effects ranging from a 1 percentage-point decrease to a 2 percentage-point increase.

But what if you can't randomize? Maybe the treatment is already live, or there are ethical reasons you can't run an experiment. That's when you turn to quasi-experimental methods.

3.3 Quasi Experiments

Sometimes you can't run an A/B test. The intervention has already happened. Legal or ethical constraints prevent randomization. The treatment is applied at a level (store, region, time period) where randomization is impractical. In these cases, quasi-experimental methods let you estimate causal effects by exploiting structure in how the treatment was assigned.

This chapter focuses on the most widely used workhorse: Difference-in-Differences. We'll also briefly cover two other methods (Regression Discontinuity and Synthetic Control) so you can recognize when your data supports them.

Difference-in-Differences (DiD)

Here's a concrete scenario. A retailer rolls out a new loyalty program in 50 stores (treatment) while keeping 50 stores without it (control). You measure average basket size before and after the rollout in both groups.

	Before	After	Change
Treatment stores	\$42	\$48	+\$6
Control stores	\$40	\$43	+\$3
DiD estimate			+\$3

The treatment stores grew by \$6, but control stores also grew by \$3, due to seasonality, inflation, or other trends that affect everyone. The DiD estimate is \$6 - \$3 = \$3: the incremental effect of the loyalty program. This is Difference-in-Differences: it compares the change in outcomes over time between a treated group and an untreated group. The "double differencing" removes both time-invariant differences between groups and common time trends.

You might be thinking, "That seems almost too simple." And in a way, it is. The elegance of DiD is that it cancels out factors you can't measure, as long as one critical assumption holds.

The Parallel Trends Assumption

DiD is valid only if the treatment and control groups would have followed the same trend in the absence of treatment. This is untestable in the post-treatment period (we can't observe the counterfactual), but you can check for parallel trends in the pre-treatment period. If the two groups tracked each other closely for several periods before the intervention, it's more plausible (though not guaranteed) that they would have continued to do so.

Figure 1: Difference-in-Differences: parallel pre-trends, then post-intervention divergence.

How to test it:

- Plot the outcome over time for both groups. Do the pre-treatment trends look parallel?
- Run a formal pre-trend test: regress the outcome on group-by-time-period interactions for the pre-treatment periods and test whether the interaction coefficients are jointly zero.
- If trends diverge before treatment, DiD is suspect. Consider matching, weighting, or a different method.

When to use DiD in marketing: Promotion rollouts across stores, policy changes that affect some markets but not others, new feature launches in one app version, or any setting where you have pre/post data for treated and untreated groups.

Other Quasi-Experimental Methods

Regression Discontinuity (RDD)

Regression Discontinuity exploits situations where treatment is assigned based on whether a continuous variable crosses a threshold. Units just above and just below the cutoff are nearly identical on all characteristics except their treatment status, creating a local experiment.

A loyalty program grants "Gold" status to customers who spend more than $500 per year. Gold members receive exclusive discounts. To estimate the effect of Gold status on subsequent spending, you compare customers who spent just above $500 with those who spent just below. A customer who spent $502 is essentially identical to one who spent $498; the only systematic difference is that one crossed the threshold.

Figure 2: Regression Discontinuity Design: regression lines show a jump at the threshold, with the bandwidth region highlighted.

The key requirements: no manipulation of the running variable (customers can't game the threshold), and all variables other than the treatment should change smoothly through the cutoff. RDD estimates a local treatment effect, valid near the cutoff, but not necessarily generalizable to customers far from the threshold.

Synthetic Control

Synthetic Control constructs a counterfactual for a treated unit by creating a weighted combination of untreated units that closely matches the treated unit's pre-treatment trajectory.

A brand launches a major campaign in California but not in other states. To estimate the campaign's effect on sales, you construct a "synthetic California" from a weighted combination of other states that collectively match California's pre-campaign sales trend, demographics, and market characteristics. After the campaign, the gap between actual California and its synthetic counterpart is the estimated treatment effect.

Figure 3: Synthetic Control: actual trajectory vs. synthetic counterfactual.

Synthetic Control works best when you have a single (or few) treated units with a long pre-treatment time series. The pre-treatment fit must be tight. If the synthetic control can't reproduce the treated unit's pre-treatment trajectory, the estimate is unreliable.

Choosing the Right Method

No method is assumption-free. The goal is to choose the method whose assumptions are most defensible for your specific setting:

- Sharp threshold that determines treatment? Consider RDD.
- Clear before/after period with treated and untreated groups? Consider DiD.
- Single treated unit with a long time series? Consider Synthetic Control.
- None of the above? You may need matching, propensity score weighting, or instrumental variables, and you should consider redesigning the next intervention to enable a cleaner evaluation (e.g., a staggered rollout for future DiD analysis).

These methods tell us the average effect of a treatment. But what if different customers respond differently? A coupon might boost purchases by 5% on average, but that average blends customers for whom the lift was 20% with customers who saw zero effect. The next chapter tackles that question.

3.4 Meta Learners for Treatment Effects

The methods in previous chapters estimate a single number: the Average Treatment Effect (ATE), or the average impact of a treatment across everyone. But in marketing, averages can be misleading. For example, a coupon might increase purchases by 5% on average, but that number hides the real story. Some customers might respond with a 20% lift, while others see no effect or even a drop. If you could identify which customers actually respond, you could spend your budget much more effectively.

Meta-learners try to estimate treatment effects for each individual customer. But what does that actually look like in practice?

Treatment Effects: ATE, CATE, ITE

To define the treatment effect, we turn to the Potential Outcomes Framework. For example, if the potential purchase rate with a coupon is 20% and the purchase rate without the coupon is 10%, the treatment effect is 10%.

> **Tip**
>
> Treatment Effect
> The difference in outcomes caused by a treatment. Formally: $\tau = Y(1) - Y(0)$, where $Y(1)$ is the outcome under treatment and $Y(0)$ is the outcome under control.

Now, there are three levels of granularity:

- ATE (Average Treatment Effect): the effect averaged across all customers. "Did the coupon increase sales?"
- CATE (Conditional Average Treatment Effect): the average effect for customers with specific characteristics. "Did the coupon work better for urban customers under 30?"
- ITE (Individual Treatment Effect): the effect for a specific individual. Unobservable in practice, but CATE provides our best estimate.

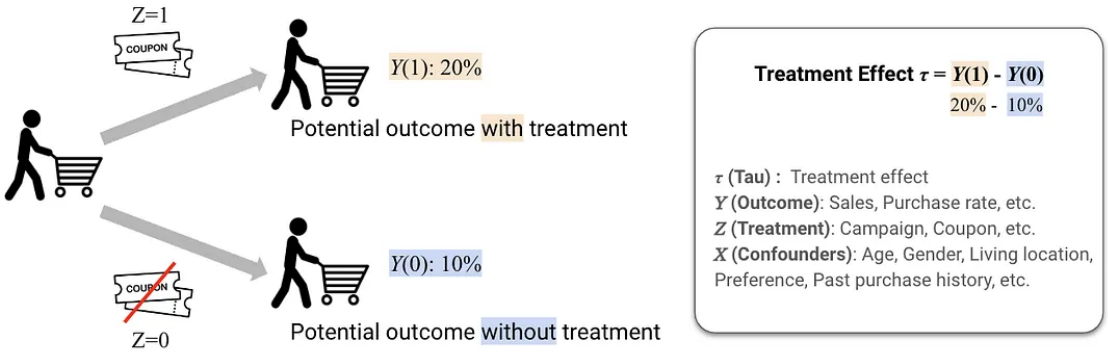


Figure 1: The treatment effect is the difference between potential outcomes: what happens with treatment versus without.

Why does this matter? Because not all customers behave the same way. A price-sensitive customer might jump at a discount, while a brand-loyal customer would have bought at full price anyway. If you know who is who, you can target discounts where they actually drive extra sales, instead of giving away margin to customers who would have purchased regardless.

The Missing Data Problem

Consider this question: did a coupon increase sales? To answer it, we look at data from various customers, including their attributes (age, gender, location), the treatment (coupon or no coupon), and the outcome (purchase or not).

The column $Y(1)$ represents the outcome when the coupon is used, while $Y(0)$ represents the outcome without the coupon. If we had both columns for the same individual, calculating the treatment effect would be as simple as subtracting one from the other. But the challenge is that we can only observe one outcome for each individual.

To solve this, we use meta-learners. These are machine learning frameworks that predict what would have happened if a customer had received the other treatment. By filling in these missing counterfactuals, we can estimate the treatment effect for each person.

S-Learner: Start with a Single Model

The "S" in S-learner stands for "Single," as it relies on a single machine learning model. Train one model on all data with the treatment indicator as an additional feature: $\hat{Y} = f(X, Z)$. Then estimate CATE by predicting with treatment set to 1 versus 0:

$$\hat{\tau}(x) = \hat{f}(x, 1) - \hat{f}(x, 0)$$

Pseudo-code:

```
# Train a single model with treatment as a feature
X = df[['age', 'gender', 'location', 'treatment']]
y = df['sales']

model = xgb.XGBRegressor()
model.fit(X, y)

# Predict outcomes under treatment and control
df['sales_treated'] = model.predict(X.assign(treatment=1))
df['sales_control'] = model.predict(X.assign(treatment=0))

# Treatment effect = difference
df['treatment_effect'] = df['sales_treated'] - df['sales_control']
ATE = df['treatment_effect'].mean()
```

S-learner is the simplest way to start, and often a reasonable baseline. But it has a key weakness: if the treatment effect is small compared to other features, the model can end up ignoring the treatment variable, especially if you use regularization. This means your CATE estimates can be biased toward zero.

T-Learner: Two Separate Models

The "T" in T-learner stands for "Two." Instead of one model, train separate models for the treated and control groups:

$$\hat{\tau}(x) = \hat{f}_1(x) - \hat{f}_0(x)$$

Pseudo-code:

```
# Split data into treated and control
df_treated = df[df['treatment'] == 1]
df_control = df[df['treatment'] == 0]

# Train separate models
model_treated = xgb.XGBRegressor()
model_treated.fit(df_treated[['age', 'gender', 'location']], df_treated['sales'])

model_control = xgb.XGBRegressor()
model_control.fit(df_control[['age', 'gender', 'location']], df_control['sales'])

# Predict and subtract
df['sales_treated'] = model_treated.predict(df[['age', 'gender', 'location']])
df['sales_control'] = model_control.predict(df[['age', 'gender', 'location']])

df['treatment_effect'] = df['sales_treated'] - df['sales_control']
ATE = df['treatment_effect'].mean()
```

T-learner is more flexible because each model can learn different patterns for treated and control groups. But there is a tradeoff: it can have high variance, especially if your treatment and control groups are imbalanced. When you subtract the predictions, errors from both models can add up.

So, S-learner is simple but tends to underestimate effects. T-learner fixes that bias but can be noisy. What can we do next?

Advanced Meta-Learners

Beyond S and T, several more sophisticated methods address their limitations:

- X-Learner: An extension of T-learner designed for imbalanced treatment groups. It imputes the missing counterfactuals and combines them using propensity-score weighting. Works well when one group is much larger than the other.
- DR-Learner (Doubly Robust): Combines outcome modeling and propensity score estimation. Consistent if either model is correctly specified, hence "doubly robust."
- DML (Double Machine Learning): Uses cross-fitting to avoid overfitting bias, with strong theoretical guarantees. More computationally expensive (~27x slower than S-learner) but provides proper confidence intervals.

Learner	Approach	Reference	Training Speed (relative to S learner)
S Learner	**S**ingle Model Approach	Künzel et al. 2019	1x
T Learner	**T**wo-Model Approach	Künzel et al. 2019	2x
X Learner	**Cross**-Fitting Approach	Künzel et al. 2019	5x
DR Learner	**D**oubly **R**obust	Kennedy et al.2020	13x
DML	**D**ouble **M**achine **L**earning	Chernozhukov et al. 2018	27x

Figure 2: Comparison of meta-learner methods: from simple (S-learner) to complex (DML).

Theory tells us what each method is supposed to optimize. But the real test is seeing how they perform on the same dataset, where you can see the practical differences.

Seeing It in Practice: The Criteo Dataset

We demonstrate meta-learners on the Criteo Uplift dataset, a large-scale advertising RCT released by Criteo AI Lab, containing ~14 million samples with 12 anonymized features. About 85% of users were shown ads (treatment) and 15% were not (control), with a ~0.3% conversion rate.

Since this is a randomized experiment, just comparing conversion rates between treated and control groups gives you an unbiased ATE. The real value of meta-learners here is in showing the differences hidden under that average.

> **i** Run this notebook
>
> `simple_criteo_meta_learners_econml.ipynb` — Open in Colab · nbviewer
> Committed with outputs; Colab is best-effort. Full list on the Notebooks & Code page.

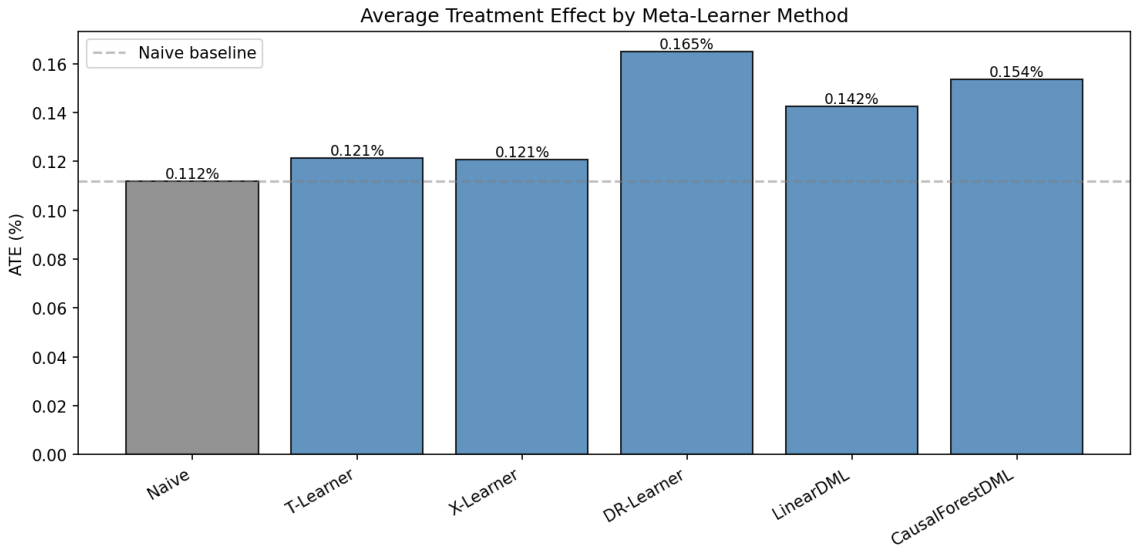


Figure 3: Five meta-learners recover ATEs on RCT data (S-learner excluded; see text).

Most Methods Recover the Average

T-learner and X-learner both match the naive baseline closely, which is what you expect in a randomized experiment with no selection bias. The DML-family estimators (like LinearDML and CausalForestDML) and DR-learner give slightly higher estimates, mostly due to cross-fitting variance, but they are still in the same ballpark. S-learner is not shown in this chart because, when the treatment effect is small compared to the noise, its single model tends to treat the treatment indicator like any other feature. Regularization then shrinks its effect toward zero, so the ATE estimate ends up much lower than the true value. This is a known limitation, not a bug. You can still see S-learner in the CATE distributions and uplift curves, where its narrow spread shows the same bias at the individual level.

But Heterogeneity Is Hidden Underneath

The histograms show what the average hides: treatment effects are not the same for everyone. Some users have much higher CATEs than average—they are persuadable. Others are clustered near zero, meaning they would convert (or not) no matter what. This kind of heterogeneity is exactly what you want to find if you are building a targeting strategy.

You can see the differences between methods in both the shape of the distributions and the summary stats. S-learner gives a narrow distribution because its single model tends to

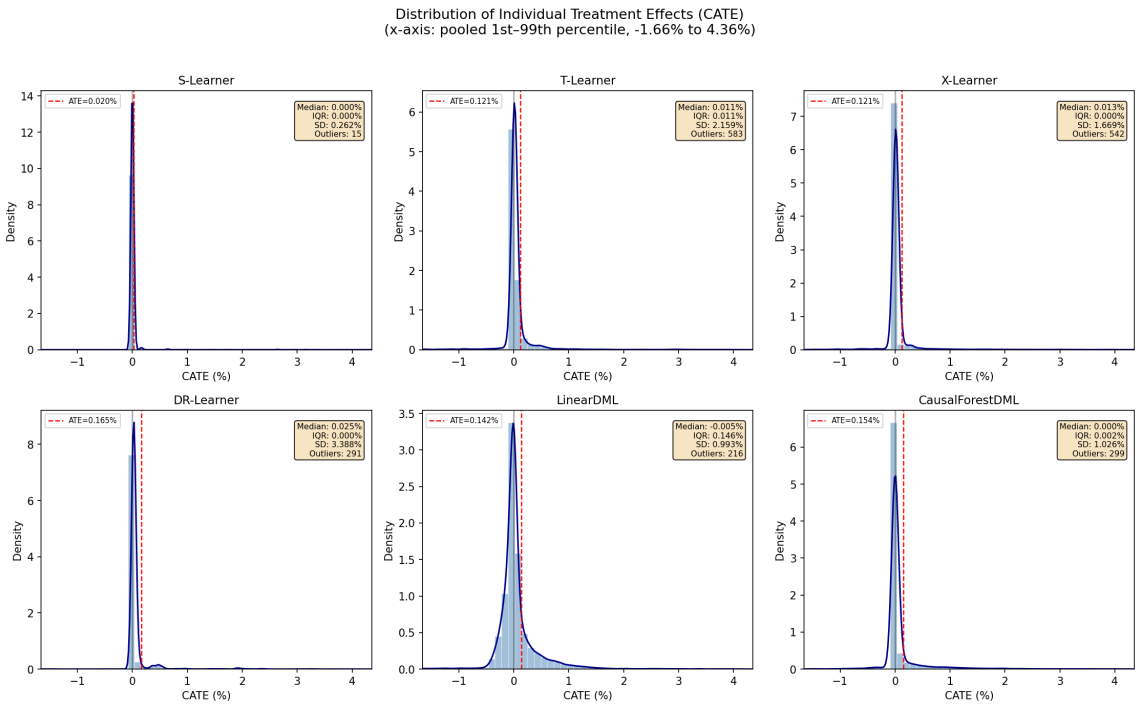


Figure 4: CATE distributions across six meta-learners on a shared x-axis (1st–99th pct).

shrink the treatment effect toward zero, so it underestimates heterogeneity. CausalForestDML gives a wider, smoother distribution, which fits with its flexible, nonparametric approach. DR-learner has the heaviest tails, since inverse-propensity weighting can amplify noise when the treatment and control groups are imbalanced.

Now we know who responds. The next question: how do we turn that into a targeting decision?

3.5 Uplift Modeling for Targeting

Who should you actually target to maximize sales? Most marketing teams use a response model: predict who is most likely to buy, then send coupons to those customers. This seems logical, but it often wastes money.

Here's the problem: many customers who are likely to buy will do so whether or not you send them a coupon. Giving them a discount just gives away margin without increasing sales. At the same time, there are customers with a moderate chance of buying who might convert if you nudge them. These are the people where the coupon actually makes a difference.

- A response model predicts: $P(\text{purchase})$
- An uplift model predicts: $P(\text{purchase}|\text{treated}) - P(\text{purchase}|\text{not treated})$

A response model ranks customers by how likely they are to buy. An uplift model ranks them by how much the treatment actually changes their behavior. These rankings can be very different. For example, a customer with an 80% chance of buying might have zero uplift (a Sure Thing), while someone with a 30% chance might have a 15% uplift (a Persuadable).

The Four Customer Types

Now, let's take a closer look at the customers to be targeted. We can segment customers into four types based on how they respond to treatment:

- Persuadables: Will buy only if they receive the treatment. These are the ideal targets: the coupons change their behavior. Your entire marketing ROI comes from this group.
- Sure Things: Will buy no matter what. Sending them a coupon costs you the discount but generates no incremental revenue.
- Lost Causes: Won't buy regardless of whether they receive the coupon. Targeting them wastes both the coupon cost and any impression cost.

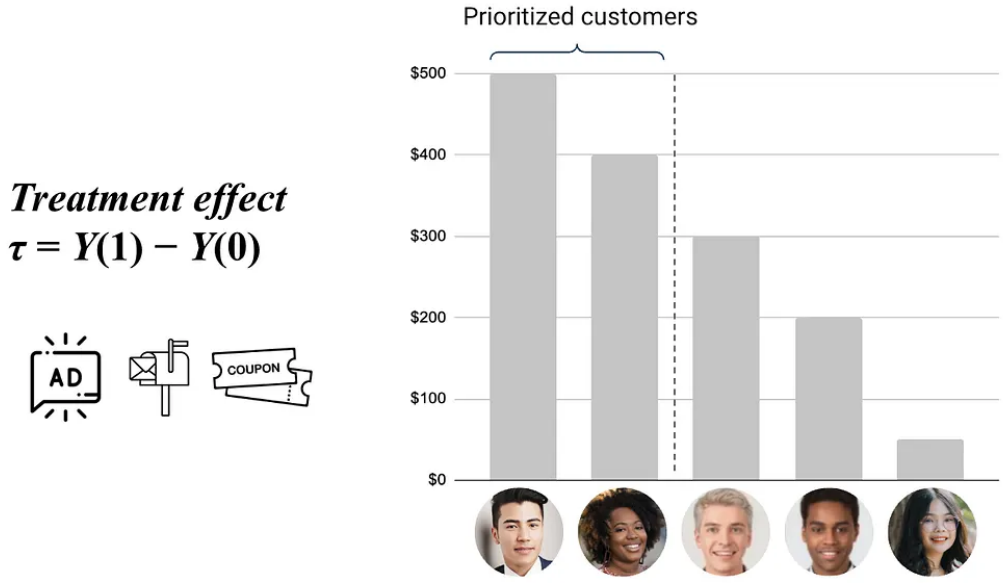


Figure 1: Uplift modeling prioritizes customers by their estimated treatment effect — not by their likelihood of purchase.

- Sleeping Dogs: Will not buy if they receive the coupon, but would buy without it. These customers are actively harmed by the intervention. The coupon may trigger annoyance, unsubscription, or brand damage.

"Don't wake the Sleeping Dogs" is not just a metaphor. In real campaigns, some customer segments actually show negative uplift from coupons. Uplift modeling helps you find these segments so you can avoid targeting them.

From Decision Trees to Uplift Trees

This four-type framework gives you a clear target. The next question is: how do you actually build a model that separates Persuadables from Sure Things? Earlier, we looked at meta-learners that wrap standard ML models. Uplift trees take a different route. They are decision trees specifically designed to find where the treatment effect varies the most.

A traditional decision tree asks: "Will the customer purchase?" An uplift tree asks: "Who should we give coupons to?"

Figure 2: The four customer segments: Persuadables, Sure Things, Lost Causes, and Sleeping Dogs.

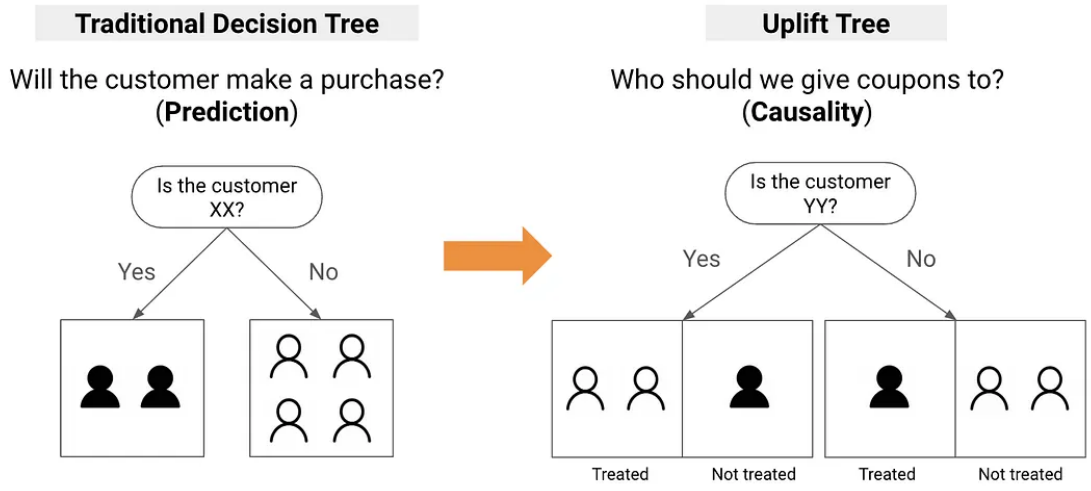


Figure 3: Decision tree vs. uplift tree: predicting outcomes vs. predicting treatment effect.

How Traditional Trees Split

Consider two possible trees. The tree on the left splits by "Age," while the tree on the right splits by "Location." Which tree provides a better split?

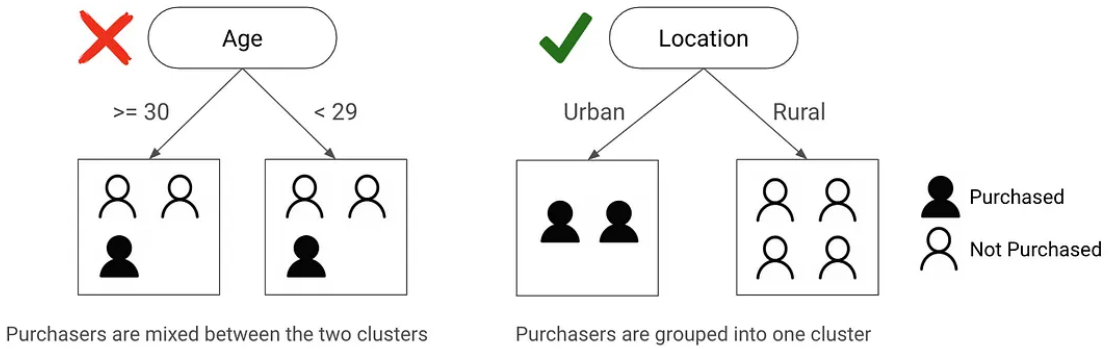


Figure 4: Traditional decision tree: splits on features to separate purchasers from non-purchasers.

The tree on the right does a better job because it separates purchasers from non-purchasers more clearly. We measure this using Gini Impurity, which tells us how mixed each node is:

$$\text{Gini} = 1 - (p_1^2 + p_0^2)$$

where p_1 is the probability of purchase and p_0 is the probability of no purchase.

Weighted Gini Impurity for the split:

$$= \frac{3}{6} \cdot Gini_{\text{left}} + \frac{3}{6} \cdot Gini_{\text{right}}$$

$$= \frac{3}{6} \cdot \left(1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)\right) + \frac{3}{6} \cdot \left(1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)\right)$$

$$= \frac{4}{9}$$

$$\approx 0.44$$

Figure 5: Gini impurity measures prediction accuracy — not uplift.

But here's the key point: Gini measures how well you can predict purchases, not uplift. A tree that predicts who will buy is not the same as a tree that finds where the treatment actually changes behavior.

How Uplift Trees Split

An uplift tree splits to maximize the difference in treatment effect between child nodes. Instead of Gini impurity, it uses a divergence metric that measures how differently treated and control groups behave within each node:

- KL Divergence: measures how the treatment group's outcome distribution diverges from the control group's.
- Euclidean Distance: simpler metric based on the squared difference between treatment and control response rates.
- Chi-Squared: tests whether the difference between treatment and control is statistically significant within each node.

In practice, KL Divergence is the most commonly used and the default in most uplift tree implementations. Euclidean Distance is simpler and computationally faster but less sensitive to small differences. For most marketing applications, the default (KL Divergence) works well.

Training an Uplift Model

Several open-source libraries implement uplift trees with these splitting criteria — `causalml`'s `UpliftRandomForestClassifier` and `scikit-uplift`'s `SoloModel` / `ClassTransformation` families, among others. They share a common shape: `fit(X, treatment, y)` returns a model whose `predict(X)` produces an uplift score per customer.

A meta-learner gives you the same ingredient by another route. A T-Learner CATE — the predicted incremental conversion probability from §3.4 — is itself a per-customer uplift score, and we can hand it to the same evaluation and targeting tools:

```
# CATE from §3.4 is already an uplift score per customer.
from econml.metalearners import TLearner
from lightgbm import LGBMRegressor

t_learner = TLearner(models=[LGBMRegressor(), LGBMRegressor()])
t_learner.fit(Y=y_train, T=treatment_train, X=X_train)
```

```
uplift_score = t_learner.effect(X_eval)   # one number per held-out customer
```

The full runnable example — fitting six meta-learners on the Criteo Uplift dataset, then evaluating their rankings — lives in the §3.4 companion notebook (see callout in Meta Learners for Treatment Effects).

Evaluating Uplift Models

Standard classification metrics like AUC or F1 measure how well you predict outcomes, not how well you target. For uplift modeling, you need metrics that are designed for targeting quality.

The uplift curve (or gain curve) plots cumulative incremental conversions against the fraction of the population targeted. Customers are ranked by their predicted uplift score, and we compute the total uplift gained as we move down the ranking:

- A good uplift model will concentrate most of the incremental conversions in the top deciles. If you target the top 20% of customers, you should capture most of the total uplift.
- A random targeting strategy follows a straight diagonal line.
- The area between the model curve and the random baseline is the AUUC (Area Under the Uplift Curve). `scikit-uplift` reports it normalized to `[0, 1]` — 1.0 is a perfect ranking, 0.0 is no better than random.

```
from sklift.metrics import uplift_curve, uplift_auc_score

# uplift_score is any per-customer ranking signal - a CATE from §3.4,
# the output of an uplift tree, anything you want to grade.
auuc = uplift_auc_score(y_true, uplift_score, treatment)
x, y = uplift_curve(y_true, uplift_score, treatment)
```

The §3.4 companion notebook (§10) runs this exact recipe on the T-Learner CATE for the Criteo Uplift dataset and plots the curve against a random-targeting diagonal — the model curve sits clearly above it, which is what tells you the ranking is doing work.

The Qini curve is a similar metric, but it normalizes by the number of treated customers instead of the whole population — more robust when the treated share varies across score buckets. Both metrics help you see if your model is actually ranking customers by their true uplift.

From Scores to Targeting Decisions

An uplift model gives you a score for every customer. The simplest way to use it is to rank all customers by their uplift score, split them into deciles, and then target the top N deciles that fit your budget.

```
import pandas as pd

scored = pd.DataFrame({'customer_id': customer_id, 'uplift': uplift_score})
ranked = scored['uplift'].rank(method='first', ascending=False)
scored['decile'] = pd.qcut(ranked, q=10, labels=False)   # 0 = highest uplift
```

If your budget lets you target 20% of customers, just take the top two deciles. This gives you a list of customers ordered by expected incremental value.

The §3.4 companion notebook (§9) extends this with a per-decile observed-uplift table and a net-value column that subtracts treatment cost — the same workflow, made operational.

Guardrails

Two failure modes to watch for:

Margin erosion. The most common failure in coupon targeting is when the extra revenue from coupons is less than the cost of the discounts. Always include the coupon cost in your uplift calculation: net uplift = estimated treatment effect times margin minus coupon cost. Only target customers where the expected extra margin is greater than the coupon value.

Customer fatigue. Even Persuadables will stop responding—or even disengage—if you contact them too often. Set frequency caps (limits on how many times you contact each customer) and watch if uplift drops for customers who have been contacted recently.

That wraps up Part 3. We started with a common question: a colleague claims the coupon doubled the purchase rate, but we saw that was just selection bias. From there, we built a toolkit to answer the real question: not just whether a campaign works, but who actually benefits. We covered A/B testing, quasi-experiments, meta-learners, and uplift modeling. The real goal is not just to know if a campaign works, but to know who it works for—so you can spend your budget where it matters.

Part 4. Customer Analytics

Photo by Vitaly Gariev on Unsplash

Part 4 zooms out. Part 3 asked which customers should receive a specific campaign. Now the question is broader: how should the business treat the customer base as a whole?

For decades, advertisers bought media using broad demographic buckets like "Women 25–54." P&G later described this as "wasteful mass marketing" and shifted toward hundreds of behavior-based smart audiences (Vizard 2019). In practice, a segment only matters if it changes your actions.

Part 4 is built around two questions:

1. Who are our customers, and how should we treat different groups differently?
2. What is each customer worth over time, and how much can we afford to spend to acquire and retain them?

Segmentation answers the first. CLV modeling answers the second. Together, they form a customer decision system: who gets attention, and how much you spend on each group.

4.1 Customer Segmentation

A useful segmentation project should deliver an operating artifact, not just a slide deck. The real output is a `segment_id` that integrates with your CRM and ad platforms, along with a one-page brief for each segment: who they are, why they matter, and what your team should do differently.

This chapter builds that artifact in three steps:

1. Rank customers by current value. Decile analysis and RFM give you a fast operating layer.
2. Group customers by behavior. Behavioral clustering uncovers patterns that simple value tiers miss.
3. Add purchase meaning. Embedding-based segmentation reveals what customers actually buy, not just how much.

Build Segments from Behavior, Profile Them with Demographics

Before building segments, separate two types of variables.

Behavioral variables describe what customers do: how often they buy, how recently they bought, how much they spend, which categories they buy, how broadly they shop, how often they return products, and how dependent they are on discounts.

Profile variables describe who they are or where you can reach them: age, income, household composition, geography, loyalty tier, and channel preference.

Use behavioral variables to create the segments. Use profile variables afterward to describe, target, and communicate with those segments.

This rule holds throughout the chapter. RFM uses behavioral signals to build value tiers. K-Means adds richer behavioral features to uncover broader patterns. Embedding-based segmentation brings in purchase text to capture what customers actually buy. Demographics still matter, but their main role comes after the segments are built: to explain, reach, and activate them.

Start Simple: Deciles and RFM

The quickest way to a usable segmentation skips clustering algorithms entirely. Start by ranking customers by revenue decile, then layer on RFM if you need more nuance.

Decile Analysis

Decile analysis divides customers into ten equal groups by revenue. The pattern usually follows the Pareto principle: a small share of customers drives most of the revenue. D1 gets retention and VIP treatment. D2 to D5 get growth campaigns—this is where the biggest upside is. D8 to D10 get only low-cost automated nurture. You can set this up in under an hour and give your team a clear action plan.

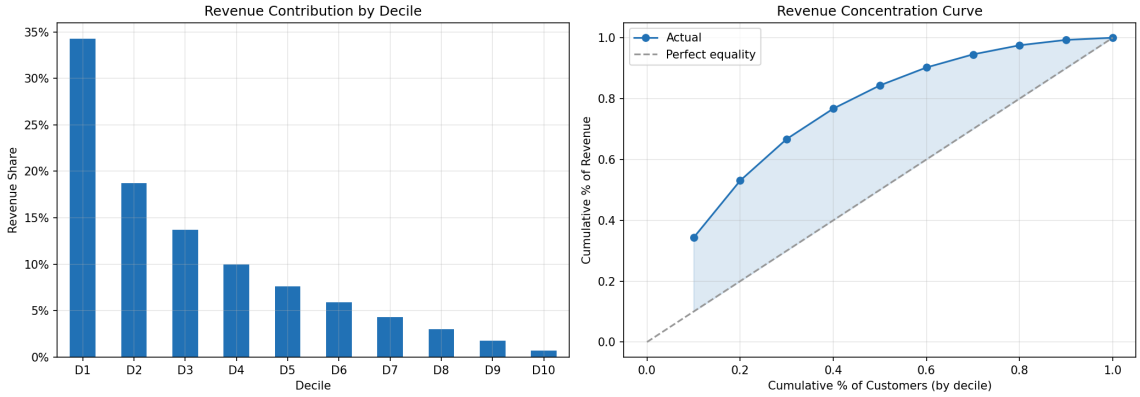


Figure 1: Typical decile distribution. Decile 1 contributes roughly 35% of revenue, with a steep drop-off.

RFM: Adding Behavioral Nuance

RFM (Recency, Frequency, Monetary) gives a more nuanced view. It is the most widely used segmentation framework in direct marketing because marketing teams already think in these terms: lapsed high-spenders, new frequent buyers, and so on.

Table 1: RFM variables and their interpretation.

Variable	Definition	Better Direction
Recency (R)	Days since the customer's last purchase	Lower is better
Frequency (F)	Number of purchases in the observation period	Higher is better
Monetary (M)	Total spend in the observation period	Higher is better

Independent sorting splits each variable into quintiles separately, creating a three-digit score (R=5, F=4, M=3) and up to 125 groups. Sequential sorting nests the splits: first by Recency, then Frequency within each Recency group, then Monetary — so Recency gets the most weight.

Decile and RFM are quick to set up and easy to use. But they only create value tiers, not full customer profiles. Two customers with similar RFM scores can behave very differently: one might buy full-price items across many categories, while another buys only discounted items in a single category. To see those differences, you need to move from value tiers to richer behavioral variables.

Behavioral Clustering with K-Means

K-Means lets you cluster on multiple behavioral variables without manual cutoffs. The practical setup is straightforward: scale features with StandardScaler so dollar amounts do not overwhelm the others, and start with 6 to 10 behavioral variables. Using fewer often just recreates RFM with extra steps.

Choosing K is part statistics, part judgment. The elbow method and silhouette score help narrow the range. For most businesses, 4 to 8 segments is the practical sweet spot. If your marketing team cannot describe a different action for each segment, you probably have too many.

Numeric clustering still flattens product meaning. With inputs like frequency, spend, recency, category breadth, and discount dependency, two customers can look similar even if the products they buy are very different. To capture that meaning, bring product text directly into the segmentation.

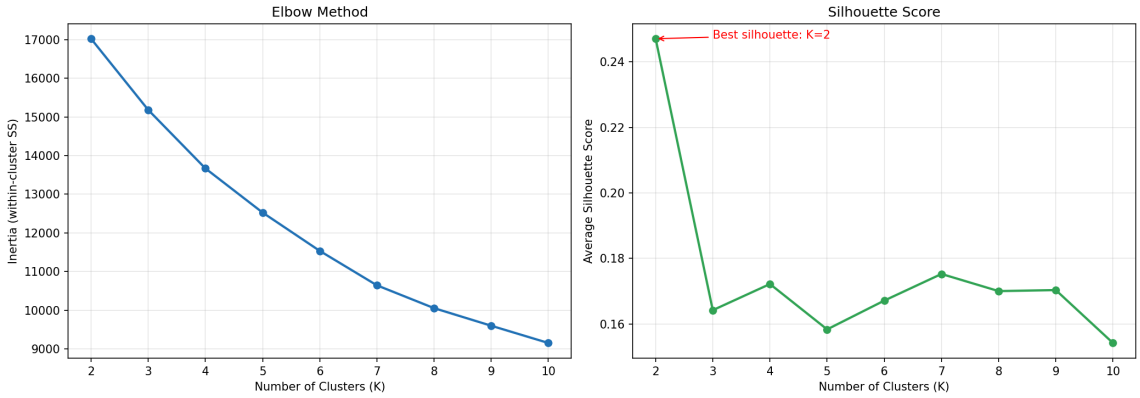


Figure 2: Elbow (left) and silhouette (right) plots; both point to K=3.

Embedding-Based Segmentation

To capture what customers buy, treat each customer's purchase history as text. Concatenate their purchased product descriptions into a single string—the customer document—and use an embedding model to turn that string into a dense vector. In this embedding space, birthday candles and party plates end up close together, while antique teapot sits far away. Cluster on these vectors and the segments split by purchase meaning, not just purchase volume.

Constructing the Customer Document

Three practical decisions shape the quality of the segments more than anything else.

Which products go in. Top-N purchased items (by frequency or revenue) usually outperform the full purchase history. The full list adds noise from one-off buys; the top items capture the customer's actual pattern.

Whether to weight by revenue. Including a high-value item once versus a low-value item once can misrepresent what the customer cares about. Revenue-weighting — repeating product mentions in proportion to spend — sharpens the segments for customers with mixed baskets. Cap the weights or use log-spend weighting when price differences across products are large; otherwise a single luxury purchase can dominate the document.

Filtering generic products. Items like "Gift Card," "Postage," or generic SKUs are noise: they appear across all segments and dilute the keywords. Filter them out by StockCode patterns or maintain a small exclusion list. This single step often makes the difference between segments that mean something and segments that mean nothing.

A lightweight sentence-embedding model is enough for most retail catalogs. The quality of your segments depends much more on how you design and validate the customer document than on which embedding model you pick.

Reading the Output

The payoff is a keyword fingerprint for each segment:

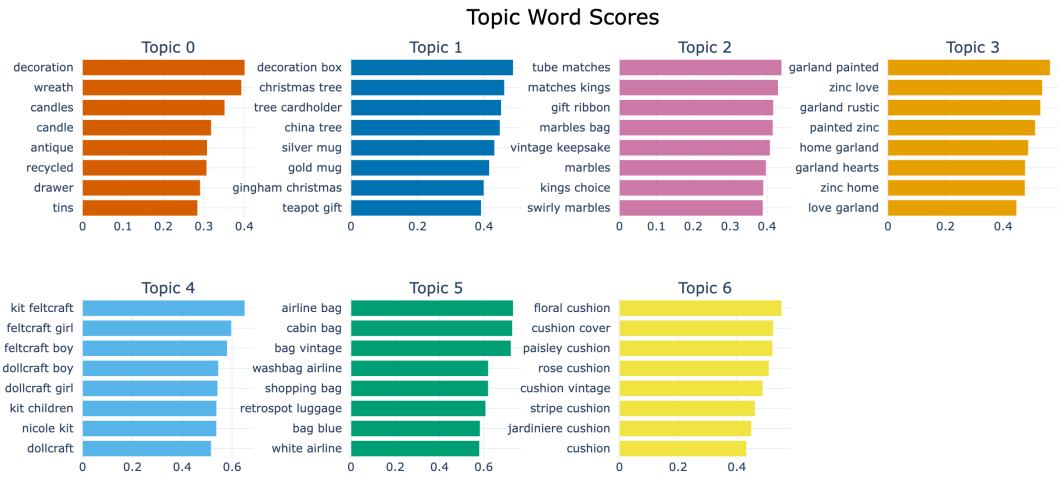


Figure 3: Top distinctive keywords per segment from product-description embeddings.

Give these panels to a marketer and they should quickly see which categories, campaigns, and creative fit each group.

When One Theme Dominates

Real catalogs often have one broad theme covering most customers and a tail of sharp niches at the edges. HDBSCAN surfaces both — but after outlier reduction the broad theme inflates into a single mega-topic that is too large to act on as one segment. The fix is surgical: run K-Means on **just the mega-topic's embeddings** and split it into activation-sized sub-segments, while keeping the small niche topics as discovery findings. This applies the "HDBSCAN for discovery, K-Means for operations" pattern inside the dominant cluster, not on the full base. The companion notebook does this automatically when any topic exceeds 50% of the base.

The LLM-Naming Trap

LLM-assisted naming can make segments feel more credible than they really are. Feed any cluster—even a random one—into an LLM and ask for a name and persona, and you will get a polished, convincing description. The narrative sounds plausible whether the cluster is meaningful or not.

Two safeguards:

- Validate the keyword evidence before reading the LLM's narrative. If you cannot see a clear theme in the top keywords yourself, the LLM's name is fiction.
- Test the name with marketing. If they would write the same campaign brief for two differently-named segments, the names are decorative, not informative.

> **Note**
>
> Companion notebooks
> The full working code is split across two notebooks, each on the dataset that suits its method:
>
> - `traditional_segmentation.py` — decile, RFM, and K-Means on the Dunnhumby "The Complete Journey" dataset. This is the same dataset the CLV chapter uses, so segment membership carries through.
> - `semantic_segmentation.py` — the embedding-based pipeline (customer-document construction with revenue weighting, generic-product filtering by StockCode, embedding, clustering, keyword extraction, and LLM-assisted naming) on the UCI Online Retail II dataset, which has the rich product descriptions this method needs.

> **Run these notebooks**
>
> - `traditional_segmentation.ipynb` — Open in Colab · nbviewer
> - `semantic_segmentation.ipynb` — Open in Colab · nbviewer. Its optional LLM-labeling step needs both the `litellm` package (`pip install litellm`) and your own `OPENAI_API_KEY` or `ANTHROPIC_API_KEY` set as an environment variable; without either, the LLM cell skips automatically and the rest of the pipeline still runs.
>
> Committed with outputs; Colab is best-effort. Full list on the Notebooks & Code page.

Is Your Segmentation Any Good?

Three checks, in priority order.

Actionability is what matters most. Show the segment profiles to your marketing team. For each pair of segments, ask: "Would you use a different message, channel, offer, or frequency?" If the answer is "I'd do the same thing for both," the segmentation is not actionable, no matter how clean the math looks. The usual culprit is a segment that is average on everything; merge it with a neighbor or revisit your variable selection.

Stability. Re-run clustering on two non-overlapping time periods and check that customers mostly land in the same segments. The Adjusted Rand Index (ARI) puts a number on this; if too many customers swap segments between periods, you cannot rely on them for campaigns. The companion notebook shows the ARI computation alongside a migration-matrix view.

Size. For broad campaign planning, each segment should usually cover at least 10 to 15 percent of the base. Niche but high-value segments can be exceptions. The practical range is 4 to 8 segments total.

Segments are built from behavioral differences, but the campaigns assigned to those segments still need to be tested. Refer back to A/B Test Design before declaring victory.

Method Cheat Sheet

Table 2: Choose the method by the question it answers, not by its sophistication.

Question	Method	Input	Output
Which customers deserve more attention right now?	Decile / RFM	Transaction log	Value tiers or R/F/M scores
What behavioral patterns exist beyond value tiers?	K-Means	Numeric behavioral features	Behavioral segment labels
What do customers actually buy?	Embedding-based clustering	Customer documents from product descriptions	Meaning-based segments with keyword fingerprints

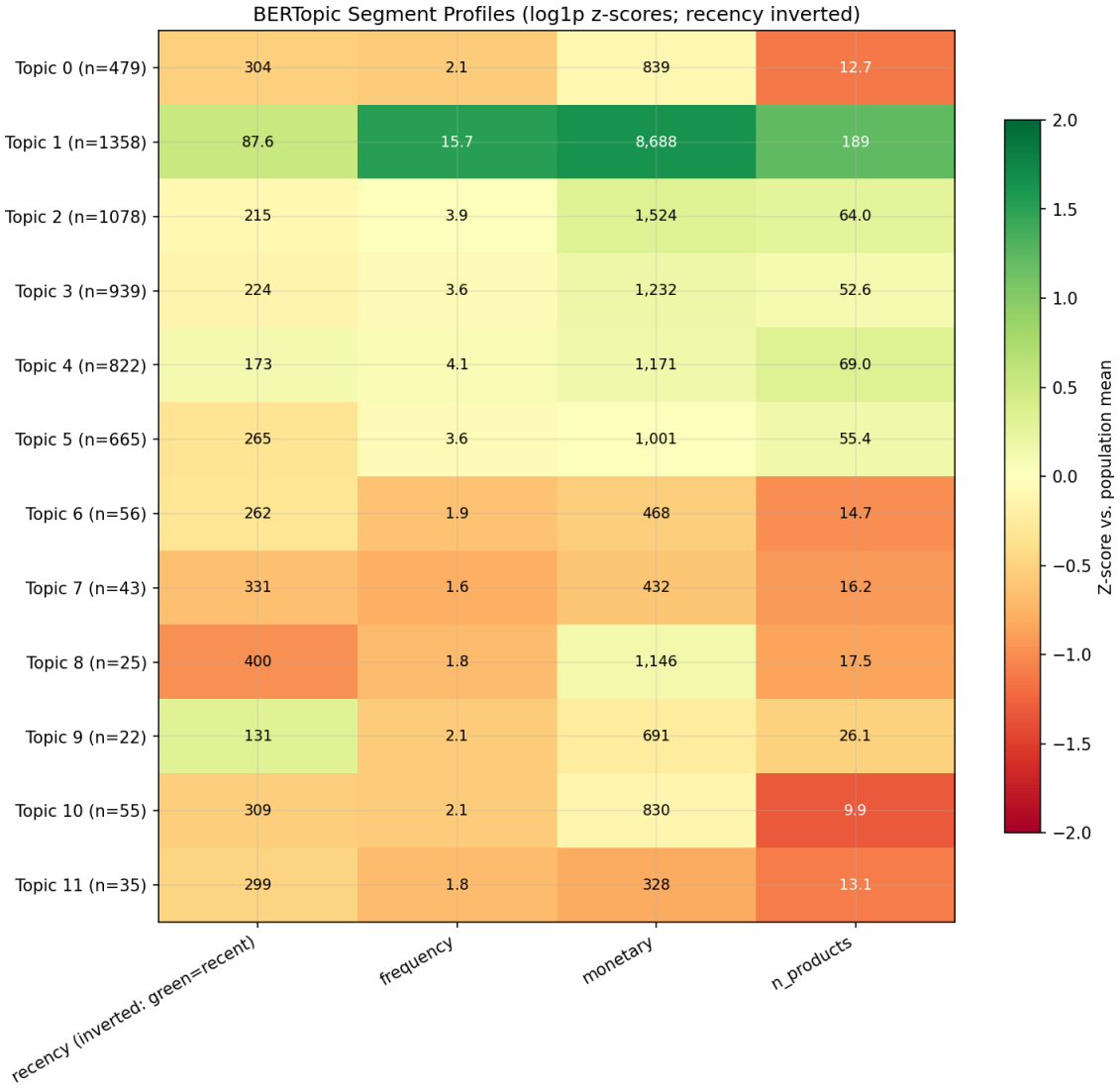


Figure 4: Segment profile heatmap: rows are segments, columns are base variables, color is deviation from the mean.

Key Takeaways

- Build segments from behavior, profile with demographics. Put behavioral variables into the algorithm; use profile variables afterward to describe, target, and reach the segments.
- Each method answers a different question. RFM ranks customers by current value. K-Means uncovers behavioral patterns beyond value tiers. Embedding-based segmentation reveals what customers actually buy. Choose the method that matches your decision.
- Validate first for actionability, then stability and size. If your marketing team would treat two segments the same way, merge them.

4.2 Customer Lifetime Value Modeling

Most marketing teams focus on Cost Per Acquisition (CPA): lower CPA means a better campaign. Customer Lifetime Value (CLV), often called LTV in analytics and SaaS contexts, reframes the question. Instead of asking how cheaply you can acquire a customer, it asks how much each customer is worth over time, and uses that answer to set the CPA ceiling.

This chapter shows how to predict CLV at the individual level using probabilistic BTYD models (BG/NBD + Gamma-Gamma with PyMC-Marketing), and how to turn those predictions into budget decisions.

Why CLV Changes the Decision

Consider a fashion e-commerce brand acquiring customers through ads and coupons. Say two segments dominate the funnel: college students and working professionals. The first purchase looks like this:

- College student: \$10 acquisition cost, \$100 purchase → 10× ROAS.
- Working professional: \$20 acquisition cost, \$100 purchase → 5× ROAS.

On day-one ROAS, the choice looks obvious: put more budget into students. But the long-run picture flips it. College students churn after the first transaction, so their lifetime spend stays near \$100. Working professionals return and spend more over time, with an average lifetime spend of \$400. Suddenly, the "expensive" segment is actually the bargain.

This is the gap CLV modeling closes. Without it, every customer gets the same CPA ceiling — which means overpaying for low-value customers and underpaying for high-value ones. With it, the ceiling scales to the predicted value, so the same budget acquires more of the right customers.

> **Note**
>
> CLV across the customer lifecycle
> CLV is useful at every stage: at Acquisition to spend more on high-value segments,

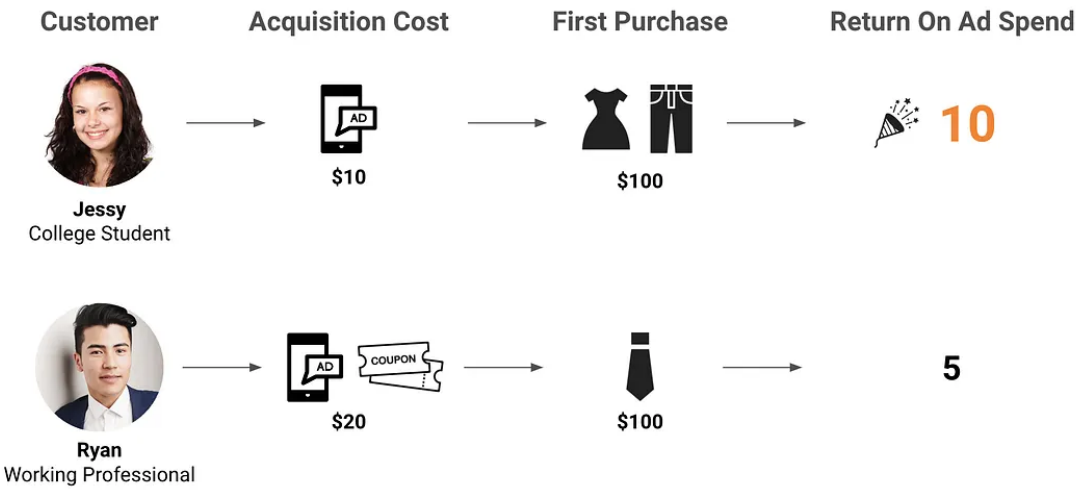


Figure 1: First purchase comparison: college students cost $10 while working professionals cost $20 to acquire, both making $100 purchases.

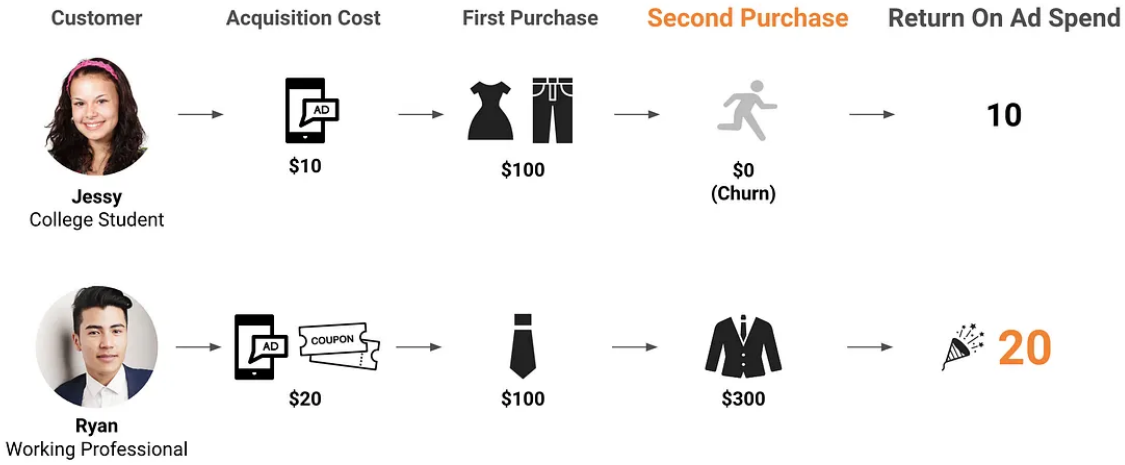


Figure 2: When repeat purchases are accounted for, the working-professional segment is far more profitable.

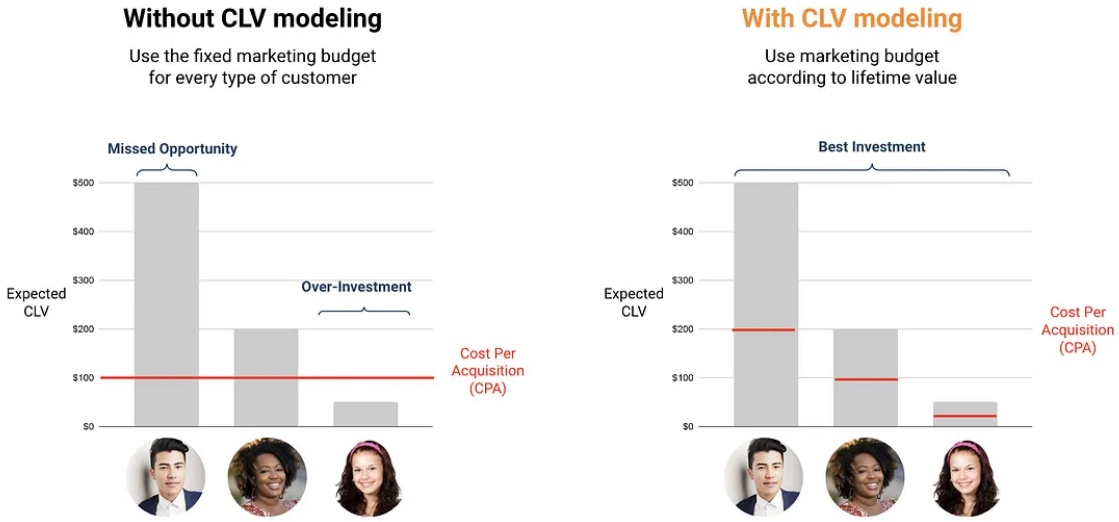


Figure 3: Fixed CPA (left) vs. CLV-scaled CPA (right).

> at Growth to prioritize service and exclusive offers for high-value customers, and at Retention to focus save-the-customer campaigns where the math actually works.

The three goals of a CLV model follow directly: (1) predict future value at the individual level, (2) identify what makes high-CLV customers different, and (3) feed both into marketing budget decisions.

Why the Traditional Formula Falls Short

The textbook CLV formula has three components:

$$\text{CLV} = \text{Average Order Value} \times \text{Purchase Frequency} \times \text{Customer Lifespan}$$

For a fashion brand where customers spend \$100 per order, shop 4 times a year, and stay loyal for 3 years: CLV $= 100 \times 4 \times 3 =$ \$1,200. Simple, intuitive, and useful for rough business sizing — but too coarse for customer-level decisions. It collapses on three points.

Limitation 1 — Not all customers are the same. Averaging across the whole base misrepresents both ends of the distribution:

Figure 4: Three limitations of the traditional CLV formula: heterogeneity, tenure, dead-or-alive.

Table 1: The average CLV of $1,090 misrepresents both groups.

	# of Customers	Average CLV	Total Sales
Normal customers	90	$100	$9,000
Big-shop customers	10	$10,000	$100,000
Total	**100**	**$1,090**	**$109,000**

The $1,090 average exists in the data and describes no one. You need CLV at the individual level.

Limitation 2 — Different first-purchase timing. A customer who started a year ago has a full year of observable data; one who started three months ago has much less. Raw frequency counts conflate "low engagement" with "new arrival" unless you adjust for tenure.

Figure 5: Tenure bias: customers who started later have shorter observable windows.

Limitation 3 — Dead or alive? In subscriptions, churn is observable: the customer cancels. In retail and e-commerce, it is not. Whether a customer counts as "alive" depends on their own pattern. Someone who buys monthly going silent for three months is a red flag; someone who buys every six months skipping three is normal.

All three problems have the same root: a single average cannot represent a heterogeneous customer base. The fix is to model CLV at the individual level — which is exactly what BTYD does.

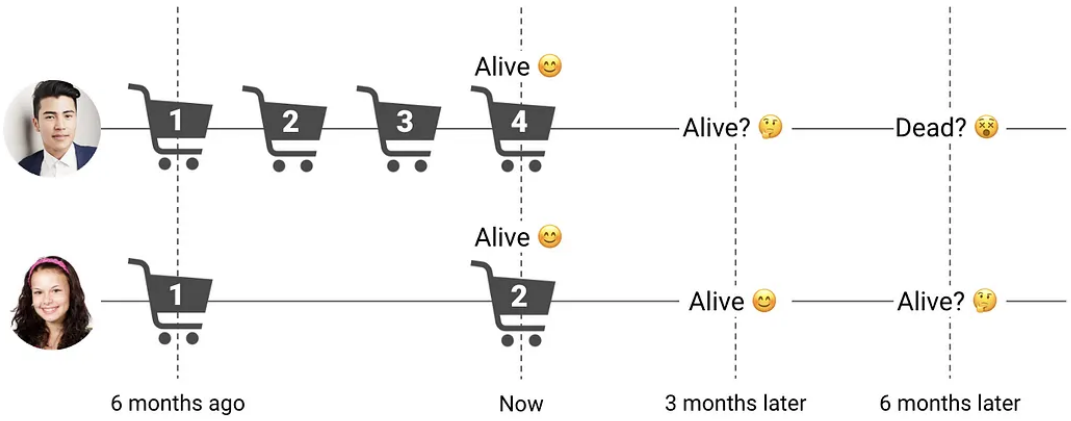


Figure 6: Dead-or-alive ambiguity: activity depends on each customer's purchase pattern.

BTYD: The BG/NBD + Gamma-Gamma Framework

Buy-Till-You-Die (BTYD) modeling is two probabilistic models stitched together:

1. BG/NBD predicts the probability a customer is still active and their future transaction count.
2. Gamma-Gamma estimates their average order value.

Multiply the two, discount over time, and you get individual-level CLV with calibrated uncertainty bounds.

BG/NBD Model

The BG/NBD model assumes two latent processes in a customer's lifecycle:

- Purchase process. While the customer is alive, purchases follow a Poisson process with rate λ.
- Dropout process. After each purchase, there is a probability p that the customer becomes permanently inactive.

Both λ and p vary across customers. A Bayesian hierarchical structure handles this: λ values follow a Gamma distribution across the base, p values follow a Beta distribution. The model learns population-level patterns while still producing individual estimates.

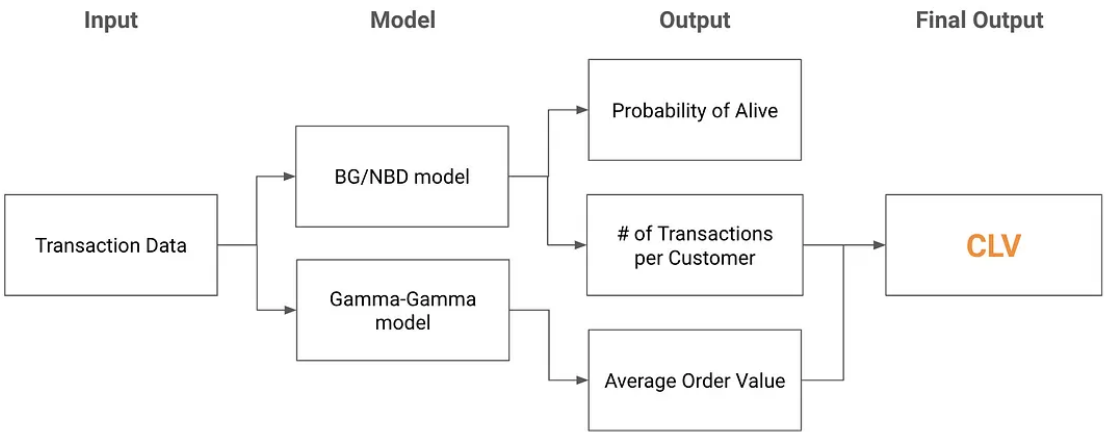


Figure 7: BTYD framework: BG/NBD (P(alive), transactions) + Gamma-Gamma (AOV) → CLV.

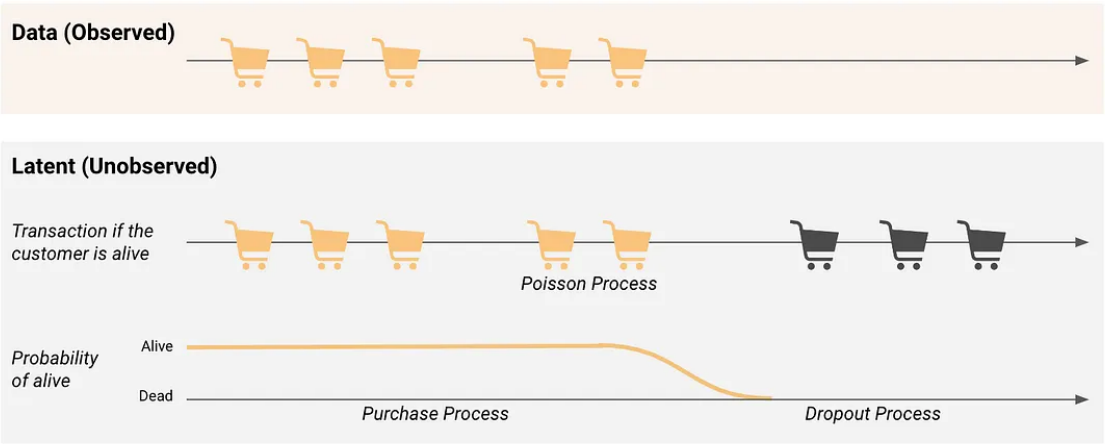


Figure 8: BG/NBD: latent Poisson purchase process plus a dropout probability.

A useful consequence: the probability a customer is alive evolves over time. It drops as the gap since the last purchase grows, then jumps back up the moment a new purchase comes in.

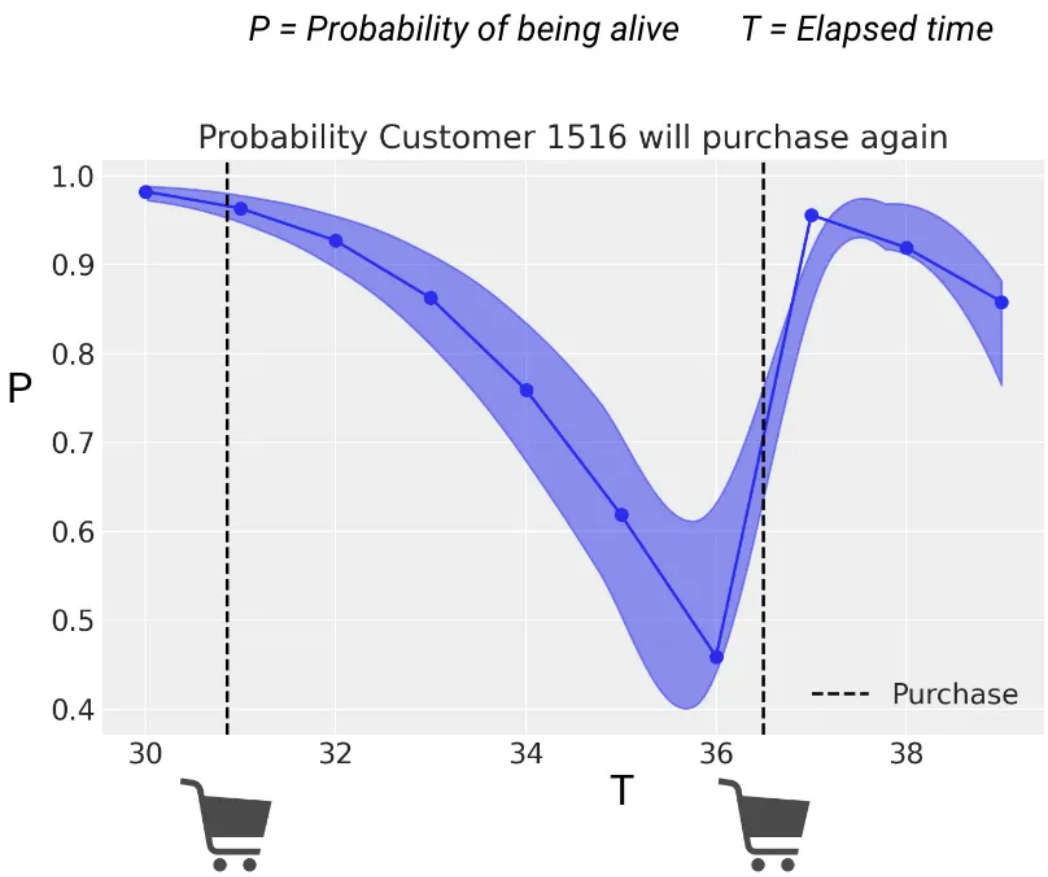


Figure 9: P(alive) decays with time since last purchase, then resets on a new purchase.

> **Tip**
>
> Further reading (BG/NBD)
> For a full derivation including the graphical model and likelihood function, see Fader et al. (2005).

Gamma-Gamma Model

The Gamma-Gamma model estimates average order value. It uses a two-layer Bayesian hierarchy: one Gamma distribution per customer (their own spending pattern), and another Gamma over the customer base (variation across customers).

> **Tip**
>
> Further reading (Gamma-Gamma)
> For a full derivation, see Fader and Hardie (2013).

> **Note**
>
> When BTYD is not the right tool
> BTYD is built for non-contractual settings (retail, e-commerce) where churn is unobservable. For contractual settings like SaaS or subscriptions, where the customer explicitly cancels, survival analysis (Kaplan-Meier, Cox Proportional Hazards) is the better fit. If you already have rich behavioral features and a production ML pipeline, an ML regression (e.g., LightGBM) on a fixed-horizon target will likely outperform BTYD — and a practical hybrid is to feed BTYD outputs (P(alive), expected purchases) into the ML model as features. The companion scripts `01_data_features.py` and `02_modeling.py` (under `notebooks/sec4.2-clv/lightgbm-companion/`) show an end-to-end LightGBM implementation.

Implementation with PyMC-Marketing

PyMC-Marketing is built on PyMC and replaces the `lifetimes` library (now in maintenance mode). The headline feature is built-in uncertainty quantification: every CLV estimate comes with HDI bounds, so you can see how confident the model is for each customer.

The CLV model needs only sales transactions: customer ID, transaction date, transaction value. Two to three years of history is usually enough.

Data and RFM-T Summary

We use the Dunnhumby "The Complete Journey" grocery-retail dataset — the same dataset used in the traditional segmentation notebook (`traditional_segmentation.py`). This lets you carry the same roughly 2,500 households from the traditional segmentation pipeline

Customer ID	Transaction Date	Transaction Value
001	2023-01-23	$40
001	2023-03-10	$60
001	2023-04-17	$30
002	2022-12-28	$100
002	2023-02-03	$200

Figure 10: Required input data for CLV modeling: customer ID, transaction date, transaction value.

into CLV; after standard cleaning (removing negative-value transactions and zero-quantity rows) and RFM-T construction, the companion notebook models 2,497 customers.

Table 2: RFM-T metrics used in the BTYD model.

Metric	Definition
Recency	Duration between a customer's first and most recent purchase
Frequency	Number of repeat purchases (total purchases minus one)
Monetary	Average value of a customer's transactions
Tenure	Duration between a customer's first purchase and the end of the study period

```
import pandas as pd
from datetime import datetime, timedelta
from pymc_marketing import clv

# Load Dunnhumby transactions and convert DAY counter to datetime
# (see companion notebook for the full pipeline)
BASE_DATE = datetime(2012, 1, 1)
data["date"] = data["DAY"].apply(lambda d: BASE_DATE + timedelta(days=int(d) - 1))
```

```
data_summary = clv.utils.rfm_summary(
    data, "CustomerID", "date", "TotalSales", time_unit="D"
)
data_summary.index = data_summary["customer_id"]
```

> **Run this notebook**
>
> Full end-to-end implementation — model fitting, CLV estimation, and demographic-segment analysis using the Dunnhumby dataset.
> PyMC_Marketing_CLV_demo.ipynb — Open in Colab · nbviewer
> Committed with outputs; Colab is best-effort. Full list on the Notebooks & Code page.

Fitting and Estimating CLV

```
# --- BG/NBD: purchase frequency + P(alive) ---
bgm = clv.BetaGeoModel(data=data_summary)
bgm.fit()

# --- Gamma-Gamma: average order value ---
nonzero = data_summary.query("frequency > 0")
gg_data = nonzero[["customer_id", "frequency", "monetary_value"]]
gg = clv.GammaGammaModel(data=gg_data)
gg.fit()

# --- Combined CLV estimate (DCF with 1% monthly discount) ---
clv_estimate = gg.expected_customer_lifetime_value(
    transaction_model=bgm,
    data=data_summary[[
        "customer_id", "frequency", "recency", "T", "monetary_value"
    ]],
    future_t=120,       # 120 months = 10 years
    discount_rate=0.01,
    time_unit="D",
)
```

The output includes not just point estimates but HDI (Highest Density Interval) bounds. In this API, `future_t` is measured in months, while `time_unit="D"` tells the model that

the RFM-T inputs are measured in days. The model tells you how confident it is for each customer.

Probability Alive Matrix

A quick way to see what the BG/NBD model has learned is the Probability Alive matrix. Recency on one axis, frequency on the other, color showing P(alive). Four corners tell four stories: new customers (low both), loyal repeaters (low recency, high frequency), at-risk (high recency, high frequency — they used to buy a lot, now they have gone quiet), and one-time passersby (high recency, low frequency).

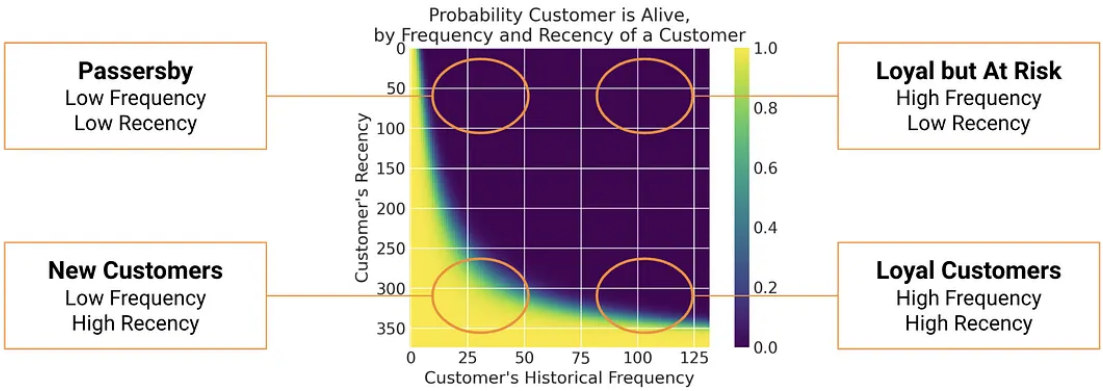


Figure 11: Probability Alive matrix by purchase frequency (x) and recency (y).

Revenue CLV → Profit CLV

> **!** Important
>
> Revenue CLV → Profit CLV
> The Gamma-Gamma model estimates revenue per transaction. To make CLV useful for budget decisions, convert revenue to profit:
>
> $$\text{CLV}_{\text{profit}} = \text{CLV}_{\text{revenue}} \times \text{Gross Margin}$$
>
> If margins vary a lot across product categories, compute customer-specific margins from purchase history instead of using a flat rate.

Combining Segments and CLV

Join the `segment_id` from the segmentation chapter (Chapter 4.1) with each customer's CLV estimate. This gives you the operating table the part has been building toward: what kind of customer this is, what they are worth, and what the business should do for them.

Demographics can still help describe or reach a segment, but the main decision layer is behavioral segment × predicted value. The companion notebook joins CLV estimates with the Dunnhumby `hh_demographic.csv` so you can profile each segment along age, income, and household composition — useful for picking creative and channel, less so for setting budget.

Putting CLV to Work

Once you have a `clv_estimate` per customer, three applications come almost for free. These patterns work for any CLV score — downstream systems only need a customer ID and a number.

1. CPA ceilings for ad bidding

Set the maximum CPA as a fraction of predicted profit CLV:

$$\text{Allowable CPA} = \text{CLV}_{\text{profit}} \times \text{Target acquisition ratio}$$

If a segment has a predicted profit CLV of $200 and your target acquisition ratio is 33%, the max CPA is $66. For bidding under uncertainty, use a conservative lower bound of predicted profit CLV rather than the mean. Many ad platforms can ingest value signals for value-based bidding or audience rules, but the exact implementation depends on the platform.

2. Churn alerts for CRM

Flag customers whose P(alive) drops more than 30 points in 30 days. The signal is the rate of change, not the absolute level. A customer whose P(alive) fell from 0.85 to 0.40 in a month is far more urgent than one who has been steady at 0.35 for six months. The first is disengaging; the second is just a low-frequency buyer.

3. Cohort forecasts for Finance

Aggregate individual CLV estimates by acquisition cohort and project revenue at 12, 24, and 36 months. Include HDI bounds so Finance knows these are probabilistic estimates, not fixed targets. This is how you answer "What is the expected return on our Q3 acquisition spend?" without making the CFO learn BG/NBD.

> **Tip**
>
> Monitoring
> Compare predicted to actual revenue per cohort at 3, 6, and 12 months. If the ratio drifts more than ±20%, retrain. Also check CLV predictions for bias against protected groups — models trained on under-served segments will perpetuate that bias by predicting lower CLV and setting lower acquisition bids.

Key Takeaways

- CLV reframes marketing budget decisions. Instead of hitting a uniform CPA target, set acquisition and retention spend in proportion to what each customer is actually worth.
- The traditional formula is too coarse. It ignores customer heterogeneity, tenure differences, and the dead-or-alive ambiguity that defines non-contractual settings.
- BTYD (BG/NBD + Gamma-Gamma) handles all three problems with transaction data alone, and returns calibrated uncertainty out of the box. For richer features or contractual settings, switch to ML regression or survival analysis respectively.
- Three immediate applications: CPA ceilings (use a conservative lower bound), P(alive) change alerts for CRM, and cohort revenue forecasts for Finance.

Part 5. Commercial Analytics

Photo by Dominik on Unsplash

Part 4 asked who your customers are and what they are worth. Part 5 turns to the operating decisions that shape the P&L: what to sell, what price to charge, and how much demand to prepare for.

Commercial analytics often goes wrong when the headline metric improves but the business value does not. A discount can boost revenue while eroding contribution margin. A new SKU can make the assortment look more complete while shifting demand away from stronger products. A forecast can look accurate overall while still leaving bestsellers stocked out and slow movers overstocked.

This part is about using analytics not just to report what happened, but to choose the better commercial decision. We will cover price elasticity and margin optimization, assortment and substitution analysis, and demand forecasting built around the cost of being wrong.

After this part, you will be able to:

- Estimate price elasticity from sales data and optimize for contribution margin
- Evaluate product assortment decisions with substitution and complementarity analysis

- Build demand forecasts for inventory, staffing, and budget decisions

5.1 Price Elasticity and Pricing Decisions

What is the right price?

The sales team wants a lower price. They are closest to the customer, and they can see the obvious effect: when price drops, units move.

The finance team wants a higher margin. Looking at the same tradeoff from the other side, they can see that revenue can go up while profit disappears if the extra volume is bought too cheaply.

Marketing sits between these views, but not by averaging them. Its job is to connect customer response, brand position, and commercial objectives: when to chase units or revenue, when to protect margin, and when price itself sends a positioning signal.

Both sides are partly right. Price affects volume, but volume is not always the main goal. Sometimes the business wants revenue, sometimes margin, sometimes traffic or share, or even just to clear inventory or defend a price position. The real missing piece is not the objective. It is understanding how much volume actually changes when price moves.

That is what price elasticity gives us. It does not decide the business goal. It gives the response curve that lets us compare pricing decisions against the goal, whether that goal is units, revenue, margin, or positioning.

What Is Price Elasticity?

Imagine you are deciding whether to raise the price of a top-selling cereal by 10%. Will you lose 5% of units, or 20%? That difference decides whether the price change improves sales, revenue, or margin, or whether it simply pushes shoppers away.

Price elasticity of demand measures the sensitivity of quantity sold to a change in price:

$$\varepsilon = \frac{\%\Delta \text{Quantity}}{\%\Delta \text{Price}}$$

For normal goods, elasticity is negative: raise the price, sell fewer units. For simplicity, we usually talk about the absolute value.

- $|\varepsilon| = 1.6$: a 10% price increase loses 16% of units. Demand is **elastic**. Price increases are risky because volume falls faster than price rises.
- $|\varepsilon| = 0.5$: a 10% price increase loses only 5% of units. Demand is **inelastic**. The product has pricing power, and a price increase can lift revenue. If unit costs are stable and known, it may also improve contribution margin.

The same elasticity estimate can support different pricing questions:

- **Should we move the price?** Elastic demand makes price increases risky. Inelastic demand makes them more attractive.
- **Which outcome should we optimize?** Elasticity plus price and volume lets us compare units and revenue. Add variable cost, and the same curve can be used to compare contribution margin.

Data Requirements for Price Elasticity

To estimate price elasticity, you need sales data with real price variation. Typical sources are POS scanner data, panel data, or internal online retail records. Use the effective price—revenue divided by units sold—instead of the shelf price:

$$\text{Price}_{\text{effective}} = \frac{\text{Revenue}}{\text{Units}}$$

Effective price captures discounts, multi-buy offers, and midweek price changes more reliably than the listed price.

Two design choices set the foundation: the time grain and the product grain. Weekly is usually the best time grain because daily data is often noisy, while monthly data can hide price changes inside the month. On the product side, start at the SKU level. A single SKU has one package size, one baseline price, and one demand response, so the price signal is easier to read.

Three things to check before fitting anything:

1. **Unit of analysis.** Use SKU $\times$ week by default. If SKU $\times$ week is too sparse, there are two defensible ways to aggregate. Move to SKU $\times$ month when price changes are not too frequent and monthly variation is still visible. Or move to SKU group $\times$ week by combining truly comparable SKUs, such as color variants, flavor variants, or same-price items with the same shopper role. Do not aggregate just because products share a brand.

2. **Price variation.** You need several distinct price points, no single price dominating the history, and enough time to separate price response from noise. As a rough rule, aim for at least three price points, a 10% price spread, and 26 weeks of history.
3. **Distorted demand signals.** Stockouts, unusual promotions, holidays, and one-off events can make volume move for reasons unrelated to price. Flag or remove these observations before treating the relationship as a normal price response.

Estimating the Demand Curve

The workhorse model is the constant-elasticity, or power-law, model:

$$Q = b_0 \times P^{b_1}$$

where Q is quantity, P is price, b_0 is a scale parameter, and b_1 is the elasticity. For normal goods, b_1 is negative. Taking logs gives a linear relationship, $\log Q = \log b_0 + b_1 \log P$, so b_1 is the slope in log-log space. The model assumes elasticity is constant across the price range. That is a strong assumption, but it is a reasonable starting point for many consumer goods.

Fit the curve in the original, non-log, space with `scipy.optimize.curve_fit` to avoid the retransformation bias that comes from fitting in log-space and converting back.

```
import numpy as np
from scipy.optimize import curve_fit

def power_model(price, b0, b1):
    """Q = b0 * Price^b1"""
    return b0 * np.power(price, b1)

# Initial parameter guesses
p0 = [df["units"].mean(), -1.5]

# Fit the model
popt, pcov = curve_fit(
    power_model,
    df["price"].values,
    df["units"].values,
    p0=p0,
    maxfev=10000,
```

```
)

b0, b1 = popt
print(f"Scale (b0): {b0:.1f}")
print(f"Elasticity (b1): {b1:.2f}")
```

The companion notebook also runs a two-pass estimation: fit once on all data, flag observations where actual divided by predicted units falls outside a sensible band, such as 0.3 to 3.0, and refit without them.

```
# Pass 1: fit on all data
popt_1, _ = curve_fit(power_model, prices, units, p0=p0, maxfev=10000)

# Identify outliers
predicted = power_model(prices, *popt_1)
ratio = units / predicted
mask = (ratio > 0.3) & (ratio < 3.0)

# Pass 2: refit on clean data
popt_2, pcov_2 = curve_fit(
    power_model, prices[mask], units[mask], p0=popt_1, maxfev=10000
)
```

Promotional spikes are the usual culprit. Include them blindly, and the model treats promotional volume as normal price response, biasing elasticity away from its true value.

Across the eight top-selling Fluid Milk items, elasticities range from -1.2 for gallon jugs, a staple format with lower elasticity, to -3.9 for smaller formats, which are easier to substitute. Figure 2 shows the fitted curves side by side.

In practice, an elasticity in the range -0.5 to -5 with $R^2 > 0.4$ and several distinct price points is usable for pricing decisions. Estimates outside that range are often data problems: too few prices, a single promotional discount dominating the fit, or too short a window. If the elasticity is not believable, neither is any pricing recommendation built on it.

Optimizing for Contribution Margin

Once we have a demand curve, we can ask different pricing questions.

The first is which price maximizes revenue:

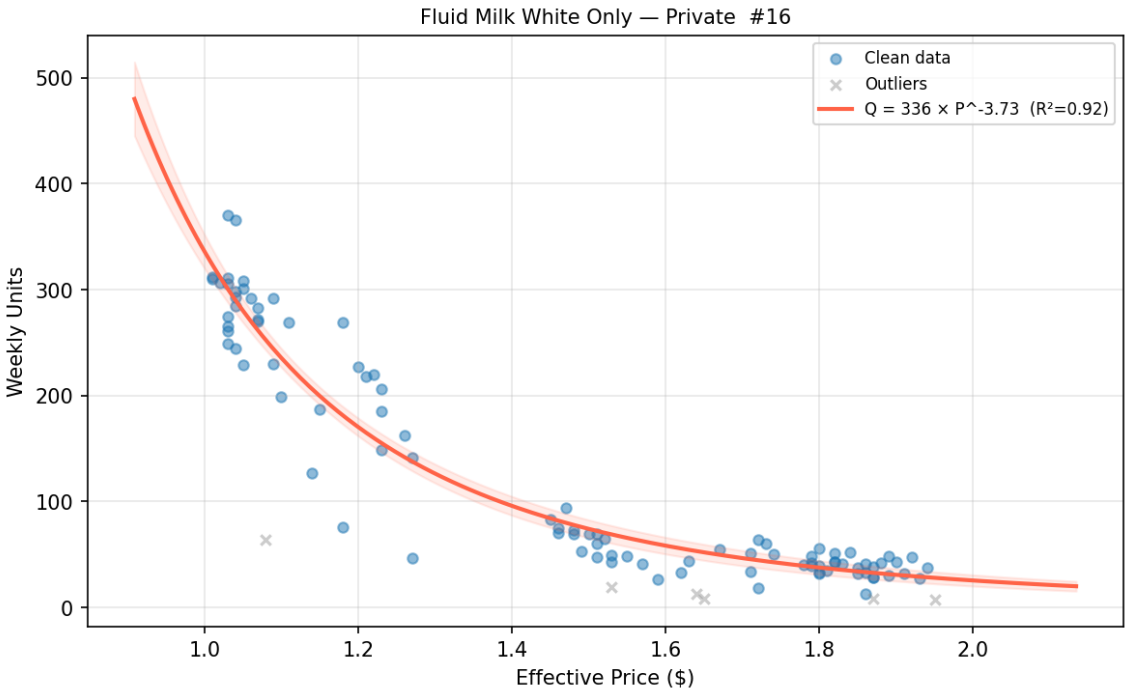


Figure 1: Price–quantity scatter and fitted power curve for a top-selling Fluid Milk SKU (Dunnhumby).

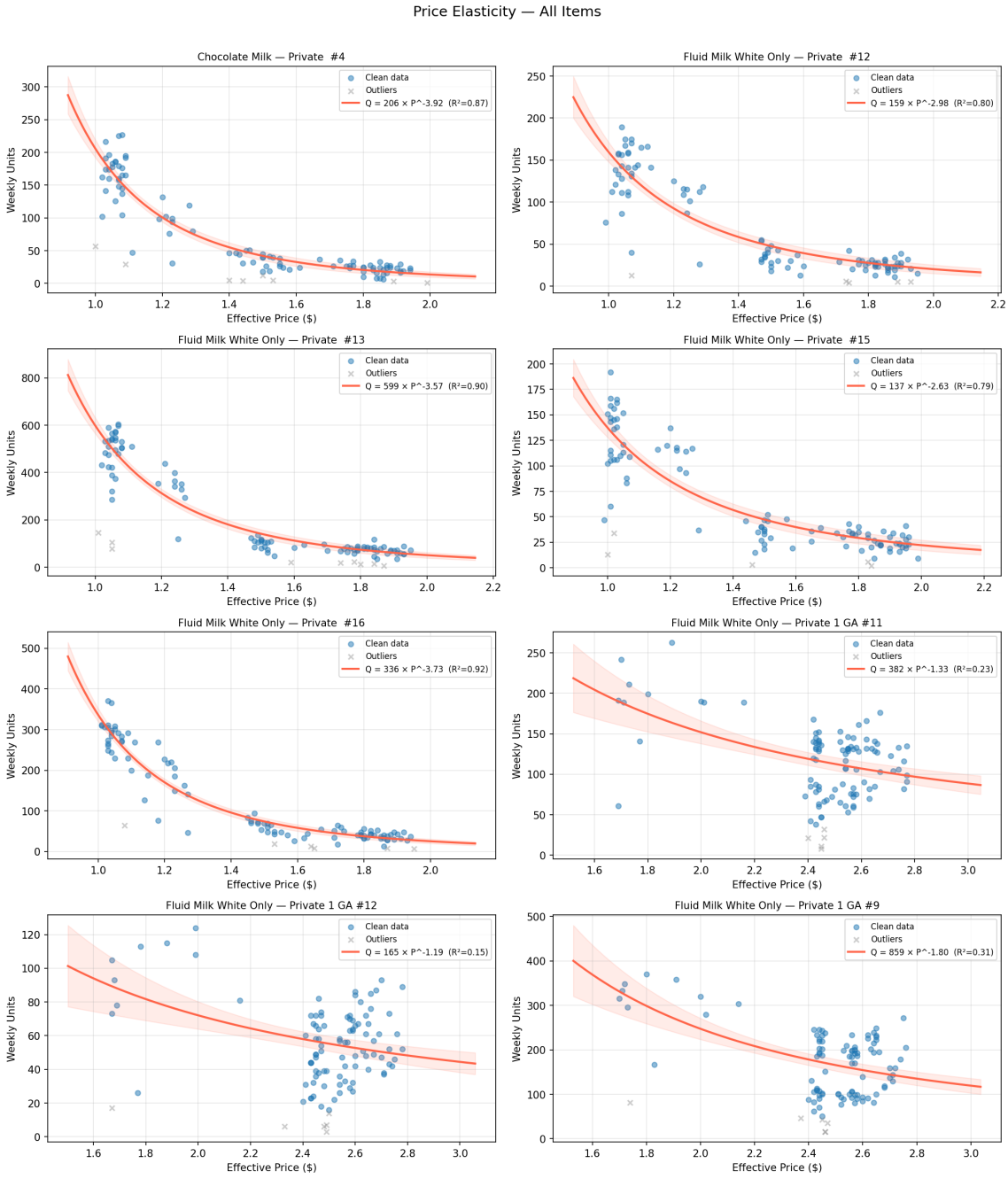


Figure 2: Fitted power-law demand curves for the eight top-selling Fluid Milk products (Dunnhumby).

$$\text{Revenue}(P) = P \times Q(P)$$

This is often the most intuitive starting point because it requires only price and units. If the business does not have reliable variable cost by product or category, revenue may be the practical metric for a first-pass pricing analysis.

But revenue is not profit. If variable cost is available, we can ask the stronger question: which price maximizes contribution margin?

$$\text{Contribution Margin}(P) = (P - C) \times Q(P)$$

where C is variable cost per unit and $Q(P) = b_0 \times P^{b_1}$.

The two answers can differ. A lower price can sell more units and increase revenue while reducing total margin. A higher price can reduce revenue but improve margin if the remaining units are profitable enough. The point is not that margin is always the only objective. The point is that the objective must be explicit before choosing the price.

Worked example: a private-label cereal

Take a private-label cereal, the same category as the part-introduction promotion story, with these parameters:

1. Current price: $3.99
2. Variable cost: $2.20
3. Current units: 1,500 per week
4. Estimated elasticity: $b_1 = -1.6$

Price	Units	Revenue	Margin per Unit	Total Margin
$3.59 (10% lower)	1,776	$6,376	$1.39	$2,469
$3.99 (current)	**1,500**	**$5,985**	**$1.79**	**$2,685**
$4.39 (10% higher)	1,288	$5,654	$2.19	$2,821
$4.79 (20% higher)	1,121	$5,369	$2.59	$2,903

The table shows the core tradeoff. Revenue is maximized by cutting the price: the 10% lower price has the highest revenue at $6,376. Margin is maximized by raising the price: the 20% higher price has the highest total margin at $2,903. At 10% higher, revenue falls 5.5% while margin rises 5.1%. Revenue and margin point in opposite directions. This is the discount trap from the part introduction, now with numbers attached.

Figure 3 visualizes the same logic for five real Dunnhumby products. In every case, the revenue curve and contribution margin curve peak at different prices. For these highly elastic private-label dairy items, the margin-maximizing price is consistently lower than the current price because the extra volume from a price cut is large enough to lift total margin.

The optimization itself is a one-dimensional search. `scipy.optimize.minimize_scalar` over the negative margin function does the job in a few lines, and the companion notebook walks through it.

```
from scipy.optimize import minimize_scalar

def neg_margin(price, b0, b1, variable_cost):
    """Negative contribution margin (for minimization)."""
    quantity = b0 * price ** b1
    return -(price - variable_cost) * quantity

result = minimize_scalar(
    neg_margin,
    bounds=(2.50, 6.00),
    method="bounded",
    args=(b0_fitted, b1_fitted, 2.20),
)

optimal_price = result.x
optimal_margin = -result.fun
print(f"Optimal price: ${optimal_price:.2f}")
print(f"Max weekly margin: ${optimal_margin:,.0f}")
```

The full pipeline covers data loading, quality checks, two-pass estimation, and the sensitivity table.

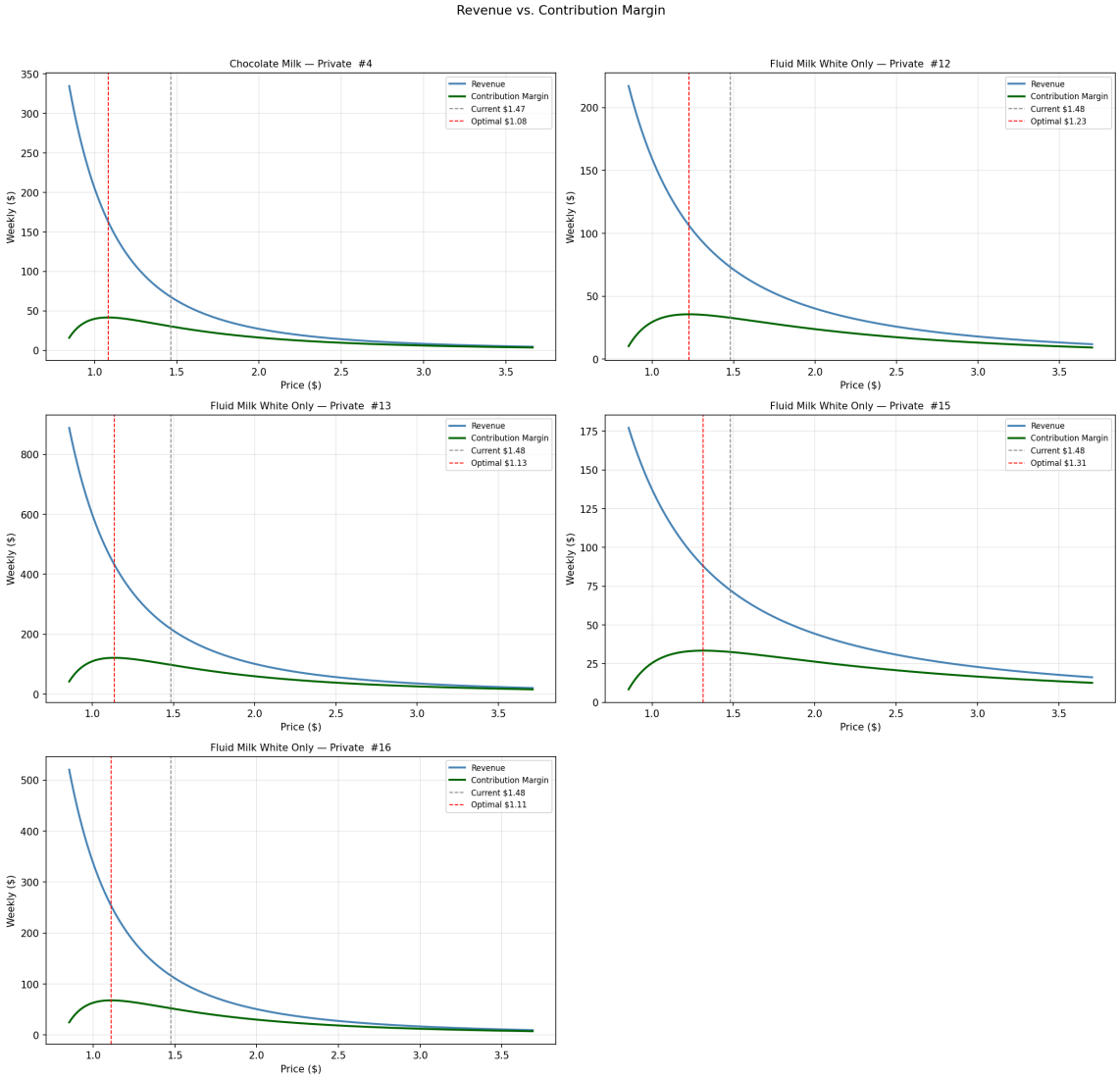


Figure 3: Revenue vs. contribution margin curves for five Fluid Milk products.

Run this notebook

`price_elasticity_scanner_data.ipynb` — Open in Colab · nbviewer
Committed with outputs; Colab is best-effort. Full list on the Notebooks & Code page.

Elasticity Varies by Product Role

The same category can contain products with very different shopper roles. In Fluid Milk Products, a gallon of white milk, a single-serve chocolate milk, and a quart of coffee creamer are not interchangeable pricing decisions. They may sit in the same broad commodity, but shoppers buy them for different occasions and compare them against different alternatives.

That is why SKU-level estimates should be aggregated only into comparable SKU groups, such as the same sub-commodity and similar package size. The goal is not to create as many segments as possible. The goal is to avoid averaging together products whose price response should be different.

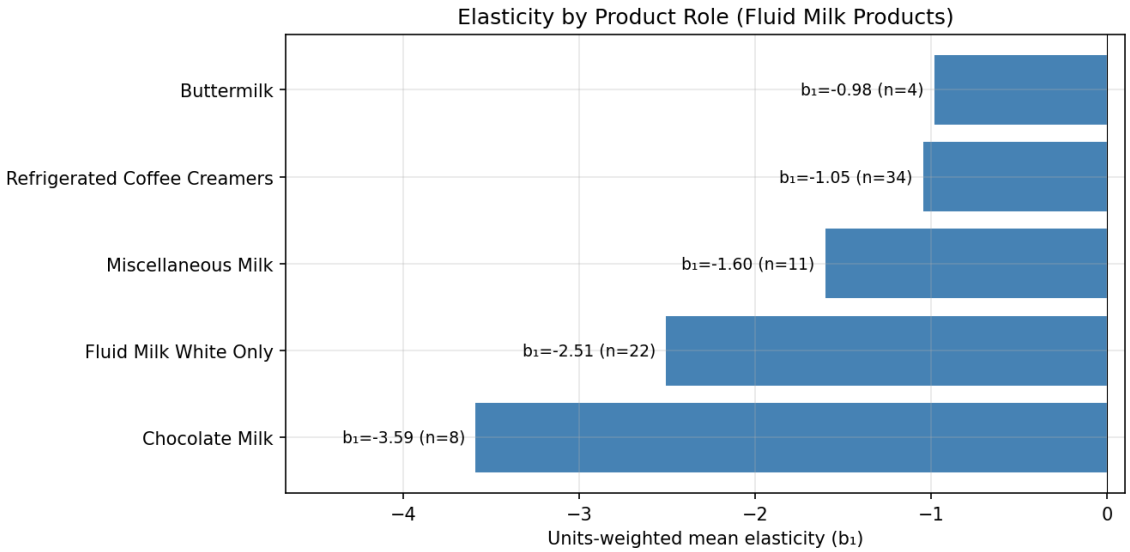


Figure 4: Units-weighted mean price elasticity by product role within Fluid Milk Products (Dunnhumby).

Before applying one pricing rule across a category, check whether major product roles

respond differently. If they do, the average elasticity is not a pricing rule. It is a warning that the category needs to be split more carefully.

Common Pitfalls in Price Elasticity

Endogeneity. Prices are not set at random. If a retailer raises prices during high-demand periods like back-to-school, the observed elasticity is biased toward zero because high prices and high demand co-occur for reasons unrelated to pricing. This is the same selection bias problem from Part 3. Address it with instrumental variables, randomized price experiments, or quasi-experimental designs.

Reference price effects. Customers remember what they paid before. Raising the price usually causes a bigger drop in volume than lowering the price brings in new buyers. The JCPenney "Fair and Square" case is the famous example: switching from promotions to everyday low prices at the same average level led to a sales collapse because customers lost their price anchor.

Competitor reaction. A price cut only works if competitors hold. In concentrated categories, competitors typically match. Everyone sells about the same volume, but at lower margins. Game theory matters more than demand curves there.

Key Takeaways

1. Price elasticity estimates how much volume changes when price moves. It can support sales, revenue, or margin decisions, depending on the business objective.
2. Revenue-maximizing and margin-maximizing prices are often different. If reliable variable cost is available, use it to evaluate contribution margin. If not, be explicit that the analysis is optimizing revenue or units, not profit.
3. SKU $\times$ week is the default grain for most retail and online retail elasticity work. If the data is sparse, use SKU $\times$ month or carefully constructed SKU groups, but do not mix different price tiers or product roles just to increase sample size.
4. A single elasticity number can hide large differences across brand tiers, channels, or regions. Check whether major segments respond differently before applying one pricing rule everywhere.
5. Watch for endogeneity, reference price effects, stockouts, unusual promotions, and competitor reaction. These can make a price response curve look more reliable than it is.

5.2 Product Assortment Optimization

Shelf space in physical retail is always at a premium. An underperforming product does more than just take up room—it ties up inventory, complicates the supply chain, and makes the shelf harder for customers to shop. The real challenge is figuring out which products to keep, which to cut, and how to validate those decisions before scaling them across the business. Here, I'll walk through a practical framework for making those calls and testing them in a way that reduces risk.

The companion script `notebooks/sec5.2-assortment/assortment_optimization.py` implements every code example in this chapter using the Dunnhumby Complete Journey grocery transaction dataset.

> ⚠ Why revenue, not margin?
>
> This chapter analyses **revenue** because the public Dunnhumby Complete Journey dataset does not include product cost or COGS. The same workflow runs on **gross margin** or **contribution margin** when production teams have cost data — only the business interpretation changes, from *revenue-risk management* to *margin optimisation*. We deliberately avoid `SALES_VALUE - discounts` as a "margin proxy" because without COGS that quantity is still revenue, not margin.

The "More SKUs, More Revenue" Fallacy

A typical grocery store carries around 50,000 SKUs. Trader Joe's carries about 4,000 (roughly one-tenth) yet generates nearly twice the revenue per square foot of Whole Foods. This comparison is suggestive, not causal—Trader Joe's also relies heavily on private-label products, operates smaller stores, and follows a different location strategy, all of which contribute to its per-square-foot performance. Fewer products mean tighter inventory, simpler supply chains, and less decision fatigue for shoppers. Psychologist Barry Schwartz calls this the **paradox of choice**: past a certain point, more options don't help customers; they overwhelm them, delay purchases, and reduce satisfaction.

Most retailers end up with the opposite problem. The push to add SKUs makes sense—buyers want broad coverage, e-commerce teams want more indexed pages, and brands are always introducing new variants. The trouble is, the cost of adding an extra SKU is spread out and easy to ignore, while the pain of removing one is immediate and visible. Over time, this leads to SKU creep: more products, higher inventory costs, and a forecasting headache as the assortment fills up with slow movers. At its core, assortment optimization is not just a data exercise. It's a decision process, and the real value of data is to help teams break through organizational inertia and make better calls.

The Two-Step Framework

The framework answers two questions in sequence:

1. **ABC → candidate pool.** Which products are in the long tail by revenue, and how many shoppers ever touch them (basket reach)?
2. **Substitution check → safety assessment.** For each candidate, if we remove this, does the demand stay in our store?

If a candidate has plausible substitutes among higher-performing products, it becomes a lower-risk *test* candidate. If not, test before acting. Nothing is confirmed safe until a matched-store test validates it.

The framework produces a 2×2 of recommended actions, indexed by whether the candidate has a credible same-subcategory substitute and whether a cross-category halo signal is present:

- **Lower-risk test candidate**: low-revenue SKU with multiple A/B-rank substitutes (by revenue or by basket reach) in the same sub-category, and no strong halo signal.
- **Test carefully, halo risk**: low-revenue SKU with substitutes but also a strong cross-category halo — removal could lose basket value elsewhere.
- **Test first, no clear substitute**: low-revenue SKU with no credible same-subcategory substitute. Demand may walk out of the store.
- **Keep or deprioritize removal**: low-revenue SKU with both a halo signal *and* no substitutes — rationalise costs instead of removing.

Step 1: Revenue + basket-reach ABC, building the candidate pool

ABC analysis ranks products by contribution (revenue, units, baskets, …) and classifies them into tiers:

- **A items** (top ~20%): the vital few. They drive roughly 80% of the chosen metric. Protect, monitor, and optimise these.
- **B items** (next ~30%): the important middle. Steady contributors that need less attention.
- **C items** (bottom ~50%): the long tail. Individually small, collectively noisy. Candidates for rationalisation.

C-rank products are where you start building your candidate pool. Not every C-rank item should be removed, but any product you consider cutting should come from this group.

A simple tweak improves the standard approach: run ABC on **two** complementary axes — **revenue** (money brought in) and **basket reach** (the number of distinct baskets that contain the SKU). Revenue ABC tells you which SKUs you would lose money on if you cut them; basket-reach ABC tells you how many shoppers ever interact with each SKU. Combining the two flags SKUs that look fine by one metric but suspect by the other — for example, A-revenue premium items that only a handful of shoppers buy, or C-revenue items that many shoppers grab as a low-price impulse purchase (treat removal of the latter cautiously).

Full Implementation: revenue_basket_abc

```python
import pandas as pd

def revenue_basket_abc(sku_summary: pd.DataFrame) -> pd.DataFrame:
    """
    Assign ABC ranks on revenue AND basket reach (n_baskets).

    SKUs with zero or negative revenue / n_baskets are filtered out
    before ABC is computed - they are out of scope for assortment
    rationalisation (no sales = no assortment decision to make).

    Parameters
    ----------
    sku_summary : DataFrame with columns ['sku', 'revenue', 'n_baskets']

    Returns
    -------
    DataFrame with added columns: 'abc_revenue', 'abc_basket_reach'.
    Rows with revenue<=0 or n_baskets<=0 are NOT present in the output.
    """
    df = sku_summary.loc[
        (sku_summary['revenue'] > 0) & (sku_summary['n_baskets'] > 0)
    ].copy()

    for metric, col_name in [('revenue', 'abc_revenue'),
                             ('n_baskets', 'abc_basket_reach')]:
        sorted_df = df.sort_values(metric, ascending=False)
        cumulative = sorted_df[metric].cumsum() / sorted_df[metric].sum()
        sorted_df[col_name] = pd.cut(
            cumulative,
            bins=[0, 0.80, 0.95, 1.0],
            labels=['A', 'B', 'C'],
            include_lowest=True,
        )
        # In case pd.cut returns NaN at a boundary edge, fall back to
        # the most conservative label so the output is NaN-free.
        sorted_df[col_name] = sorted_df[col_name].astype(object).fillna('C')
        df[col_name] = sorted_df[col_name]

    return df

# Build a SKU-level summary (revenue, n_baskets per SKU)
sku_summary = (
    transactions
    .groupby('sku')
    .agg(revenue=('revenue', 'sum'), n_baskets=('basket_id', 'nunique'))
    .reset_index()
)
sku_summary = revenue_basket_abc(sku_summary)

# Candidate pool: C-rank on revenue (the long tail by money in)
candidates = sku_summary[sku_summary['abc_revenue'] == 'C']
```

Figure 1 shows the dual Pareto curves from real grocery data: ranking the ~39,000 Dunnhumby GROCERY SKUs (after filtering out 38 SKUs with no sales) by **revenue** and by **basket reach** separately. Both curves are heavily concentrated — roughly 17–20% of SKUs account for 80% of either metric — and the low-revenue pool (`abc_revenue == "C"`) holds 63% of SKUs but only 5% of revenue. A large share of those low-revenue candidates are also C-rank on basket reach, meaning few shoppers ever touch them — that secondary signal is what makes the candidate pool worth testing rather than blindly defending on volume grounds.

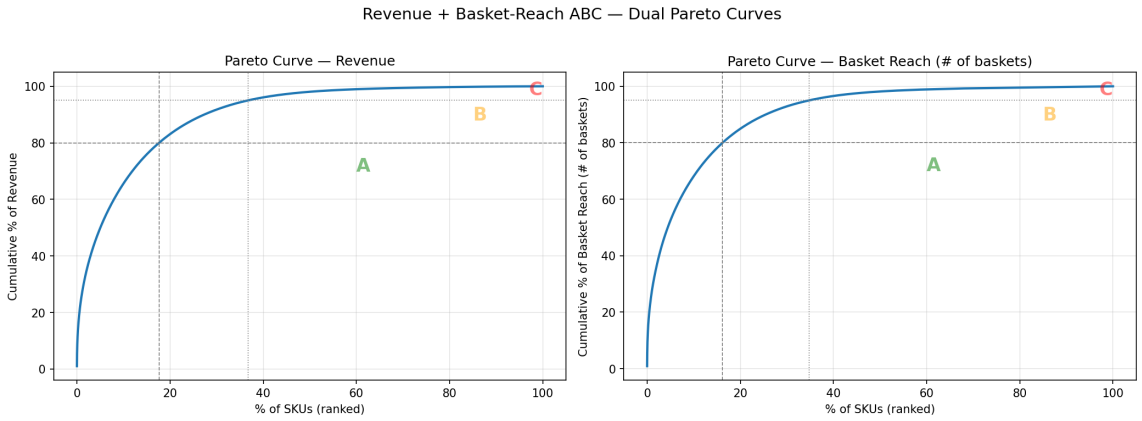


Figure 1: Cumulative revenue and basket-reach share across the Dunnhumby GROCERY SKUs.

C-rank revenue is the entrance to the candidate pool, not the exit. Step 2 determines which candidates are lower-risk test candidates and which need a more cautious test design.

Step 2: Substitution check, "Will the demand stay?"

For each C-rank candidate, the critical question is: if we remove this product, will customers buy something else from us, or will they leave the store?

If close substitutes exist among the higher-performing products in the remaining assortment, removal carries lower risk—demand is more likely to redirect rather than leave the store. But basket co-occurrence is a proxy, not proof: two products that rarely appear in the same basket might simply serve different shopping occasions or customer segments rather than being true substitutes. The matched-store test in the next section is what turns this hypothesis into evidence.

Using Lift from basket analysis. Basket analysis examines which products are purchased together. Lift measures whether two products co-occur more or less often than

expected by chance:

$$\text{Lift}(A, B) = \frac{P(A \cap B)}{P(A) \times P(B)}$$

Lift > 1 signals complementarity (bought together), Lift 1 signals independence, and Lift < 1 signals substitution (one instead of the other). Here we use Lift in reverse: Lift $<$ 1 between two products in the **same sub-category** suggests substitution, and Lift $= 0$ (zero co-occurrence) is the strongest signal. The same-subcategory constraint is critical—without it, low Lift could simply mean the products serve different needs entirely (e.g., cereal vs. pasta). Within a sub-category, low co-occurrence is more likely to reflect genuine interchangeability. Additionally, the companion notebook restricts substitute candidates to peers that are **A- or B-rank by revenue OR by basket reach**, ensuring the substitute has either enough revenue volume or enough shopper reach to absorb redirected demand.

Note that low co-occurrence is a *proxy* for substitutability, not proof. It tells us customers rarely buy both in the same trip, which is consistent with substitution but also with differences in shopping occasions or customer segments. The real test of whether demand stays in-store comes from the matched-store experiment described below.

We compute pairwise Lift directly from basket co-occurrence rather than mining frequent itemsets with FP-Growth. FP-Growth's `min_support` threshold silently drops sparse C-rank products, exactly the candidates we need to evaluate. Pairwise Lift handles sparse products naturally: zero co-occurrence is the strongest substitution signal.

Full Implementation: build_substitution_map

```
from collections import defaultdict

def build_substitution_map(
    basket_sets, candidates, subcat_map, total_baskets,
    high_rank_skus=None, lift_threshold=0.8,
):
    """
    Identify substitutes for each candidate SKU using pairwise Lift.

    Lift(A,B) = P(A   B) / (P(A) × P(B)).
    A substitute is a same-subcategory peer with Lift < threshold.
    Zero co-occurrence is treated as the strongest substitution signal.

    If `high_rank_skus` is supplied (the set of SKUs that are A/B-rank
    on revenue OR A/B-rank on basket reach), the substitute pool is
    restricted to that set - a peer needs either enough revenue volume
    or enough shopper reach to credibly absorb redirected demand.

    Parameters
    ----------
    basket_sets : dict mapping sku -> set of basket_ids
    candidates : DataFrame with column 'sku'
    subcat_map : dict mapping sku -> subcategory
    total_baskets : int, total number of baskets
    high_rank_skus : set of SKUs eligible as substitutes
                     (A/B revenue or A/B basket reach).  If None,
                     no rank filter is applied.
    lift_threshold : Lift below this indicates substitution

    Returns
    -------
    dict: {candidate_sku: [{"substitute": sku, "lift": float, "method": str}, ...]}
    """
    # Group products by sub-category
    subcat_groups = defaultdict(set)
    for sku in basket_sets:
        sc = subcat_map.get(sku, '')
        if sc:
            subcat_groups[sc].add(sku)

    candidate_ids = set(candidates['sku'])
    substitution_map = {}
```

Cross-price elasticity as reinforcement. If cross-price elasticity data is available, positive cross-price elasticity between two products confirms substitution: when product A's price goes up, product B's quantity goes up. Treat it as a bonus signal, not a requirement.

Product attribute similarity as a fallback. When basket data is sparse (new products, small stores), product attributes (same brand, same price tier, same use case) provide a reasonable proxy for substitution.

The output: a scored candidate list

Steps 1 and 2 produce a table like this:

SKU	ABC (Rev)	ABC (Basket Reach)	Substitute?	Action
Budget cereal	C	C	→ Store brand cereal (Lift 0.6)	Lower-risk test candidate
500g yogurt	C	C	→ 1kg yogurt (Lift 0.0)	Lower-risk test candidate
Older pasta sauce	C	B	→ New recipe variant (Lift 0.4)	Lower-risk test candidate
Niche imported cheese	C	C	—	Test first, no clear substitute
Organic juice (with halo)	C	A	→ Store brand juice (Lift 0.5)	Test carefully, halo risk

A Lift of 0.0 means the two products never appeared in the same basket—the strongest substitution signal available from transaction data, though not definitive on its own (it can also reflect low purchase frequency or different shopping occasions). Values between 0 and the threshold (default 0.8) indicate weaker but still suggestive substitution; higher values within that range mean the products are less likely to be interchangeable.

The decision logic: if multiple plausible substitutes exist among A/B-rank peers (by revenue or by basket reach) and no strong halo is detected, the candidate is a **lower-risk test candidate**—but still a *candidate to test*, not a confirmed removal. If no substitutes exist,

run a matched-store test before making any move. For products with no substitute and no way to test, keep them for now, but look for ways to reduce their cost—smaller pack sizes, better supplier terms, or lower safety stock. Not every low-revenue SKU belongs in the candidate pool, either. Some SKUs earn their shelf space by driving store trips, anchoring price image, or establishing category credibility—roles that ABC and Lift alone cannot capture.

One important caveat before finalising the removal list: watch for halo effects. Some low-revenue products help drive sales of other, higher-revenue items. For example, a C-revenue snack might often be bought with a premium beverage in a different category. You can spot these by looking for Lift above 1 across sub-categories—the opposite of the substitution check—anchored on A/B-revenue *or* A/B-basket-reach partners. In practice, treat this as a guardrail, not a hard rule. In dense grocery data, almost every C-rank product will co-occur with some A or B item. Set a high bar—as a rule of thumb, look for very high Lift or a large number of cross-category partners—and only flag a candidate as "test carefully, halo risk" if it clearly stands out. The exact thresholds are context-dependent; the companion notebook uses Lift > 15 or 30+ cross-category partners as starting points, but these should be calibrated to your data density.

Figure 2 is the **primary decision view**: a 2×2 of low-revenue candidates indexed by whether a credible substitute exists and whether a strong halo is detected. The four cells map directly to the four actions defined above. Figure 3 is a complementary scatter view of the same candidates plotted by revenue and basket reach — useful to inspect the within-cell distribution but **not** the decision criterion.

Testing Before Scaling

Treat the candidate list as a working hypothesis. Even with strong substitution signals, customer behavior can be unpredictable. That's why it's critical to run a matched-store test before rolling out changes across the chain.

As a starting rule of thumb, select 10 to 20 treatment stores where you remove the SKU, and 10 to 20 control stores matched on pre-period sub-category revenue, location, and customer profile. Run the test for at least 8 weeks—loyal customers might not shop every week; adjust upward for lower-frequency categories. Focus on measuring **sub-category revenue, revenue per basket, and basket count** in treatment versus control stores. Don't bother tracking the removed SKU's sales; those will be zero by definition. The right analysis here is a Difference-in-Differences (DiD) regression with store and week fixed effects, comparing changes in outcomes before and after removal between the two groups. DiD relies on the **parallel trends assumption**: absent the intervention, treatment and control stores would have followed the same trajectory. Verify this visually by plotting

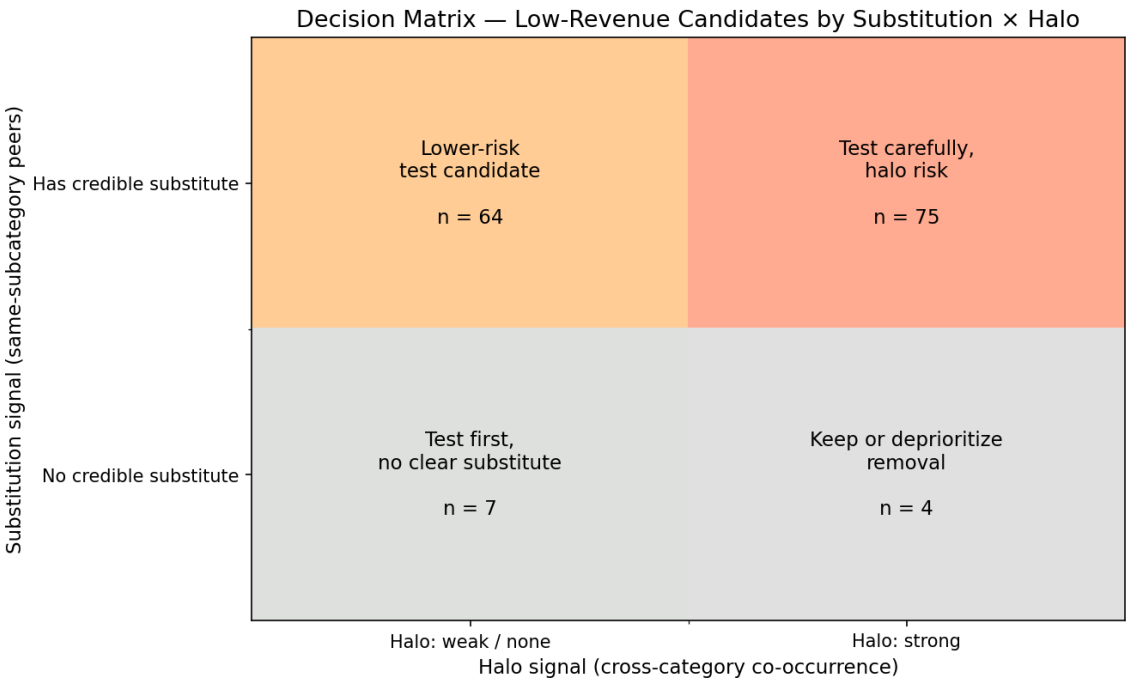


Figure 2: Decision matrix for low-revenue candidates: substitution signal × halo signal.

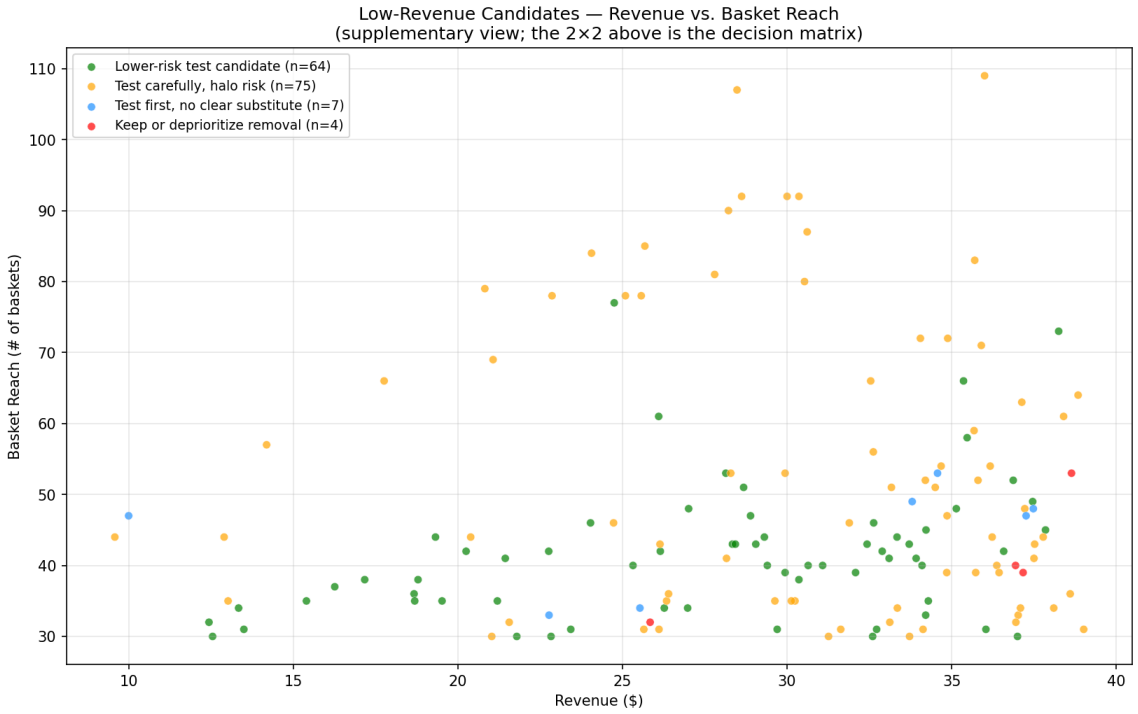


Figure 3: Low-revenue candidates by revenue and basket reach (supplementary view).

both groups in the pre-period (Figure 4, left of the intervention line). If trends diverge before the test starts, the DiD estimate is unreliable—consider re-matching stores or using a synthetic control approach instead.

If you have historical stockout data, use it to estimate substitution rates before running the test. When a product is out of stock, substitute sales usually spike, giving you a preview of how much demand is likely to shift.

Implementation: evaluate_removal_test

```python
import statsmodels.formula.api as smf

def evaluate_removal_test(test_data):
    """
    Evaluate a matched-store assortment test using DiD on revenue.

    Parameters
    ----------
    test_data : DataFrame with columns:
        - store, week, treatment, post
        - subcategory_revenue, revenue_per_basket, basket_count
    """
    metrics = [
        ('subcategory_revenue', 'Sub-category Revenue'),
        ('revenue_per_basket',  'Revenue per Basket'),
        ('basket_count',        'Sub-category Traffic (baskets)'),
    ]

    fits = {}
    print("=== Assortment Test Results (DiD) ===")

    for col, label in metrics:
        model = smf.ols(
            f'{col} ~ treatment * post + C(store) + C(week)',
            data=test_data
        ).fit(cov_type='HC1')

        did_coeff = model.params.get('treatment:post', 0)
        did_pval = model.pvalues.get('treatment:post', 1)

        fits[col] = {'coeff': did_coeff, 'p_value': did_pval}
        sig = '*' if did_pval < 0.05 else ''
        print(f"\n{label}:")
        print(f"  DiD estimate: {did_coeff:+.2f} (p={did_pval:.3f}) {sig}")

    # Decision logic on revenue (the primary metric).
    rev = fits['subcategory_revenue']
    if rev['p_value'] < 0.05 and rev['coeff'] < 0:
        print("\nRESULT: Significant revenue decline. Do not scale removal.")
    elif rev['coeff'] >= 0:
        print("\nRESULT: No revenue decline. "
              "Candidate may proceed to broader rollout or extended test.")
    else:
        print("\nRESULT: Inconclusive. Extend test period - do NOT call this a success.")

    return {
        'revenue_change':            rev['coeff'],
        'revenue_pval':              rev['p_value'],
        'revenue_per_basket_change': fits['revenue_per_basket']['coeff'],
        'traffic_change':            fits['basket_count']['coeff'],
    }
```

Figure 4 shows the output from a simulated matched-store removal test on Dunnhumby grocery data, focused on the EGGS - LARGE sub-category. Nine treatment stores had a low-revenue SKU removed (with 60% of demand redirected to substitutes); nine control stores, matched on pre-period sub-category revenue, remained unchanged. The treatment and control lines run roughly parallel in the pre-period, supporting the DiD assumption. In the post-period, treatment sub-category revenue sits modestly below control, producing a borderline-insignificant DiD estimate (-$0.42 per store-week, p 0.06). That kind of result is **inconclusive** — it is *not* a green light to scale. The decision rule in the implementation above is honest about this: an inconclusive p-value with a negative point estimate means "extend the test, expand the sample, or test a different candidate", not "ship the removal".

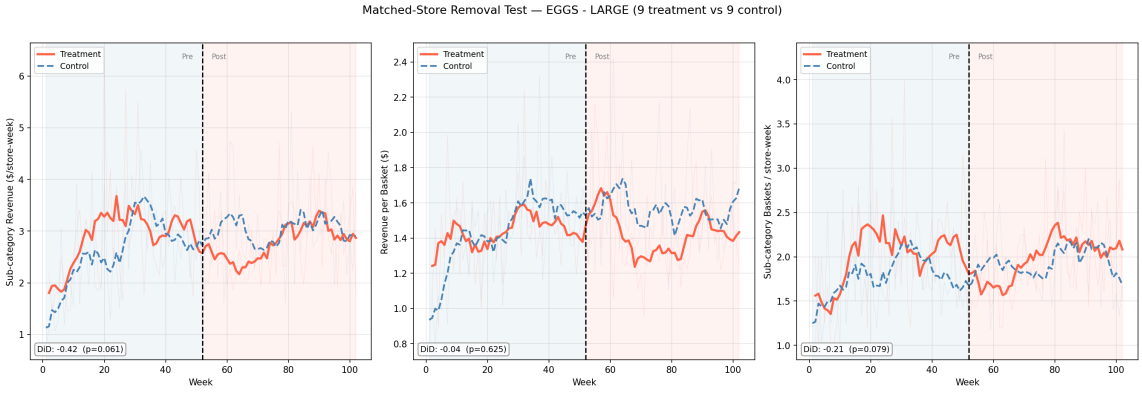


Figure 4: Matched-store removal test: parallel trends for treatment vs. control stores (EGGS - LARGE).

A few common pitfalls: exclude stockout weeks from your ABC calculations, since supply issues can look like low demand. Separate promotional sales from regular sales before ranking products. Never change price and assortment at the same time in a test. And make sure the substitute has enough safety stock to handle redirected demand.

Key Takeaways

- **Assortment optimisation here is revenue-risk management, not profit optimisation.** Without product cost in the dataset, we use revenue (and basket reach) as the candidate-pool signal. Production teams with COGS should swap revenue for **gross margin** or **contribution margin** — the same workflow runs unchanged.
- **Revenue + basket-reach ABC is the starting point, not the answer.** Low-revenue SKUs form the candidate pool; basket-reach rank is a secondary signal show-

ing how many shoppers ever touch each candidate. Only after substitution and halo checks does a candidate become testable.

- **The two-step framework (ABC then substitution) converts diagnostic metrics into testable hypotheses.** Substitutes exist among A/B-revenue or A/B-basket-reach peers → lower-risk test candidate. No substitutes → test first.
- **Halo effects are a guardrail, not a gate.** Scan for cross-category complementarity before finalising the removal list, but set a high bar. In dense basket data, weak halo signals are everywhere.
- **Never roll out without a matched-store test.** Minimum 8 weeks. Measure subcategory revenue, revenue per basket, and basket count, not the removed product's own sales. An inconclusive result is **not** a success — extend the test or test a different candidate.

5.3 Demand Forecasting

What Is Demand Forecasting?

In retail and manufacturing, demand forecasts drive real decisions. The forecast number sets the production plan, the marketing budget, the staffing schedule, and the order quantities. When it misses, the impact is immediate: too much inventory or empty shelves, wasted ad spend or missed growth.

For retailers, under-forecasting means stockouts and lost customers; over-forecasting means excess inventory and markdowns. For brands and manufacturers, the forecast also locks in raw material orders, factory schedules, and logistics—overshoot and you write off inventory, undershoot and you lose shelf space at retailers. In both cases, forecast error flows straight to the P&L.

The challenge is that every department sees the forecast differently. Sales pushes the number up to avoid missing targets; supply chain biases it down to avoid excess inventory. Even with a single forecast, each team runs its own version, and planning coherence breaks down.

Typical Approach

The standard toolkit for demand forecasting—ARIMA, Exponential Smoothing (ETS), Prophet—has a common structure: fit one model to one series, produce a point forecast, and check the error rate (usually MAPE) against a threshold (e.g. 10%). If the number looks reasonable, the job is done. This single-model, single-series workflow is where many organizations stop.

Three Limitations of the Typical Approach

This single-model workflow runs into the same problems again and again. Three specific limitations motivate the modern approach that follows.

Too many methods, no systematic way to choose. ARIMA, ETS, Prophet, LightGBM, Theta—the list of available methods and libraries keeps growing, but there is no universal winner. AutoETS might be best for one dataset, ARIMA for another, and what works in benchmarks can fall flat on your actual SKU mix. Without a structured comparison framework, most teams just stick with whatever someone set up first.

Granularity mismatch. When you add up SKU-by-store forecasts, the total often doesn't match the category-level forecast. Marketing looks at the overall brand number, operations looks at the granular weekly forecast, and the two don't line up. This is not a model failure; it is a structural consequence of forecasting each level independently.

Upside and downside risks are not symmetric. MAPE treats over-forecasting and under-forecasting the same way. But in practice, the business cost of each direction is different. For a retailer where stockouts mean lost sales and customers defecting to competitors, under-forecasting is far more expensive than holding a bit of extra inventory. On the other hand, a cash-flow-constrained manufacturer may prefer to under-forecast rather than tie up capital in unsold goods. The point is that the cost ratio depends on the business context—but symmetric metrics like MAPE ignore this entirely.

Modern Approach: Three Pillars

To address these three limitations, this chapter centers on three mechanisms: FVA, Cost-Weighted Evaluation, and Hierarchical Reconciliation. Together, they transform demand forecasting from "producing a number" into a decision-making tool connected to business value.

Forecast Value Added (FVA)

FVA asks a simple question: "Did adding this method actually improve the forecast?"

Start with the simplest baseline—Naive (just repeat the last value). Then line up every candidate method against Naive and keep only those that reduce cost. This is a structured comparison: any method that does not beat Naive on a cost basis is dropped, regardless of its sophistication.

Cost-Weighted Evaluation

Instead of MAPE, set the cost of stockouts and excess inventory separately, and evaluate forecasts based on actual business cost:

$$\text{Cost-Weighted Loss} = \begin{cases} c_u \times |e| & \text{if } y_{\text{actual}} > y_{\text{pred}} \quad \text{(under-forecast} \rightarrow \text{stockout)} \\ c_o \times |e| & \text{if } y_{\text{actual}} < y_{\text{pred}} \quad \text{(over-forecast} \rightarrow \text{excess)} \end{cases}$$

For example, if the per-unit cost of stockout is \$65 and excess inventory is \$30, the optimal forecast should target a quantile above the median:

$$\tau^* = \frac{c_u}{c_u + c_o} = \frac{65}{65 + 30} \approx 0.68$$

This tells you to target the 68th percentile—forecasting a bit high—to minimize total cost. Symmetric metrics like MAPE can't surface this business-optimal solution. In the implementation section, we target this directly using LightGBM's quantile regression objective.

Hierarchical Reconciliation

Hierarchical reconciliation mathematically aligns forecasts across all levels, so the numbers always add up. For retailers, the sum of commodity × store forecasts matches the category total exactly. For manufacturers, account-level shipment forecasts line up with the brand-level plan. The debate over which number to believe goes away.

```
Total (1 series)
   SOFT DRINKS            → Store 292, 356, 367, 381, 406  (5)
   FLUID MILK PRODUCTS    → Store 292, 356, 367, 381, 406  (5)
   BAKED BREAD/BUNS/ROLLS → Store 292, 356, 367, 381, 406  (5)
   CHEESE                 → Store 292, 356, 367, 381, 406  (5)
   BAG SNACKS             → Store 292, 356, 367, 381, 406  (5)
   YOGURT                 → Store 292, 356, 367, 381, 406  (5)
   COLD CEREAL            → Store 292, 356, 367, 381, 406  (5)

Bottom level: 35 series | Total nodes: 43
```

Two representative methods: BottomUp aggregates forecasts from the most granular level upward—simple, but noise at the bottom flows straight to the top. MinTrace combines forecasts from all levels using the error covariance matrix, often improving accuracy across the board, though the improvement depends on the quality of the estimated covariance.

The summing matrix S encodes the hierarchy. If the bottom-level forecast vector is $\hat{b}$, the bottom-up forecast for all levels is simply $S\hat{b}$.

Implementation

Now we implement the three pillars as a single pipeline. The input is a weekly sales history—one row per product-store-week combination, with columns for the series identifier (`unique_id`), date (`ds`), units sold (`y`), and any known future regressors like price or promotion flags. The output is a forecast for each series over the planning horizon (here, 12 weeks), evaluated not just on statistical accuracy but on business cost, and reconciled so that granular forecasts add up to the totals.

The pipeline has four steps. First, we fit a set of simple statistical baselines (Naive, SeasonalNaive, AutoETS, etc.) to establish the accuracy floor. Second, we add a LightGBM model that incorporates causal features like price and promotions—which the univariate baselines in Step 1 do not use. Third, we apply hierarchical reconciliation so that bottom-level forecasts are coherent with category totals. Fourth, we line everything up in an FVA table to see which steps actually added value.

The implementation uses Nixtla, an open-source forecasting ecosystem with three libraries: `statsforecast` for statistical baselines, `mlforecast` for ML models with regressors, and `hierarchicalforecast` for reconciliation. All share a common data format (`unique_id / ds / y`), making it straightforward to pass data between steps. A data-preparation companion script and one runnable companion notebook apply this pipeline to real Dunnhumby grocery panel data (7 commodity categories × 5 stores, 35 bottom-level series, 102 weeks of household panel purchases—a proxy for store-level demand, not total store sales): `01_eda.py` builds the weekly panel, and `demand_forecasting.ipynb` runs the full pipeline—modeling, evaluation, and hierarchical reconciliation.

> **Run these notebooks**
>
> - `01_eda.py` — data preparation and EDA (a companion script; requires the raw Dunnhumby CSVs and produces the committed `dunnhumby_grocery_weekly.parquet`)
> - `demand_forecasting.ipynb` — Open in Colab · nbviewer — modeling, evaluation, and hierarchical reconciliation
>
> Committed with outputs; Colab is best-effort. Full list on the Notebooks & Code page.

Step 1: Statistical Baselines (Naive → AutoETS)

Following the FVA framework, start from the simplest model and add complexity one step at a time.

```
from statsforecast import StatsForecast
from statsforecast.models import (
    Naive, SeasonalNaive, AutoETS, AutoARIMA, AutoTheta,
)
models_stat = [
    Naive(),
    SeasonalNaive(season_length=52),
    AutoETS(season_length=52),
    AutoARIMA(season_length=52),
    AutoTheta(season_length=52),
]
sf = StatsForecast(
    models=models_stat,
    freq='W-MON',
    n_jobs=-1,       # parallel across series
)
sf.fit(df_train[['unique_id', 'ds', 'y']])
fcst_stat = sf.predict(h=horizon)
```

For the Dunnhumby data, the 35 bottom-level series are each combination of commodity category (7 categories like Soft Drinks, Cheese, Cold Cereal) and store (5 stores). Each of the 5 statistical models is fit independently to each of these 35 series, producing 175 model fits in total. The companion notebook adds two intermittent demand specialists—CrostonOptimized and IMAPA—since some series have zero-sales weeks, and uses `ConformalIntervals` to produce 80% and 95% prediction intervals around the statistical baselines. The FVA table in Step 4 also includes the LightGBM models from Step 2.

Step 2: ML (LightGBM + Causal Regressors)

The statistical baselines in Step 1 are univariate—they use only past sales to predict the future. (Statistical models *can* accept exogenous regressors, e.g. ARIMAX, but the baselines here intentionally do not, to keep the FVA comparison clean.) Gradient Boosted Trees, like LightGBM, treat forecasting as a tabular regression problem and naturally accommodate additional features: price, promotions, calendar effects, and cross-series patterns like learning that a price cut at Store A should behave similarly to one at Store B.

In the Dunnhumby grocery data, three regressors are available as known future inputs: `effective_price` (average price per unit), `display_pct` (fraction of products on in-store

display), and `mailer_pct` (fraction featured in mailer/flyer). These are planned inputs—promo calendars are set 4–8 weeks ahead and price changes are scheduled—so using them is not leakage. (The general rule: any feature you use must be available at the time you produce the forecast. If it won't be, you have leakage, and your backtest will look great while production fails.)

The LightGBM model uses lags (1, 4, 12, 52 weeks), rolling statistics (4-week mean and standard deviation), calendar features (month, week-of-year), and the three causal regressors. The regressors are what make this step worth adding to the pipeline—they let the model learn the demand–price and demand–promotion relationships directly, rather than treating promo-driven spikes as unexplained noise.

Feature importance confirms what the model learned. In the Dunnhumby data, all three causal regressors appear in the top features by gain: `effective_price` (3rd), `display_pct` (5th), and `mailer_pct` (8th). Price and in-store display vary materially from week to week, giving the model strong signal; mailer features fluctuate less and rank lower. The takeaway: always check whether your regressors actually vary in the training data before assuming they will help.

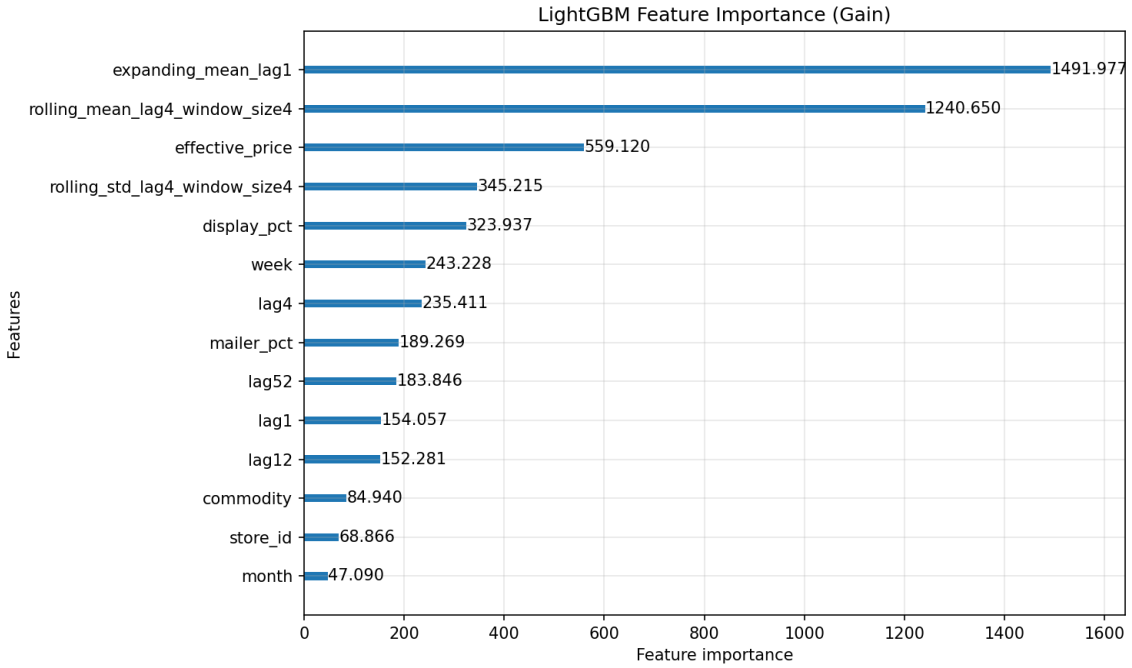


Figure 1: LightGBM feature importance (gain) for Dunnhumby grocery data.

Step 3: Hierarchical Reconciliation (MinTrace)

The hierarchicalforecast library uses tags—dictionaries that map each aggregation level to its bottom-level series—to define the hierarchy. For the Dunnhumby data, the hierarchy is three levels: Total (1) → Commodity (7 categories) → Commodity × Store (35 bottom-level series).

The workflow is: generate base forecasts at all levels with AutoETS, then apply reconciliation using BottomUp and MinTrace. MinTrace estimates the error covariance from fitted residuals, not from the raw training data. The reconciled forecasts are guaranteed to be coherent: the sum of commodity × store forecasts exactly equals the category total. If Section 's assortment rationalization removes a product, the summing matrix must be updated and reconciliation re-run. See the companion notebook `demand_forecasting.ipynb` for the full implementation.

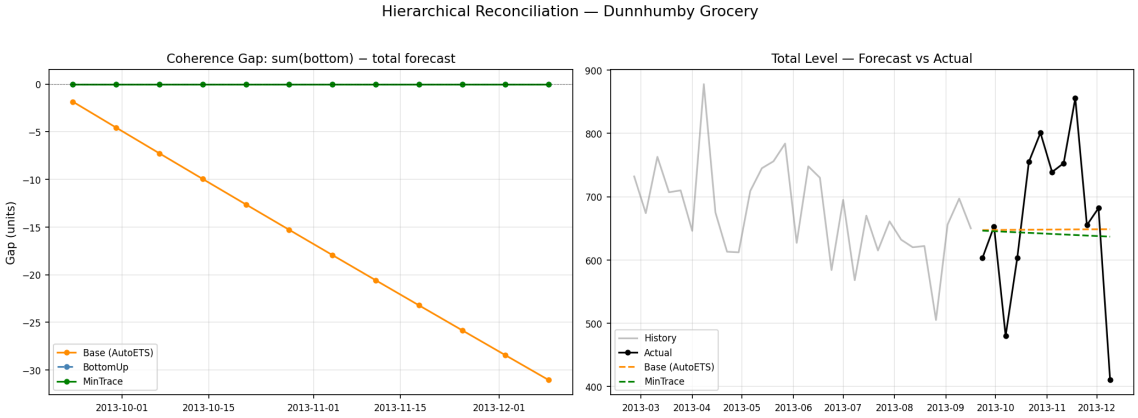


Figure 2: Hierarchical reconciliation: coherence gap (left) and total-level forecast vs. actual (right).

Step 4: FVA Table, Comparing All Stages

Now line up every step in the FVA table and verify the marginal contribution of each stage.

Every standard metric has its limits. MAPE blows up when actuals are near zero. RMSE penalizes big misses more. MASE scales each series' forecast error by its own in-sample naive error, making it dimensionless and comparable across series of different scales. It's usually the safest default for accuracy. But as argued above, symmetric metrics alone aren't enough; always include cost-weighted loss for the business-focused comparison.

Model	MASE	Cost-Weighted Loss	FVA vs Naive (CWL)
Naive	0.99	428	(baseline)
SeasonalNaive	1.16	446	-4.2%
AutoETS	0.84	367	+14.3%
AutoARIMA	0.77	326	+23.9%
AutoTheta	0.77	313	+27.0%
CrostonOptimized	0.77	331	+22.8%
IMAPA	0.77	332	+22.5%
LightGBM	0.77	368	+14.1%
LightGBM (q=0.68)	0.76	313	+27.0%

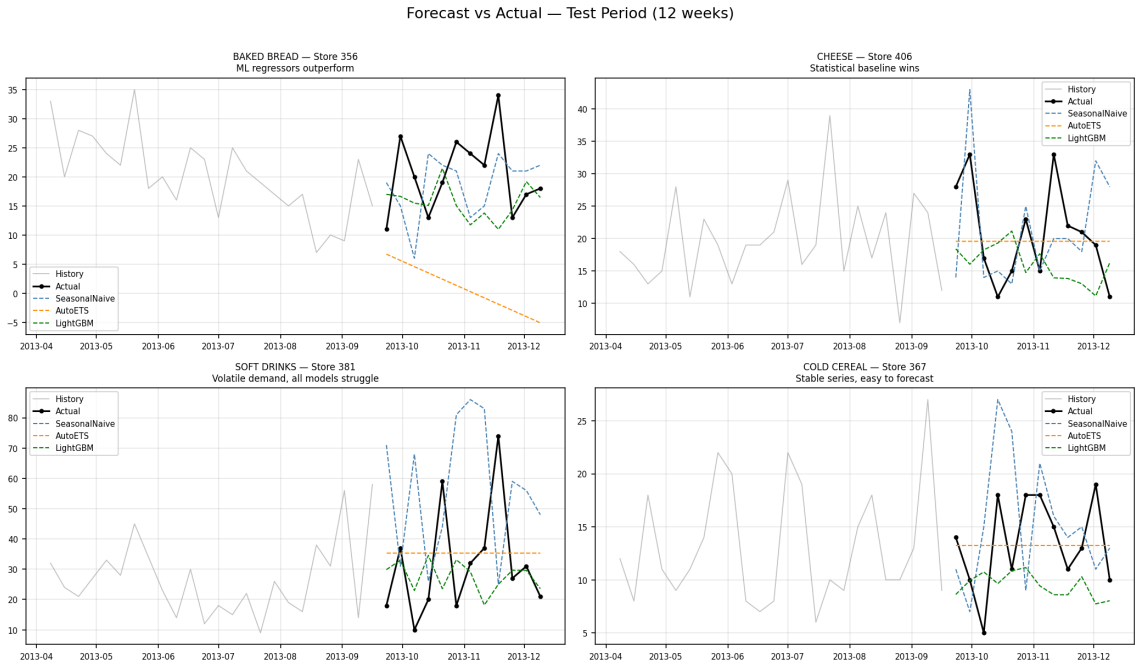


Figure 3: Forecast vs. actual for four representative series, each illustrating a different outcome.

Note that raw statistical forecasts can go negative (visible in the top-left panel); in production, the guardrails described in "From Forecast to Action" clip forecasts to zero.

The key finding is what I call the **Accuracy Trap**: LightGBM (mean) matches the best statistical models on MASE (0.77, tied with AutoARIMA / AutoTheta / CrostonOptimized / IMAPA) — yet its cost-weighted loss (368) is significantly *worse* than AutoTheta (313).

Adding ML complexity improves point accuracy slightly, but the symmetric objective misses the asymmetric cost structure. This is exactly why cost-weighted evaluation matters. This ranking depends on the assumed cost ratio (c_u/c_o); a different ratio will shift the optimal method.

The fix is quantile regression. Recall that the business-optimal target is the 68th percentile ($* = 0.68$). LightGBM can target this directly by setting `objective='quantile'` with `alpha=0.68`. The result: the quantile model forecasts 19% higher on average than the mean model, and its cost-weighted loss drops from 368 to 313—matching the best statistical model while maintaining the interpretability of ML feature importance.

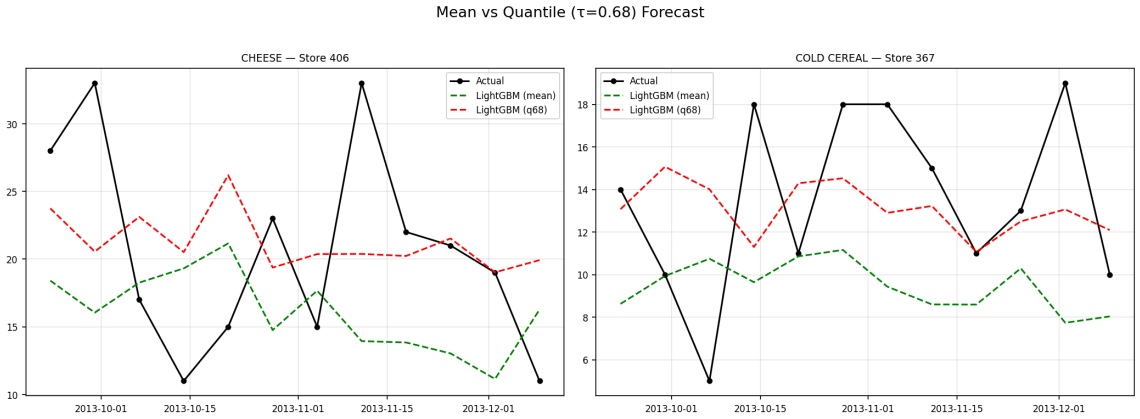


Figure 4: Quantile forecasts under asymmetric costs: optimal bias upward when stockout > overstock.

Other findings: seven of eight models improve on Naive by both MASE and CWL. SeasonalNaive is the only model with negative FVA—the data does not have strong enough year-over-year patterns at this aggregation level.

Hierarchical reconciliation does not appear in this table because its primary target is coherence—making the numbers add up across levels—not point-forecast accuracy at the bottom level. The `demand_forecasting.ipynb` notebook evaluates reconciliation separately and confirms that MinTrace-reconciled forecasts are perfectly coherent while maintaining bottom-level accuracy comparable to the base forecasts.

In practice, model selection follows a clear decision tree: if you care only about point accuracy, the statistical baselines are already strong. If you care about operating cost, quantile regression (LightGBM q=0.68) or AutoTheta is better. If you need cross-level consistency for planning, reconcile after model selection.

From Forecast to Action

A forecast that sits in a notebook is not useful. The final step is connecting the number to real decisions—and making sure it stays reliable over time.

In most organizations, the forecast feeds a recommendation (order this many units, staff this many hours), and a human approves or overrides it. This middle ground—automated recommendation, human sign-off—gets most of the efficiency while keeping judgment in the loop for edge cases. Critically, every override should be tracked and evaluated through FVA, the same way model changes are.

Once the forecast is in production, monitor two things: rolling error and bias direction. Track the 8-week moving average of absolute errors and alert if it exceeds twice the historical standard deviation. If 7 out of 8 weeks are off in the same direction, the model has drifted. For gradual drift, shorten the training window. For sudden breaks (a new competitor, a supply disruption), add a regime indicator feature and retrain.

The model predicts demand, but actions—order quantities, staffing, budgets—need business constraints. Apply these as post-processing: clip negative forecasts to zero, cap at warehouse capacity, and round to the minimum order size.

> ⚠ **Do Not Let the Model Learn Your Constraints**
>
> If past stockouts are not flagged, the model learns that demand drops to zero during those weeks—and perpetuates under-ordering. Stockout correction must happen *before* the data reaches the model: identify stockout periods (zero sales + zero inventory), replace with NaN, and impute using forward-fill or rolling average from non-stockout periods. Inventory caps and minimum order quantities are post-processing steps applied *after* the forecast.

Key Takeaways

- Demand forecasting directly drives actions. The forecast number sets order quantities, staffing, and budgets. Errors show up right away as excess inventory or empty shelves.
- Upside and downside risks are not the same. Cost-weighted evaluation, which sets stockout and overstock costs separately, ties forecast accuracy to real business value.
- Hierarchical reconciliation is essential for credibility. When granular and aggregate forecasts don't match, neither is trusted. Mathematical reconciliation removes the debate.

- FVA (Forecast Value Added) is the accountability check. Every step—including human overrides—should show positive FVA or be dropped.

Connections from this chapter

Promotional patterns—baseline, uplift, and the post-promo dip—are key inputs to the ML model's causal regressors. Price elasticity from Section and the `effective_price` feature here estimate the same underlying demand–price relationship; use price elasticity analysis for pricing strategy, this chapter for operational forecasting. When Section removes a product, update the hierarchy's summing matrix.

Part 6. Media Investment and Optimization

Photo by Andreas M on Unsplash

Large advertisers spend millions, sometimes hundreds of millions, on media every year. For a $5B brand, even a modest media budget can easily exceed $100M across TV, search, social, retail media, and direct mail.

Even a small shift in how that budget is allocated — just one or two percent — can mean millions in business impact.

This is the core question in media measurement: not which ad drove a click, but where the next dollar should actually go.

Platform ROAS reports and multi-touch attribution (MTA) help with campaign management, but they are not built for budget allocation across channels, formats, or seasons.

In this part, I'll walk through the modern measurement stack. Attribution explains what we observe in customer journeys. Media Mix Modeling (MMM) estimates how each channel contributes using aggregate data. Experiments provide the causal validation that ties it all together. When used together, these tools move media measurement from simple reporting to a system that actually supports budget decisions.

After this part, you will be able to:

- Place attribution, MMM, and experiments in the measurement stack
- Build and interpret a Media Mix Model
- Allocate media budgets using response curves and experimental calibration
- Calibrate MMM with experiments and run reliability checks before using the model for budget decisions

6.1 Mapping the Measurement Stack

Can We Cut TV Spend by 20%?

A CFO asks: "We spent millions on TV last year. If we cut that budget by 20%, would sales actually change?"

The dashboard offers ROAS by channel: paid search looks strong, retargeting looks efficient, and TV looks weak. But nobody can confidently answer the CFO's question.

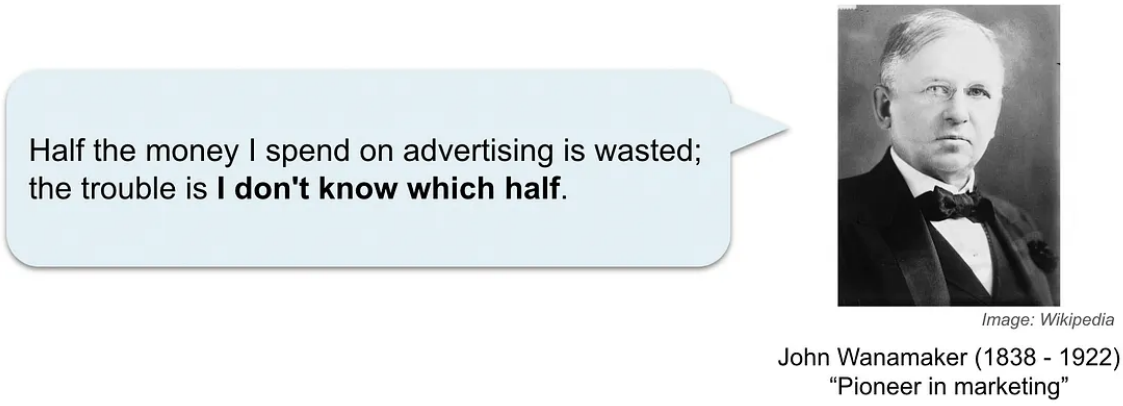


Figure 1: Wanamaker's lament on advertising waste.

That is because the dashboard is answering a different question. It can show which tracked touchpoints appeared before conversion. It cannot estimate the counterfactual: *what would have happened if TV had not run.*

TV might have created demand that later appears as branded search. If that's the case, cutting TV would not only lower TV's measured impact, but could also shrink the volume that search captures later.

This isn't just a last-click issue. Linear, time-decay, and even Google's data-driven attribution all have the same limitation: they split credit among observed touchpoints, but

they cannot estimate the counterfactual. And that's exactly what you need for budget allocation.

Attribution is not the wrong answer — it just answers a different question.

Three Questions, Three Tools

Marketing measurement has three different questions, and each needs a different tool.

- **Attribution** answers: *What happened on the path to conversion?* — the right tool for tactical questions inside a trackable channel (Creative A vs. B in Google Ads, which keyword converted, which audience responded).
- **MMM** answers: *How should we allocate budget across channels?* — the right tool when comparing TV, search, social, and offline channels that don't share a single click trail. We develop MMM in Section 6.2.
- **Experiments** answer: *Did this marketing activity actually cause incremental sales?* — the right tool for causal validation, typically via geo-lift designs. We come back to experiments in Section 6.3.

The TV example shows what happens when you try to answer a budget allocation question with a tool designed for attribution. The next section covers how MMM addresses this gap.

6.2 Building and Optimizing an MMM

What Is Media Mix Modeling?

Media Mix Modeling (MMM) is a statistical approach that uses aggregated time-series data — typically weekly sales and media spend by channel — to estimate how much each marketing channel contributes to business outcomes. Unlike user-level attribution, MMM works with aggregate data and does not require individual tracking, which makes it resilient to cookie loss, App Tracking Transparency, and walled-garden restrictions.

MMM serves three purposes:

1. **Measure**: quantify each channel's contribution and return on investment (ROI).
2. **Simulate**: answer "what if" questions — what happens to sales if we cut TV spend by 30%?
3. **Optimize**: allocate a fixed budget across channels to maximize the expected outcome.

Figure 1: The three purposes of MMM: measure, simulate, optimize.

The renewed interest in MMM is not just nostalgia. As user-level tracking becomes less reliable, aggregate modeling is now the most practical way to measure cross-channel media effectiveness.

The MMM Equation

At its simplest, MMM is a regression of weekly sales on media spend, controlling for holidays, promotions, and other non-media drivers:

$$y_t = \beta_0 + \sum_{c=1}^{C} \beta_c \cdot x_{c,t} + \sum_{k} \gamma_k \cdot z_{k,t} + \varepsilon_t$$

where y_t is sales at time t, $x_{c,t}$ is spend in channel c, $z_{k,t}$ are control variables, and ε_t is noise.

Figure 2: Simple linear MMM: sales ~ media spend + controls.

This simple regression is not enough for two reasons. First, media effects are not immediate — a TV ad this week can still influence purchases in the following weeks (carryover). Second, media effects are not linear — the hundredth GRP in a week adds less than the first (diminishing returns).

A more realistic model applies adstock (carryover) and saturation (diminishing returns) transforms before the linear combination:

$$y_t = \beta_0 + \sum_{c=1}^{C} \beta_c \cdot \text{Saturation}(\text{Adstock}(x_{c,t})) + \sum_{k} \gamma_k \cdot z_{k,t} + \varepsilon_t$$

Saturation (Diminishing Returns)

Every media channel eventually saturates. The first million dollars of TV spend drives significant incremental sales; the tenth million drives far less. The Hill function is the most common way to model this:

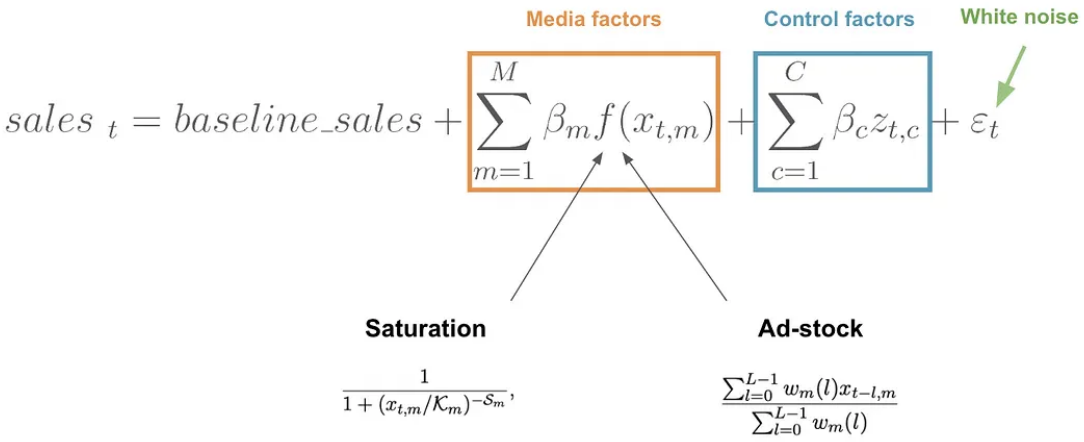


Figure 3: Full MMM: adstock and saturation transforms on media inputs.

$$\text{Hill}(x) = \frac{x^S}{x^S + K^S}$$

where K (the half-saturation point, often called EC50) is the spend level at which the channel reaches 50% of its maximum effect, and S (the slope) controls how sharply the curve bends.

The practical takeaway is clear: if Channel A is already saturated and Channel B is still on the steep part of its curve, shifting budget from A to B can increase total return — even if Channel A's average ROI looks higher.

Adstock (Carryover Effects)

Advertising does not stop working the moment it stops airing. Adstock captures this lagged effect. The simplest form is geometric decay:

$$\text{Adstock}_t = x_t + \lambda \cdot \text{Adstock}_{t-1}$$

where $\lambda \in [0, 1]$ is the retention rate. A higher λ means longer carryover. In practice, TV typically shows retention rates around 0.7–0.8 (effects linger for weeks), while digital channels like paid search show rates around 0.3–0.5 (effects decay within days).

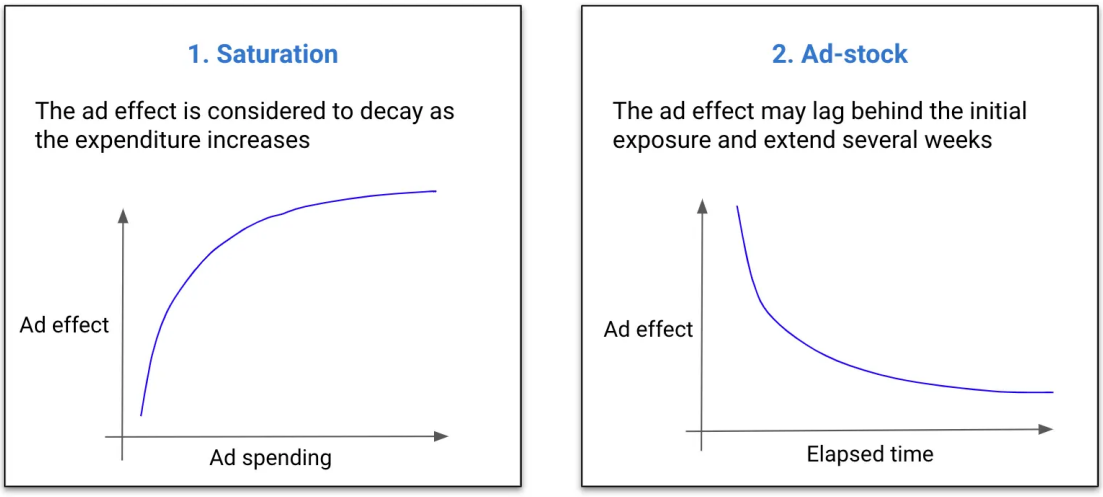


Figure 4: Saturation (left) and adstock (right) transforms.

More flexible alternatives exist. The Weibull adstock allows the decay shape to be non-monotonic: the effect can peak a few days after exposure before decaying, which better captures channels like email or direct mail where there is a natural processing delay.

Data Requirements

MMM requires three categories of input data:

1. **Target variable**: weekly revenue or sales volume at the brand or product level.
2. **Media spend**: weekly spend per channel (TV, social, display, search, direct mail, etc.).
3. **Control variables**: holidays, seasonality indicators, promotions, pricing changes, competitor activity, macroeconomic factors, or any non-media driver that might explain sales variation.

Recommended granularity:

- **Time**: Weekly. Daily data is noisier; monthly data has too few observations.
- **History**: At least 2–3 years of data (about 104 weeks). With less history, it becomes hard to separate media effects from seasonality.
- **Geography**: National or brand-level is typical. Sub-national (geo-level) models can be more powerful, but they need more data and careful modeling.

Variables type	Data category	Variables	Priority
Objective variable	KPI	Sales	Must
Explanatory variable	Media	Media spending	Must
	Other	Non-media marketing metrics (price, promotion, product distribution)	Optional
		Seasonality, holiday, weather, macro economy	Optional

Figure 5: MMM input data: target, media spend by channel, controls.

- **Channels**: Only include channels where spend actually varies. If a channel spends the same amount every week, the model cannot learn its effect.

Example MMM dataset — 157 weeks, 5 channels, controls (preview)

week_start	sales	TV/CTV	OOH	Google Search	Meta	TikTok	seasonality	christmas
2020-12-28	34,715,849	280,724	0.0	278,759	696,975	295,662	-0.039	1
2021-01-04	82,877,633	661,180	0.0	576,814	1,405,250	697,357	0.069	0
2021-01-11	83,350,211	639,517	0.0	644,509	1,215,721	582,311	0.188	0
2021-01-18	82,932,861	591,692	0.0	642,807	1,222,140	642,676	0.305	0
2021-01-25	83,627,834	558,524	0.0	630,917	1,266,757	568,506	0.417	0

Figure 6: Example MMM dataset: date, sales, spend by channel, controls.

The Bayesian Angle

Before walking through an implementation, one design choice deserves attention: modern MMM is predominantly Bayesian. The reason is practical, not philosophical.

1. **Priors encode domain knowledge.** If your team knows from past experiments that TV ROI is roughly 2.0, you can encode that as a prior. The model updates this

belief with the data. Without priors, the model might produce implausible estimates simply because of multicollinearity.

2. **Credible intervals, not point estimates.** Instead of "TV ROI = 2.3," you get "TV ROI: median 2.3, 90% CI [1.1, 3.8]." A wide interval means you should be cautious about large budget shifts based on that estimate.
3. **Natural integration with experiments.** When you run a geo-lift experiment and estimate a channel's incremental effect, you can feed that directly into the MMM as an informative prior. This tightens the posterior and improves identifiability — a workflow we develop in detail in Section 6.3.

That's why tools like Meridian and PyMC-Marketing use Bayesian inference under the hood.

A Walkthrough with Meridian

The companion notebook demonstrates a complete MMM workflow using Google's Meridian library.

> **Run this notebook**
>
> `mmm_end_to_end_demo.ipynb` — Open in Colab · nbviewer
> Committed with outputs; Colab is best-effort. Full list on the Notebooks & Code page.

> **Why Meridian?**
>
> Meridian is Google's actively maintained MMM library, the successor to the now-deprecated LightweightMMM. Other strong alternatives include Meta's Robyn and PyMC-Marketing. The core concepts (adstock, saturation, Bayesian estimation, and budget optimization) are the same across all of these tools. We use Meridian here because it offers a mature API with built-in diagnostics and optimization.

Data loading and preparation. The demo dataset contains 157 weeks of sales data with 5 media channels (TV/CTV, OOH, Google Search, Meta, TikTok) and holiday/seasonality controls. Meridian uses a `DataFrameInputDataBuilder` to construct its input data from a pandas DataFrame. Scaling is handled internally; no manual normalization is needed. The important detail is that Meridian applies saturation to `media_cols` (exposure volume such as impressions or clicks), while `media_spend_cols` are used as the cost basis for ROI.

```
channels = ["TV/CTV", "OOH", "Google Search", "Meta", "TikTok"]
media_cols = [
    "tv_ctv_impressions", "ooh_impressions", "google_clicks",
    "meta_impressions", "tiktok_impressions",
]
media_spend_cols = [
    "tv_ctv_spend", "ooh_spend", "google_spend",
    "meta_spend", "tiktok_spend",
]
control_cols = [
    "seas_sin52", "seas_cos52",
    "hldy_thanksgiving", "hldy_christmas",
    "trend", "trend_sq",
]

builder = data_frame_input_data_builder.DataFrameInputDataBuilder(
    kpi_type='revenue',
    default_kpi_column='sales',
    default_time_column='time',
    default_geo_column='geo',
)

data = (
    builder
    .with_kpi(df)
    .with_media(df,
        media_cols=media_cols,
        media_spend_cols=media_spend_cols,
        media_channels=channels)
    .with_controls(df, control_cols=control_cols)
    .build()
)
```

Model training. Meridian uses the No-U-Turn Sampler (NUTS) for MCMC. We first draw from the prior for sanity checks, then sample the posterior.

```
mmm = model.Meridian(input_data=data, model_spec=model_spec)
mmm.sample_prior(500)
mmm.sample_posterior(
    n_chains=4, n_adapt=500, n_burnin=500, n_keep=1000, seed=SEED
```

```
)
# Production runs typically use n_chains=10, n_keep=2000 (  5× compute).
```

Diagnostics. Meridian's `ModelReviewer` runs automated quality checks (convergence, baseline validation, goodness of fit, and prior-posterior shift) and reports a pass/review/fail status.

Meridian model summary — per-channel spend, ROI (with 90% CI), and incremental outcome

Channel	Spend ($M)	ROI mean	ROI 90% CI low	ROI 90% CI high	Inc. outcome ($M)
TV/CTV	116.8	3.23	0.38	9.30	378
OOH	127.3	10.19	1.81	18.18	1,298
Google Search	118.4	0.96	0.22	2.37	113
Meta	113.1	7.04	2.19	11.90	796
TikTok	139.0	3.51	0.48	8.82	488

Figure 7: Meridian model summary: per-channel spend, ROI mean with 90% credible interval, and incremental outcome.

Model fit. With a random 25% week-level holdout, the demo recovers in-sample R^2 0.95 and out-of-sample R^2 0.94. A tight in/out gap is expected here — synthetic data plus a random (interpolation) holdout is close to a best case. On production data with real noise, broken-trend periods, and lumpy promotional activity, out-of-sample R^2 in the 0.6 to 0.7 range is typical and acceptable. Trust the direction of the media estimates, not just the headline fit.

Media insights. The model gives ROI estimates with credible intervals for each channel. Wide intervals mean there is real uncertainty — this is a feature, not a bug. Point estimates alone can be misleading; credible intervals show how much confidence you should have in the results.

Budget Allocation

The model's three purposes — measure, simulate, optimize — converge here. Once the response curves are fitted, the optimizer searches for the allocation that maximizes predicted KPI under a fixed budget constraint:

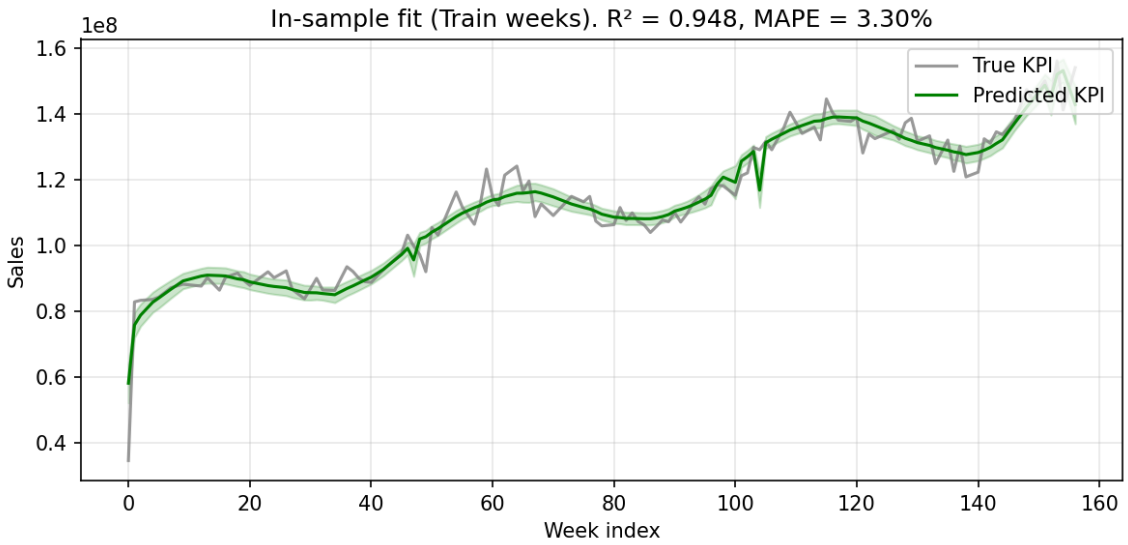


Figure 8: In-sample model fit: predicted vs. actual sales (random-holdout train period; R^2 0.95).

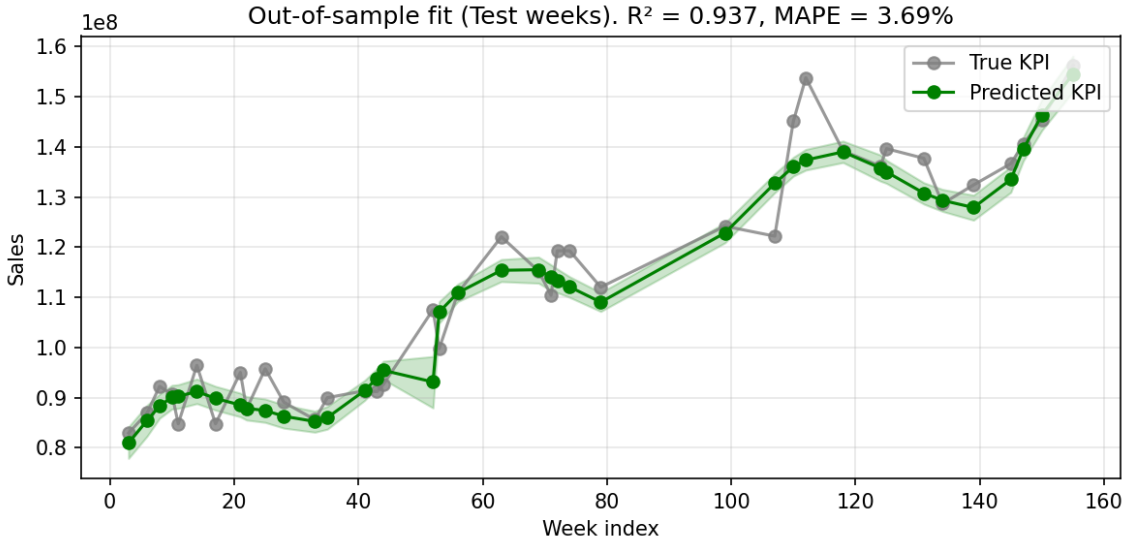


Figure 9: Out-of-sample model fit: predicted vs. actual sales on the random-holdout weeks (R^2 0.94).

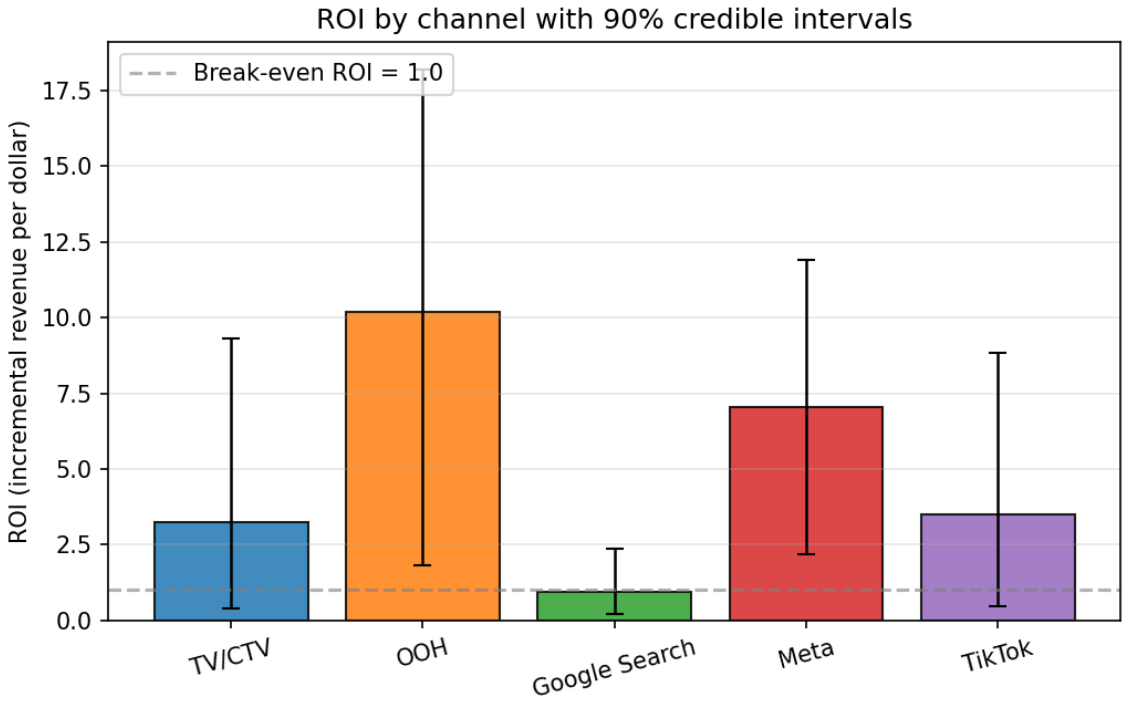


Figure 10: ROI estimates with 90% credible intervals by channel.

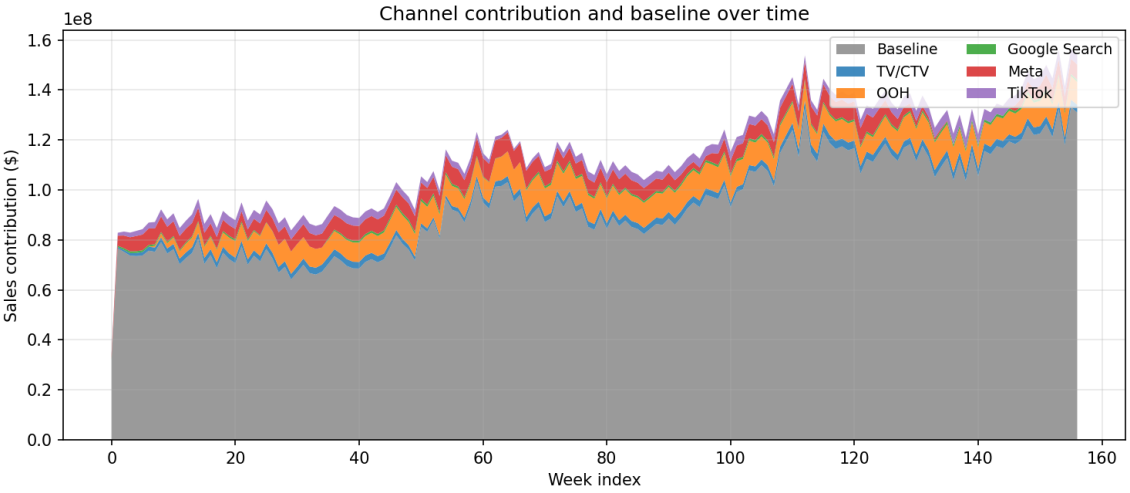


Figure 11: Channel contributions and baseline over time.

```
budget_optimizer = optimizer.BudgetOptimizer(mmm)
optimization_results = budget_optimizer.optimize()
```

The intuition is straightforward. If one channel is already saturated and another is still on the steep part of its response curve, shifting dollars from the saturated channel to the unsaturated one increases total predicted sales — even if the saturated channel's average ROI is higher.

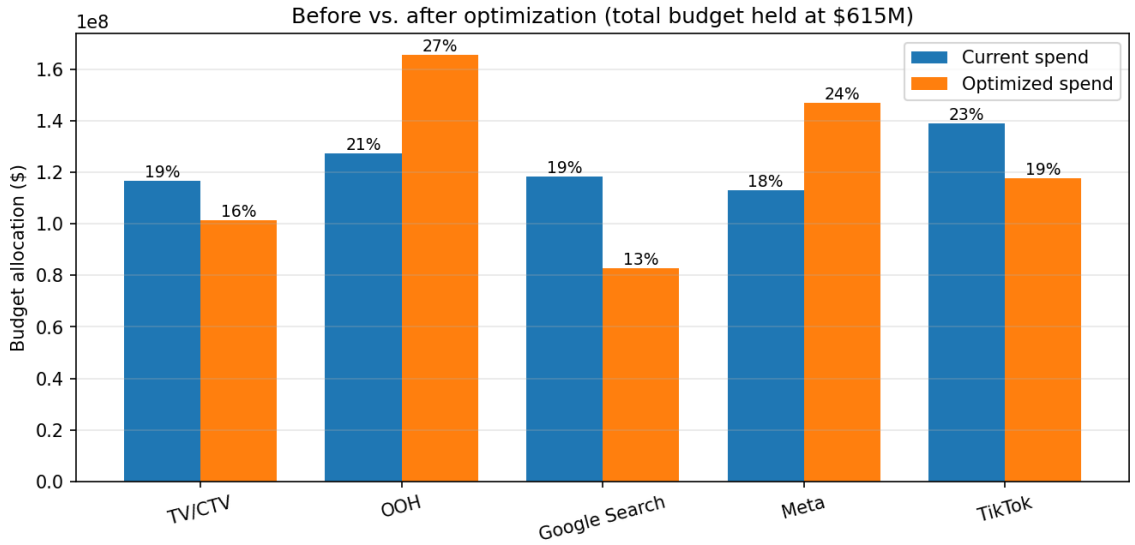


Figure 12: Pre- vs. post-optimization budget allocation and predicted sales by channel.

In the demo, the optimizer reallocates spend across the five modeled channels (TV/CTV, OOH, Google Search, Meta, and TikTok), moving budget away from channels with flatter marginal response curves toward channels with more remaining headroom. The total budget stays the same — only the mix changes. Gains of 5–10% in predicted sales from reallocation alone are common when teams first implement MMM; this demo shows about +7%.

Treat the optimizer's output as a principled starting point, not a finished media plan. The optimizer does not know about contractual commitments, minimum channel presence, operational lead times, or seasonality — those constraints need to come from outside the model. And since the response curves are based on observational data, the next step is to anchor them with experimental evidence. That's the focus of Section 6.3.

Key Takeaways

- MMM estimates channel contribution using aggregate data. It does not need user-level tracking, so it is resilient to cookie deprecation, App Tracking Transparency, and walled-garden limits.
- Hill saturation and geometric adstock are the two main transforms. Each channel gets its own set of parameters.
- **Optimize based on marginal ROI, not average ROI.** A channel with high average ROI might already be deep into its saturation curve. Shifting dollars to a channel that is still on the steep part of its curve can increase total return, even if its average ROI is lower.
- Data requirements are specific: weekly data, at least 2–3 years of history, and real spend variation for each channel.
- Bayesian estimation is now the default. Priors let you encode domain knowledge, and the posterior gives credible intervals instead of just point estimates.

6.3 Calibrating MMM with Experiments

The MMM in Section 6.2 measures, simulates, and optimizes — but its estimates are fundamentally observational. The model observes that weeks with higher TikTok spend tend to have higher sales, and attributes part of that lift to TikTok. It cannot, by itself, distinguish genuine media effects from confounders: seasonal demand that coincides with campaign flights, promotions that run alongside media pushes, or competitive dynamics that move several variables at once.

MMM also has a structural identification problem. Media channels are usually correlated — brands raise spend across all of them during the same peak periods. When Meta, Google, and TikTok spend all spike in Q4, the model cannot cleanly separate their individual effects. The result is wide credible intervals and unstable ROI estimates.

Controlled experiments close the gap. A **geo-lift test** measures the causal incremental effect of a specific channel by varying spend across test and control geos. Feeding that estimate back into the MMM as an informative prior tightens the posterior and anchors one channel's ROI in experimental evidence.

This section walks through the loop: design the experiment, estimate the lift, convert it to a prior, plug it back into the model, and run calibration as a continuous program.

Geo-Lift Tests

The most practical experiment design for MMM calibration is the **geo-lift test** (also called a matched-market test). Divide your markets into test and control groups, increase (or decrease) spend in the target channel for the test group, and measure the difference in outcomes.

Why geo-level? Because MMM operates at the geo-week level. A geo-lift test produces an estimate at the same granularity as the model's inputs, making integration natural.

A typical design involves five decisions:

1. **Target channel.** Prioritize channels with the widest credible intervals in the uncalibrated model, or channels where the optimizer recommends large shifts.

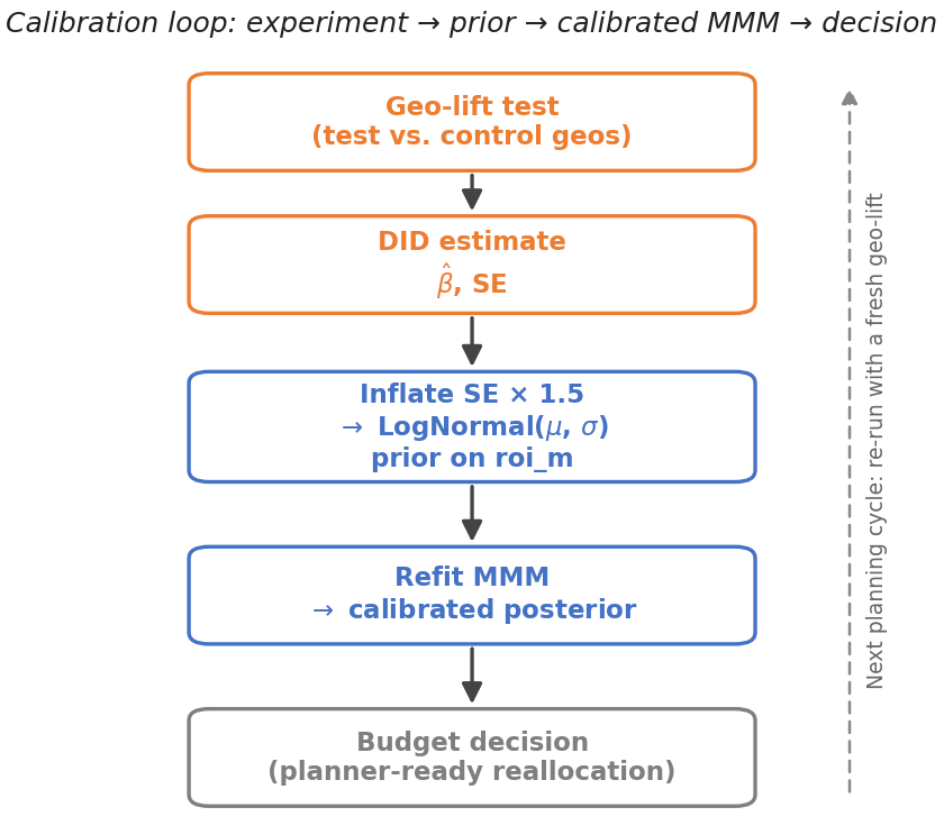


Figure 1: MMM calibration workflow: experiment → lift → prior → posterior.

2. **Test and control geos.** Split markets into groups with similar pre-period revenue trends. Parallel trends is the foundation of the DID estimator — if test geos were already growing faster before the experiment, the estimate will be biased.
3. **Intervention magnitude.** A 5% spend increase is unlikely to produce a detectable signal in noisy weekly geo data. 20–50% is more typical.
4. **Duration.** Four to eight weeks of treatment is common. Include a pre-period of similar length for the parallel trends check.
5. **Geo assignment.** Assign larger markets to the test group for more statistical power.

```yaml
experiment:
  target_channel: TikTok
  spend_multiplier: 1.2        # +20% spend in test geos
  pre_weeks: 4
  geo_assignment:
    test: [DE, FR, IT, ES]
    control: [NL, BE, AT, SE, DK]
  periods:
    experiment_start: "2025-07-07"
    experiment_end: "2025-09-29"
```

From Lift to ROAS

The standard estimator for geo-lift tests is Difference-in-Differences (DID): compare the change in revenue from pre to post in test geos against the same change in control geos. The DID coefficient $\hat{\beta}$ is average incremental revenue per geo-week. Convert to ROAS:

$$\text{ROAS}_{\text{lift}} = \frac{\hat{\beta} \times n_{\text{geos}} \times n_{\text{weeks}}}{\sum \Delta\text{spend}_{g,t}}$$

> **!** Important
>
> This is a **marginal ROAS** — the return on the incremental dollar — not the average return across all dollars spent. The distinction is critical when converting the estimate to a prior, because the MMM posterior reflects average ROAS. We unpack this in the next section before showing the integration code.

Before using the estimate for calibration, verify three things: (1) parallel trends hold in the pre-period; (2) the ROAS estimate is positive; (3) the 95% CI is narrow enough to be useful as an anchor.

Marginal vs. Average ROAS

This point deserves emphasis because it is the most common source of confusion in calibration.

The geo-lift experiment measures **marginal ROAS** — the return on the incremental dollar of spend above the baseline level. The MMM posterior for `roi_m` reflects **average ROAS** — total media-driven revenue divided by total spend over the modeling period. Because of the Hill saturation curve, these are different quantities.

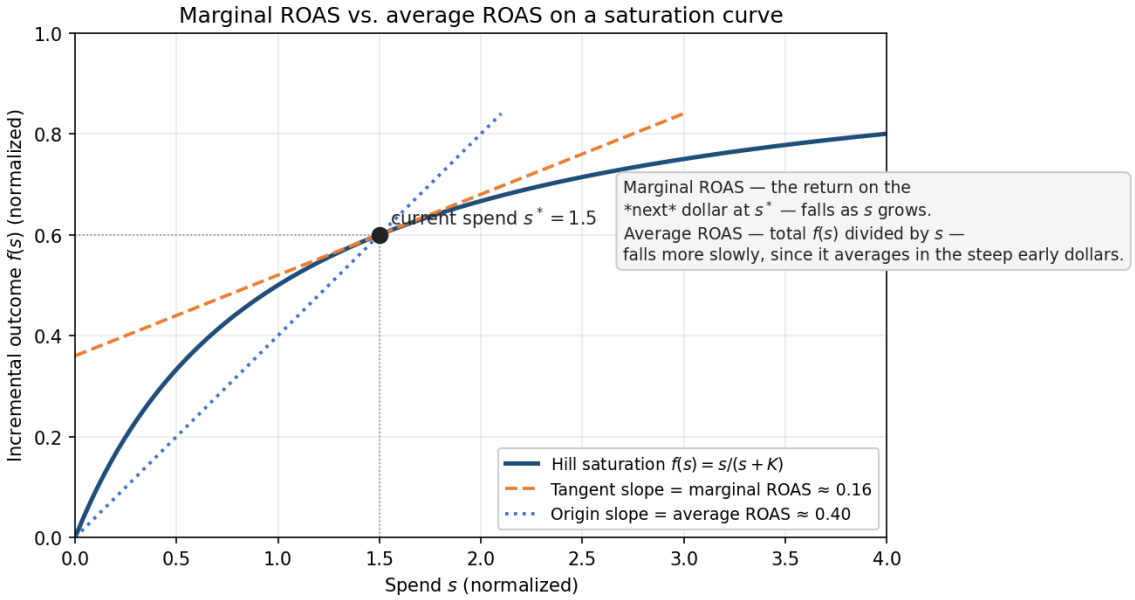


Figure 2: Marginal vs. average ROAS on a saturation curve.

Consider a channel on the steep part of its curve (under-invested). Marginal ROAS is high: each additional dollar generates substantial incremental revenue. But average ROAS, which includes the inframarginal dollars, is lower. Conversely, a saturated channel has low marginal ROAS but may have reasonable average ROAS because the early dollars were highly productive.

This is why we cannot plug the experimental ROAS directly into the model as a tight prior. Doing so would force the average-ROAS parameter to take a value that only makes sense for the marginal dollar. The fix is to inflate the standard error of the experimental estimate before converting it to a prior — giving the posterior room to land somewhere between the marginal and average values, weighted by the data.

Integration with Meridian

Meridian supports two calibration mechanisms:

- **Informative ROI prior (`roi_m`)**: a channel-specific LogNormal prior based on experimental results. This is a soft constraint that shifts the MCMC starting point but allows the posterior to move away if the data disagree strongly.
- **Calibration period (`roi_calibration_period`)**: a boolean array marking specific weeks and channels as experimentally validated. During sampling, the model constrains the predicted ROI for the marked channel during the marked period to be consistent with the experimental estimate. This is a stronger, more targeted constraint.

Both mechanisms can be used together. The informative prior shifts the starting point; the calibration period adds a tighter constraint during the experiment window.

Converting the DID estimate to a LogNormal prior. Note the `se_inflated` step — this is the marginal-vs-average adjustment we just discussed.

```
import numpy as np

def to_lognormal_params(mean, std):
    """Convert mean/std to LogNormal mu/sigma."""
    variance = std ** 2
    sigma_sq = np.log(1 + variance / mean ** 2)
    sigma = np.sqrt(sigma_sq)
    mu = np.log(mean) - sigma_sq / 2
    return mu, sigma

# Example: DID estimate → LogNormal prior
roas_hat = 5.78  # DID estimate
roas_se  = 3.73  # DID standard error

# Inflate SE (marginal vs. average ROAS adjustment) and apply floor
se_inflated = max(roas_se * 1.5, 0.50)

mu, sigma = to_lognormal_params(roas_hat, se_inflated)
# → mu   1.42, sigma   0.81
```

Setting channel-specific priors in Meridian:

```
import tensorflow as tf
import tensorflow_probability as tfp
from meridian.model import prior_distribution, spec
from meridian import constants

channels = ["Meta", "Google", "TikTok", "TV", "OOH"]

# TikTok: calibrated from geo-lift experiment
# Others: default weakly informative prior
roi_mu    = tf.constant([0.20, 0.20, 1.42, 0.20, 0.20])
roi_sigma = tf.constant([1.00, 1.00, 0.81, 1.00, 1.00])

prior = prior_distribution.PriorDistribution(
    roi_m=tfp.distributions.LogNormal(
        loc=roi_mu,
        scale=roi_sigma,
        name=constants.ROI_M
    ),
)

model_spec = spec.ModelSpec(
    prior=prior,
    media_prior_type="roi",
    max_lag=12,
)
```

The calibrated channel should show a tighter credible interval and a posterior mean closer to the experimental estimate. Other channels may also benefit from reduced uncertainty, since tightening one channel helps the model attribute remaining variation more precisely.

The companion notebook applies this same recipe to **Google Search** — the channel whose uncalibrated posterior was the demo's least well-identified — and produces the before/after comparison in Figure 3. The TikTok worked example above and the Google Search illustration below run through the same `to_lognormal_params` → `roi_m` prior → refit pipeline; only the input DID estimate differs.

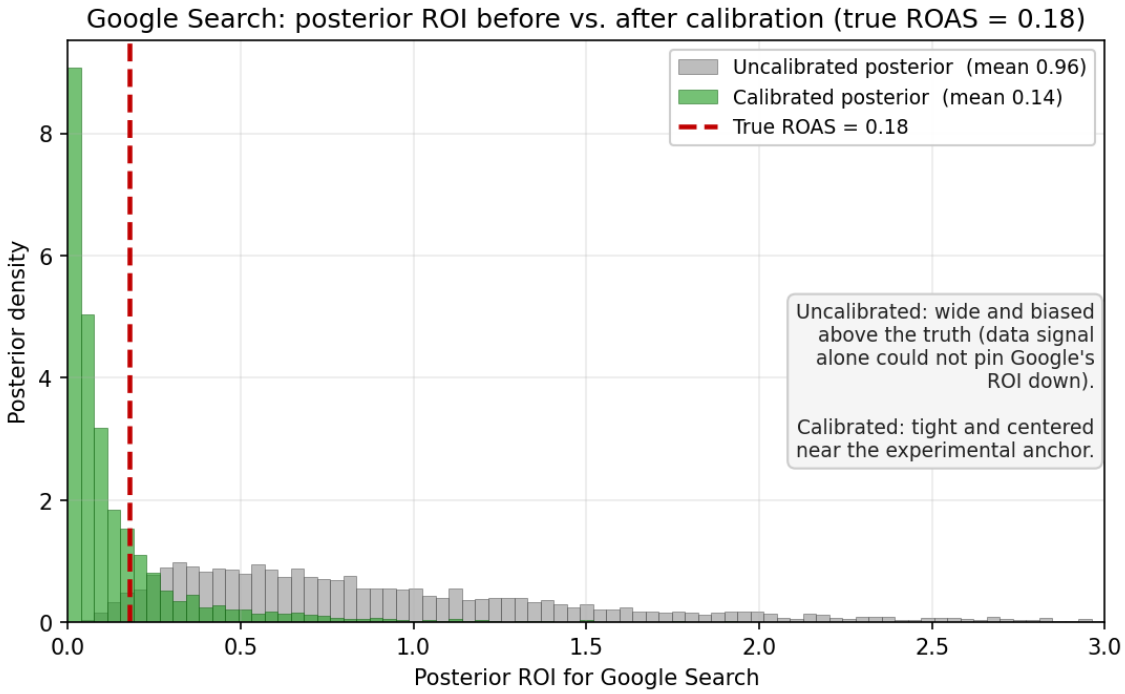


Figure 3: Prior vs. posterior ROI distributions before and after calibration on Google Search.

Running a Calibration Program

Calibration is not a one-time event. ROI changes as audiences evolve, creative fatigues, and competitive dynamics shift.

> **Note**
>
> **A practical annual cycle:** Run the MMM in Q1, identify the channels with the widest credible intervals, run geo-lift tests in Q2–Q3, refresh the model in Q4 with calibrated priors. The Q4 model feeds into the next year's budget planning.

Cost framing. A 20% spend increase in four geos for eight weeks may cost \$50K–\$200K. Frame this as an investment in decision quality: if the calibrated model shifts \$2M in annual allocation 5% in the right direction, the payoff exceeds the experiment cost many times over.

When calibration fails. Sometimes the DID estimate is negative, nonsensical, or excessively noisy. Common causes: parallel trends violation, insufficient spend change, or external shocks during the test period. When this happens, do not force the estimate into the model. An unreliable calibration is worse than no calibration. Diagnose, adjust, and try again.

Calibration is iterative: better data → better models → better experiments → better data, looping each annual planning cycle.

Reliability Checklist

Calibration makes MMM more useful, but it does not automatically make every recommendation trustworthy. Before sharing budget recommendations with a CMO or CFO, run the model through a short reliability checklist. The goal is not to prove that the model is perfect. The goal is to catch the issues that would make the optimizer's output unsafe to use.

Data Quality

1. **Enough history.** Use at least two years of weekly data when possible. With fewer than 80 weekly observations, it becomes difficult to estimate adstock and saturation reliably.

2. **Real variation in spend.** Only include channels where spend actually moves. If a channel spends the same amount every week, the model has little signal to estimate its effect.
3. **Correlated media flights.** Check whether channels move together. If Meta, Google, TV, and TikTok all spike in the same weeks, the model may struggle to separate their individual effects.

Model Design

1. **Controls for non-media demand.** Include seasonality, holidays, promotions, pricing, distribution, and other business drivers where relevant. Otherwise, the model may attribute normal demand spikes to media.
2. **Reasonable response curves.** Inspect saturation and adstock curves. If a curve implies extreme lift from tiny spend changes, or no diminishing returns at all, the model may be fitting noise.
3. **No unsafe extrapolation.** Treat optimizer recommendations cautiously when they push spend far outside the historical range. MMM is much more reliable for interpolation than extrapolation.

Validation

1. **Predictive performance.** Check both in-sample and out-of-sample fit. A strong in-sample fit alone is not enough.
2. **Uncertainty.** Look at credible intervals, not just point estimates. Wide intervals are a warning that the channel's ROI is not well identified.
3. **Experiment consistency.** Compare MMM estimates against geo-lift or holdout results where available. If the two disagree materially, investigate before using the model for budget reallocation.

If the model passes these checks, the optimizer's output is a reasonable starting point for planning. If it fails several of them, the right next step is not to optimize harder. It is to improve the data, redesign the model, or run a better calibration experiment.

Key Takeaways

- **MMM estimates are observational; experiments provide causal anchors.** A geo-lift test can anchor one channel's estimate and make downstream budget recommendations more trustworthy.

- **Geo-lift matches the model's granularity.** MMM operates at the geo-week level, and geo-lift produces estimates at the same level.
- **Marginal ROAS is not the same as average ROAS.** The experiment measures the incremental dollar, while the MMM estimates average return. Inflate the standard error to give the posterior room to reconcile the two.
- **Calibration is iterative.** One to two experiments per year, rotating across channels, is more realistic than trying to validate every channel at once.
- **Do not use the optimizer blindly.** Before sharing recommendations, check data quality, model design, uncertainty, extrapolation risk, and consistency with experiments.

Further Reading

This is a curated list of **recommended further reading** — books, papers, datasets, and code repositories that go deeper on each topic, organized by part and chapter. These are suggestions for going further, not the works formally cited in the text; for the book's curated bibliography see References.

Part 1: Introduction

What is Marketing Science?

- Provost, F. & Fawcett, T. (2013). *Data Science for Business*. O'Reilly Media. — Frames data science around business decisions rather than model accuracy — the book's core stance; worth revisiting here in Part 1.
- Taddy, M. (2019). *Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions*. McGraw Hill. — Connects machine learning to business decision-making, closely aligned with Part 1's thesis.
- Davenport, T. H. & Harris, J. G. (2007). *Competing on Analytics: The New Science of Winning*. Harvard Business School Press. — On how organizations actually collect, analyze, and act on data — a fitting introduction beyond "run the analysis."
- Agrawal, A., Gans, J. & Goldfarb, A. (2018). *Prediction Machines: The Simple Economics of Artificial Intelligence*. Harvard Business Review Press. — Prediction is only one input to a decision; pairs well with the "don't stop at AUC 0.92" message.

Three Characteristics of Marketing Data

- Pearl, J. & Mackenzie, D. (2018). *The Book of Why: The New Science of Cause and Effect*. Basic Books. — An intuitive guide to confounding and unobserved causes; supports the "observed data is the tip of the iceberg" framing (also relevant to Part 3).

- Chan, D. & Perry, M. (2017). "Challenges and Opportunities in Media Mix Modeling." Google Inc. — Concurrent campaigns, multicollinearity, and extrapolation risk illustrate why marketing data is messy (also cited in Part 6).
- Muller, J. Z. (2018). *The Tyranny of Metrics*. Princeton University Press. — On how measurement distorts behavior; relevant to data as a byproduct of operations (and to KPI design in 2.2).

Part 2: Analysis for Better Decisions

Framing the Right Question

- Shron, M. (2014). *Thinking with Data: How to Turn Information into Insights*. O'Reilly Media. — A framework for scoping a project, the data it needs, and the problem actually worth solving — the best single fit for this chapter.
- Hubbard, D. W. (2014). *How to Measure Anything: Finding the Value of Intangibles in Business*. Wiley. — On reducing a fuzzy business question to something measurable at the precision a decision requires.
- Rumelt, R. P. (2011). *Good Strategy/Bad Strategy: The Difference and Why It Matters*. Crown Business. — Bad strategy is vague aspiration; good strategy names the real obstacle — the same discipline applied to problem framing.

Designing KPIs for Diagnosis

- Farris, P. W., Bendle, N. T., Pfeifer, P. E. & Reibstein, D. J. (2020). *Marketing Metrics: The Manager's Guide to Measuring Marketing Performance*. Pearson. — The standard reference for marketing metrics, ROI, and quantifying marketing's contribution to profit.
- Croll, A. & Yoskovitz, B. (2013). *Lean Analytics: Use Data to Build a Better Startup Faster*. O'Reilly Media. — The "One Metric That Matters" idea pairs naturally with building a KPI tree.
- Goodhart, C. A. E. (1975). "Problems of Monetary Management: The U.K. Experience." *Papers in Monetary Economics*. — The origin of Goodhart's Law: a measure that becomes a target stops being a good measure.
- Muller, J. Z. (2018). *The Tyranny of Metrics*. Princeton University Press. — The downside of metrics — not just designing KPIs, but the danger of managing people by them.

Turning Data into Story

- Knaflic, C. N. (2015). *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Wiley. — The most natural fit for this chapter; foundational guidance on data visualization and communication.
- Minto, B. (2009). *The Pyramid Principle: Logic in Writing and Thinking*. Pearson. — The source of the executive Situation–Complication–Resolution structure, close to this chapter's storytelling arc.
- Duarte, N. (2019). *DataStory: Explain Data and Inspire Action Through Story*. Ideapress Publishing. — Argues a data story should inspire action — a decision story, not a list of findings.
- Tufte, E. R. (2001). *The Visual Display of Quantitative Information*. Graphics Press. — The classic on statistical graphics, charts, and tables.
- Few, S. (2013). *Information Dashboard Design: Displaying Data for At-a-Glance Monitoring*. Analytics Press. — A practical complement to Knaflic for dashboard and business-review audiences.
- Wexler, S., Shaffer, J. & Cotgreave, A. (2017). *The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios*. Wiley. — Real-world dashboard examples that connect to the outputs shown to decision-makers.

Part 3: Causal Inference for Marketing

Causal Thinking & Selection Bias

- Pearl, J. & Mackenzie, D. (2018). *The Book of Why: The New Science of Cause and Effect*. Basic Books. — Accessible introduction to DAGs, do-calculus, and the counterfactual framework.
- Angrist, J. D. & Pischke, J.-S. (2009). *Mostly Harmless Econometrics*. Princeton University Press. — The standard reference for the potential outcomes framework and practical identification strategies.
- Hernán, M. A. & Robins, J. M. (2020). *Causal Inference: What If*. Chapman & Hall/CRC. — Graduate-level treatment of causal inference from observational and experimental data; freely available online.
- Cunningham, S. (2021). *Causal Inference: The Mixtape*. Yale University Press. — Practitioner-friendly coverage of DiD, IV, RDD, and synthetic control with code examples.

A/B Testing: Design and Pitfalls

- Kohavi, R., Tang, D. & Xu, Y. (2020). *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press. — The definitive practitioner reference, covering design, analysis, pitfalls, and organizational considerations.
- Deng, A., Xu, Y., Kohavi, R. & Walker, T. (2013). "Improving the Sensitivity of Online Controlled Experiments by Utilizing Pre-Experiment Data." *WSDM '13*. — The original CUPED paper, showing how pre-experiment covariates reduce variance.
- Johari, R., Koomen, P., Pekelis, L. & Walsh, D. (2017). "Peeking at A/B Tests: Why It Matters, and What to Do About It." *KDD '17*. — Formalizes the peeking problem and proposes always-valid inference as a solution.

Quasi-Experiments in Practice

- Angrist, J. D. & Pischke, J.-S. (2009). *Mostly Harmless Econometrics*. Princeton University Press. — Core reference for DiD, IV, and RDD with accessible derivations.
- Abadie, A., Diamond, A. & Hainmueller, J. (2010). "Synthetic Control Methods for Comparative Case Studies." *Journal of the American Statistical Association*, 105(490), 493–505. — The foundational synthetic control paper.
- Cattaneo, M. D., Idrobo, N. & Titiunik, R. (2020). *A Practical Introduction to Regression Discontinuity Designs*. Cambridge Elements. — Covers sharp and fuzzy RDD with the `rdrobust` package.
- Cunningham, S. (2021). *Causal Inference: The Mixtape*. Yale University Press. — Practitioner-oriented treatment of DiD, RDD, and synthetic control with code examples in R and Stata.

Meta-Learners for CATE

- Künzel, S. R., Sekhon, J. S., Bickel, P. J. & Yu, B. (2019). "Metalearners for Estimating Heterogeneous Treatment Effects Using Machine Learning." *Proceedings of the National Academy of Sciences*, 116(10), 4156–4165. — Defines the S/T/X-learner framework and compares performance across settings.
- Chernozhukov, V. et al. (2018). "Double/Debiased Machine Learning for Treatment and Structural Parameters." *The Econometrics Journal*, 21(1), C1–C68. — Introduces DML with cross-fitting to remove regularization bias.
- EconML — Microsoft's Python library for CATE estimation (DML, DR-learner, Causal Forest, and more) with DoWhy integration.

- DoWhy — Causal inference library providing a four-step workflow: model → identify → estimate → refute.

Uplift Modeling: Find the Persuadables

- Gutierrez, P. & Gérardy, J.-Y. (2017). "Causal Inference and Uplift Modelling: A Review of the Literature." *JMLR Workshop and Conference Proceedings*, 67, 1–13. — Comprehensive survey of uplift modeling methods and evaluation metrics.
- Radcliffe, N. J. & Surry, P. D. (2011). "Real-World Uplift Modelling with Significance-Based Uplift Trees." *White Paper, Stochastic Solutions.* — Introduces uplift trees with statistical significance-based splitting criteria.
- CausalML — Uber's Python library for uplift modeling and CATE estimation, including `UpliftRandomForestClassifier` and Qini/AUUC evaluation tools.
- Diemert, E., Betlei, A., Renaudin, C. & Amini, M.-R. (2018). "A Large Scale Benchmark for Uplift Modeling." *KDD '18 Workshop.* — Describes the Criteo uplift dataset (13M samples) used in benchmarking throughout this chapter.

Part 4: Customer Analytics

Customer Segmentation

- Bult, J. R. & Wansbeek, T. (1995). "Optimal Selection for Direct Mail." *Marketing Science*, 14(4), 378–394. — the foundational paper on optimal selection for direct mail using RFM-based targeting. Establishes the breakeven framework for evaluating segment-targeted campaigns.
- Provost, F. & Fawcett, T. (2013). *Data Science for Business.* O'Reilly Media. — comprehensive introduction to data science for business decisions, including practical coverage of segmentation and clustering in a business context.
- Vizard, S. (2019). "P&G shifts from targeting 'generic demographics' to 'smart audiences'." *Marketing Week*, July 30, 2019.
- Marketing Week (2019). "Why behaviour beats demographics when it comes to segmentation." April 15, 2019.

Customer Lifetime Value Modeling

- Fader, P. S., Hardie, B. G. S., & Lee, K. L. (2005). "Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model. *Marketing Science*, 24(2), 275-284. https://www.brucehardie.com/papers/bgnbd_2004-04-20.pdf

- Fader, P. S. & Hardie, B. G. S. (2013). The Gamma-Gamma Model of Monetary Value. https://www.brucehardie.com/notes/025/gamma_gamma.pdf
- PyMC-Marketing: https://github.com/pymc-labs/pymc-marketing
- Online Retail Dataset (License: CC BY 4.0): https://archive.ics.uci.edu/dataset/352/online+retail

Part 5: Commercial Analytics

Commercial Metrics

- Byron Sharp, *How Brands Grow* (2010) — the empirical case for penetration-driven growth.
- Tan, Steinbach, & Kumar, *Introduction to Data Mining* — chapter on association analysis for a deeper treatment of support, confidence, and lift.

Price Elasticity and Pricing Decisions

- Hoch, S. J., Kim, B., Montgomery, A. L., & Rossi, P. E. (1995). Determinants of store-level price elasticity. *Journal of Marketing Research*, 32(1), 17–29.
- Tellis, G. J. (1988). The price elasticity of selective demand: A meta-analysis of econometric models of sales. *Journal of Marketing Research*, 25(4), 331–341.
- Hermann Simon, *Confessions of the Pricing Man* (2015) — accessible treatment of pricing strategy, written by the founder of Simon-Kucher.
- The JCPenney "Fair and Square" pricing case — a cautionary tale about eliminating promotions without understanding reference price effects. Customers revolted not because the new prices were higher, but because the *feeling* of getting a deal disappeared.

Product Assortment Optimization

- Kök, A. G., Fisher, M. L., & Vaidyanathan, R. (2015). "Assortment Planning: Review of Literature and Industry Practice." In *Retail Supply Chain Management*, Springer. — The comprehensive academic review of assortment optimization methods.
- Iyengar, S. S., & Lepper, M. R. (2000). "When Choice Is Demotivating: Can One Desire Too Much of a Good Thing?" *Journal of Personality and Social Psychology*. — The original "jam study." Influential but contested.

- Scheibehenne, B., Greifeneder, R., & Todd, P. M. (2010). "Can There Ever Be Too Many Options? A Meta-Analytic Review of Choice Overload." *Journal of Consumer Research*. — The meta-analysis that found the average choice-overload effect is close to zero, with high variability across contexts. Treat the paradox of choice as a hypothesis to test, not a universal law.
- Train, K. E. (2009). *Discrete Choice Methods with Simulation*, 2nd ed., Cambridge University Press. — The standard textbook for MNL and Mixed Logit. Free online at https://eml.berkeley.edu/books/choice2.html.
- `pylogit` (Python) and `xlogit` (Python, GPU-accelerated) — accessible libraries for discrete choice modeling.

Demand Forecasting

- Hyndman, R. J., & Athanasopoulos, G., *Forecasting: Principles and Practice* (3rd ed.) — free online at otexts.com/fpp3. The standard reference for time-series forecasting. Chapters on ETS, ARIMA, and hierarchical reconciliation are particularly relevant.
- Nixtla open-source libraries: statsforecast, mlforecast, hierarchicalforecast — fast, scalable, well-documented. The unified `unique_id / ds / y` format is a genuine productivity advantage.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting.* — The competition that demonstrated gradient boosting's advantage over statistical models for large-scale retail forecasting.
- Ansari, A. F., et al. (2024). Chronos: Learning the Language of Time Series. — The architecture and pretraining approach behind Chronos 2. Relevant for understanding zero-shot forecasting.
- Olivares, K. G., et al. (2022). HierarchicalForecast: A Reference Framework for Hierarchical Forecasting in Python. — Theory and implementation of MinTrace, ERM, and other reconciliation methods.
- Fildes, R., & Goodwin, P. (2007). Against your better judgment? How organizations can improve their use of management judgment in forecasting. *Interfaces, 37*(6), 570–576. — The evidence on human override bias and FVA.
- GIFT-Eval (Salesforce Research). github.com/SalesforceAIResearch/gift-eval — A benchmark for fair comparison of Foundation Models on time-series tasks.

Part 6: Media Investment and Optimization

Attribution: Concepts and Limits

- Google. "About Data-Driven Attribution." Google Ads Help.
- Berman, R. (2018). "Beyond the Last Touch: Attribution in Online Advertising." *Marketing Science*, 37(5), 771-792.
- Dalessandro, B. et al. (2012). "Causally Motivated Attribution for Online Advertising." *ADKDD '12*.

MMM Fundamentals

- Jin, Y., Wang, Y., Sun, Y., Chan, D. & Koehler, J. (2017). "Bayesian Methods for Media Mix Modeling with Carryover and Shape Effects." Google Inc. — Google's paper on Bayesian MMM with hierarchical priors and geo-level data, the methodological foundation for Meridian.
- Chan, D. & Perry, M. (2017). "Challenges and Opportunities in Media Mix Modeling." Google Inc. — Companion paper covering common pitfalls in MMM (multicollinearity, extrapolation, insufficient variation) and how to address them.
- Meridian — Google's actively maintained Bayesian MMM library (successor to LightweightMMM), used in the walkthrough in this chapter.
- Robyn — Meta's open-source MMM library using Ridge regression with hyperparameter optimization via Nevergrad.
- PyMC-Marketing — Bayesian MMM and CLV library built on PyMC, with flexible model specification and saturation/adstock transforms.

Budget Optimization

- Jin, Y., Wang, Y., Sun, Y., Chan, D. & Koehler, J. (2017). "Bayesian Methods for Media Mix Modeling with Carryover and Shape Effects." Google Inc. — Describes the Bayesian MMM framework that underpins Meridian's budget optimizer, including the posterior-based optimization approach.
- Chan, D. & Perry, M. (2017). "Challenges and Opportunities in Media Mix Modeling." Google Inc. — Discusses practical challenges in MMM-based optimization, including extrapolation risks and multicollinearity.
- Meridian Budget Optimization Guide — Documentation for Meridian's `BudgetOptimizer`, including per-channel constraints and scenario analysis.

- Fischer, M., Albers, S., Wagner, N. & Frie, M. (2011). "Dynamic Marketing Budget Allocation Across Countries, Products, and Marketing Activities." *Marketing Science*, 30(4), 568–585. — Multi-level budget allocation framework with diminishing returns and cross-effects.

Calibrating MMM with Experiments

- Jin, Y., Wang, Y., Sun, Y., Chan, D. & Koehler, J. (2017). "Bayesian Methods for Media Mix Modeling with Carryover and Shape Effects." Google Inc. — Google's Bayesian MMM paper describing the prior calibration framework that Meridian implements.
- Vaver, J. & Koehler, J. (2011). "Measuring Ad Effectiveness Using Geo Experiments." Google Inc. — The foundational paper on geo-lift testing methodology.
- Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N. & Scott, S. L. (2015). "Inferring Causal Impact Using Bayesian Structural Time-Series Models." *The Annals of Applied Statistics*, 9(1), 247–274. — Bayesian structural time-series approach to geo experiments, an alternative to DID.
- Meridian Calibration Guide — Official documentation for `roi_m` priors and `roi_calibration_period`.
- GeoLift — Open-source R package for designing and analyzing geo-lift experiments.

Notebooks & Code

Notebooks & Code

Every hands-on chapter has a companion notebook, committed **with its outputs** — so you can read the code and see the figures and results right on GitHub, open it in Google Colab and run it yourself with zero local setup, or read a static render on nbviewer.

Colab is best-effort. The heavier notebooks (PyMC-Marketing, Google Meridian, CausalML, BERTopic) pull large dependencies and some need external datasets, so a Colab session may need extra `pip install`s or be slow. For a reliable run, follow the local setup. The CLV chapter also ships a three-step LightGBM companion pipeline under `sec4.2-clv/lightgbm-companion/` (`01_data_features.py`, `02_modeling.py`, `03_evaluation.py`), linked directly from that chapter.

Status legend. *Done* — ready to read as-is, outputs match the chapter prose. *Almost Done* — published output is usable as a draft; minor caveats (noted below each table) remain. *WIP* — a re-execution, dependency, or chapter-prose alignment is still unresolved, so treat the current output as a preview only.

Part 3 — Causal Inference for Marketing

Notebook	Topic	Status	Run
`simple_criteo_meta_learners_ecml.ipynb`	Meta-learners, decile targeting, and uplift / Qini diagnostics (EconML, DoWhy, scikit-uplift)	Done	Colab · nbviewer · GitHub

The meta-learners notebook is executed against the real Criteo 10 % subset (~1.4 M samples), so CATE distributions, decile targeting, and the uplift / Qini diagnostics in §10 all reflect production-grade data. §3.5 (Uplift Modeling for Targeting) explains the conceptual framing — four customer types, uplift-tree splitting criterion, AUUC / Qini — and reuses this same notebook for the runnable workflow.

Part 4 — Customer Analytics

Notebook	Topic	Status	Run
`traditional_segmentation.ipynb`	Segmentation — decile / RFM / K-Means (scikit-learn)	Done	Colab · nbviewer · GitHub
`semantic_segmentation.ipynb`	Segmentation — embedding / BERTopic (BERTopic, sentence-transformers)	Done	Colab · nbviewer · GitHub
`PyMC_Marketing_CLV_demo.ipynb`	Customer lifetime value (PyMC-Marketing)	Done	Colab · nbviewer · GitHub

`traditional_segmentation` agrees with the chapter (silhouette and elbow both point to K=3) and ships with per-cluster action tiers and a CRM-ready export schema. `semantic_segmentation` runs the LLM-naming cell against a real API in the published outputs; in the `marketing-science` conda env `litellm` and `anthropic` are pre-pinned, while Colab users still need to `pip install litellm` and set `OPENAI_API_KEY` or `ANTHROPIC_API_KEY` to reproduce the naming step. `PyMC_Marketing_CLV_demo` fits BG/NBD and Gamma-Gamma with NUTS MCMC, exports per-customer revenue and profit CLV at 3-/5-/10-year horizons with HDI bounds, a behavioral-segment join from sec4.1, an allowable-CPA column with a conservative (HDI 3%) variant, a 30-day P(alive)-drop churn watchlist, and a 180-day holdout calibration table (Pearson r 0.80, cohort revenue ratio 0.92 — inside the chapter's ±20% retrain tolerance). The 18 high-frequency customers whose 10-year DCF posterior becomes numerically unstable are flagged `clv_estimation_status="nonfinite_review"` rather than silently zeroed.

Part 5 — Commercial Analytics

Notebook	Topic	Status	Run
`price_elasticity_scanner_data.ipynb`	Price elasticity from scanner data (statsmodels)	Done	Colab · nbviewer · GitHub
`assortment_optimization.ipynb`	Assortment optimization — revenue + basket-reach ABC, substitution, matched-store DiD (statsmodels)	Done	Colab · nbviewer · GitHub
`demand_forecasting.ipynb`	Demand forecasting — models, evaluation & hierarchical reconciliation (Nixtla)	Done	Colab · nbviewer · GitHub

Chapter 5.2 (Product Assortment Optimization) ships both the runnable notebook above and the companion `assortment_optimization.py` script, linked from that chapter.

Part 6 — Media Investment and Optimization

Notebook	Topic	Status	Run
`mmm_end_to_end_demo.ipynb`	Media mix modeling (Google Meridian)	Done	Colab · nbviewer · GitHub

The MMM demo is now end-to-end re-executable: TF 2.19 / TFP 0.25 / Meridian 1.6 are pinned in `environment.yml`, the published outputs use the current synthetic-data CSV with positive ground-truth ROI for all five channels, the pre-fit identifiability check from §6.3's Reliability Checklist runs in Step 2b (history, OFF rate, share std, channel correlations) with concrete PASS/WARN/FAIL verdicts, a random 25% week-level holdout produces in-sample R^2 0.95 and out-of-sample R^2 0.94, and the optimizer's reallocation table (channel-level deltas plus expected lift +7%) lands in the notebook itself. The uncalibrated 90% credible interval contains the true ROAS for **four of the five channels**; the fifth (Google Search) misses by a hair, and the notebook then runs the **full §6.3**

calibration loop on Google: convert a hypothetical geo-lift DID into a tight LogNormal prior, refit the model, compare before/after, and re-optimize from the calibrated posterior. After calibration, **all five channels' 90% CIs contain the true ROAS**, the other four channels are essentially unchanged, and Google's posterior mean shrinks from 0.957 (5× truth) to 0.144 (close to truth 0.18). The calibrated optimizer recovers Google's true ROAS exactly (0.18) and shifts the TV/CTV recommendation from -13% to -19% while easing TikTok's cut from -15% to -11% — calibration changes both the estimate and the budget recommendation it drives. A **credible interval guardrail** cell converts each channel's ROI posterior into an act/hold rule on the uncalibrated fit; the resulting planner-ready move is much smaller than the optimizer's raw point-estimate recommendation. The §6.3 worked TikTok example (`mu  1.42`, `sigma  0.81`) is reproduced byte-for-byte via an embedded `assert`. Local runtime on CPU is roughly ten minutes with the demo budget (4 chains × 1000 keep, two MCMC passes share an XLA cache); production should still use 10 chains × 2000 keep.

References

This is the book's curated bibliography — the academic papers, books, and library documentation that underpin the methods presented throughout. Entries are generated from `references.bib` and ordered alphabetically by author. For recommended and supplementary readings organized by topic, see Further Reading.

Brodersen, Kay H, Fabian Gallusser, Jim Koehler, Nicolas Remy, and Steven L Scott. 2015. "Inferring Causal Impact Using Bayesian Structural Time-Series Models." *The Annals of Applied Statistics* 9 (1): 247–74.

Bult, Jan Roelf, and Tom Wansbeek. 1995. "Optimal Selection for Direct Mail." *Marketing Science* 14 (4): 378–94.

Chan, David, and Matthew Perry. 2017. *Challenges and Opportunities in Media Mix Modeling*. Google Inc. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45998.pdf.

Fader, Peter S, and Bruce GS Hardie. 2013. *The Gamma-Gamma Model of Monetary Value*. Wharton School, University of Pennsylvania. https://www.brucehardie.com/notes/025/gamma_gamma.pdf.

Fader, Peter S, Bruce GS Hardie, and Ka Lok Lee. 2005. "Counting Your Customers the Easy Way: An Alternative to the Pareto/NBD Model." *Marketing Science* 24 (2): 275–84.

Jin, Yuxue, Yueqing Wang, Yunting Sun, David Chan, and Jim Koehler. 2017. *Bayesian Methods for Media Mix Modeling with Carryover and Shape Effects*.

Knaflic, Cole Nussbaumer. 2015. *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Wiley.

Pritchard, Marc. 2019. "Marc Pritchard on 5 Actions P&G Is Taking to Improve the Media Supply Chain." https://www.pg.co.uk/blogs/PritchardANA2019/.

Provost, Foster, and Tom Fawcett. 2013. *Data Science for Business*. O'Reilly Media.

Rogers, Charlotte. 2019. "Why Behaviour Beats Demographics When It Comes to Segmentation." *Marketing Week*, April. https://www.marketingweek.com/behaviour-demographics-segmentation/.

Vaver, Jon, and Jim Koehler. 2011. *Measuring Ad Effectiveness Using Geo Experiments*. Google Inc. https://research.google/pubs/pub38355/.

Vizard, Sarah. 2019. "P&G Shifts from Targeting 'Generic Demographics' to 'Smart Audiences'." *Marketing Week*, July. https://www.marketingweek.com/pg-targeting-generic-demographics-to-smart-audiences/.