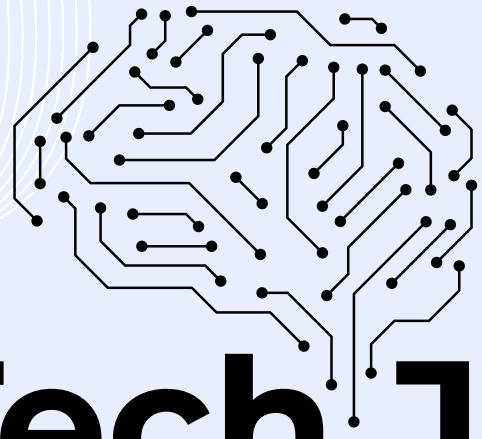


Tech Jargon

Part 5

for

Product Managers

When Your Product Uses AI

Models | Training Data | Hallucinations | Guardrails

From an Engineer-Turned-PM

The AI Risk & Value Mental Model

A way of thinking about AI features as **models plus data plus risk** — not magic.

It helps PMs move past the hype and ask: What data trains it? How confident is it? What happens when it's wrong? Get this wrong, and you ship features that hallucinate, discriminate, or erode trust. Get it right, and AI becomes a product advantage with guardrails from day one.

When you understand this mental model, you can:

- Scope AI features realistically — what needs training data and iteration vs. what can be rule-based
- Ask about confidence scores, inference speed, and failure modes instead of treating the model as a black box
- Design for hallucinations, bias, and edge cases with guardrails and human review built in

This guide covers the 10 essential terms that bring this mental model to life.

AI vs ML

AI is the umbrella; ML is data-driven learning

AI is the broad field of making machines act intelligent — from rule-based systems to generative models. Machine learning is a subset of AI where systems learn patterns from data instead of following hand-written rules.

Think of a spam filter that blocks emails containing "free money" — that's AI using fixed rules. A spam filter that learns from millions of emails what spam looks like and catches tricks it's never seen before — that's ML. Same product, different engine under the hood.

This isn't just a rules engine — it's an ML model. So we can't just write if/else logic, we need labeled training data and time to iterate.

For PMs, this means the first question for any AI feature is: rules, ML, or a hybrid? Rule-based parts ship like normal software. ML parts need data, training cycles, and ongoing evaluation — which means different timelines, milestones, and a plan for how the model improves after launch.

Model

The "brain" that makes predictions or generates content

A model is the trained system that takes input and produces output — whether that's classifying an image, recommending a product, or generating text. It's the result of training on data.

Think of it as a map built from satellite photos — incredibly detailed where the satellites had clear views, but full of blank spots and guesses where there was cloud cover. A model's accuracy is a direct reflection of where its training data was rich and where it was thin.

The model performs well on English queries, but accuracy drops hard once you go multilingual. We'd need to retrain on a broader dataset.

For PMs, this means every model has blind spots shaped by its training. Your job is to find them before users do — ask where the model was trained, what's missing, and what happens in the gaps.

Training data

Historical examples the model learns from

Training data is the set of labeled examples a model learns from. The quality, size, and representativeness of this data directly determine how well the model performs.

Think of it as the textbooks a student studies from: if the books are outdated, biased, or missing key topics, the student's answers will reflect those gaps.

We only have six months of training data. If you want the model to handle seasonal patterns, we need at least two years' worth.

For PMs, this means you need to ask where training data comes from, how fresh it is, and what's missing — because bad data in means bad predictions out, no matter how sophisticated the model.

Inference

Using the trained model to produce outputs

Inference is when a trained model processes new, real-world input and returns a prediction or result. Training happens once (or periodically); inference happens every time a user interacts with the feature.

Think of it as the difference between studying for an exam and taking the exam: training is the studying, inference is answering questions under real conditions.

Training took two weeks on a GPU cluster. But inference needs to return results in under 200ms or the UX is going to feel laggy.

For PMs, this means you care about inference speed and cost — because that's what users actually experience. A brilliant model that takes 10 seconds to respond will still feel broken.

AI feature /Rec System

Product behavior powered by a model

An AI feature is any product capability where the behavior is driven by a model rather than explicit rules. Recommendation systems are the most common example — suggesting content, products, or actions based on user patterns.

Think of it as the difference between a bookstore sorted alphabetically and one that rearranges its front table based on what people in your neighborhood have been buying. One follows a fixed rule; the other adapts based on patterns — and sometimes gets it wrong.

The rec engine is driving 30% of homepage clicks, but we haven't audited whether it's creating a filter bubble. It might just be reinforcing the same content.

For PMs, this means you own the "why" and "what should happen" — defining what good recommendations look like, what guardrails exist, and how to measure success beyond just engagement.

Confidence score

How sure the model is about its output

A confidence score is a number (usually 0–1 or a percentage) that reflects how certain the model is about a specific prediction. High confidence doesn't always mean correct — it means the model thinks it's correct.

Think of it as a weather forecast saying "90% chance of rain" — it's the system's best estimate, not a guarantee. You still might want an umbrella at 60%

Anything below a 0.7 confidence score, we should probably not show to the user. We can fall back to the default experience instead.

For PMs, this means you use confidence scores to design the UX: when to show results confidently, when to hedge with "we think," and when to fall back to a safer default.

Hallucination

When a model confidently returns wrong info

A hallucination is when an AI model generates output that sounds plausible and confident but is factually incorrect or completely made up. It's not lying — it's pattern-matching without understanding truth.

Think of it as predictive text finishing your sentence — it picks the most statistically likely next word, not the most accurate one. "The capital of Australia is..." might autocomplete to "Sydney" because that pairing appears more often. Fluent, confident, and wrong.

The model hallucinated a return policy that doesn't exist. If this goes to production, customers will quote it back to support and we'll have a real problem.

For PMs, this means you must design for hallucinations, not just hope they don't happen. That means verification layers, user-facing disclaimers, and clear escalation paths.

Guardrails

Rules and filters that constrain model behavior

Guardrails are the rules, filters, and boundaries placed around a model to prevent harmful, off-topic, or low-quality outputs. They're the safety net between what the model can do and what it should do.

Think of it as parental controls on a streaming service — the content library is huge and the recommendations are still personalized, but certain categories are blocked entirely. The system works freely within the boundaries you set.

We need input validation to block prompt injection, and output filters to catch anything that mentions competitor pricing or internal data.

For PMs, this means you define what the model should never do — topics to avoid, tone to maintain, data it shouldn't reveal — and work with engineers to enforce those boundaries.

Bias & fairness

Systematic skew in outputs across groups

Bias in AI means the model systematically produces different quality or types of results for different groups of people — often reflecting biases present in the training data or design choices.

Think of it as a hiring manager who unconsciously favors candidates from their own university: the bias isn't intentional, but it's real and measurable in the outcomes.

We ran the fairness audit and the approval model has a 15% higher rejection rate for certain demographics. It's picking up patterns in the historical data that we need to correct for.

For PMs, this means you need to ask "who might this model treat unfairly?" before launch, not after. Define fairness metrics, test across user segments, and plan for ongoing audits.

Human-in-the-loop

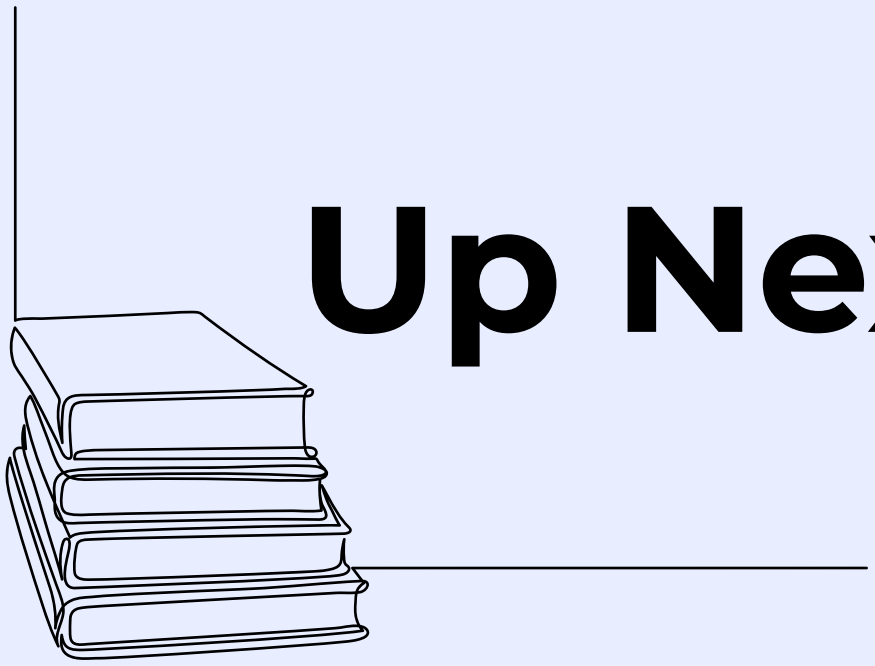
Humans reviewing or overriding AI decisions

Human-in-the-loop means a person reviews, approves, or corrects the model's output before it reaches the end user or triggers an action. It's a design pattern for high-stakes or low-confidence scenarios.

Think of it as a pharmacist double-checking a doctor's prescription before it's filled — the doctor makes the call, but the pharmacist verifies it before it reaches the patient. The system does the heavy lifting; the human catches what it misses.

For anything involving money — loan approvals, refund decisions — we're keeping a human in the loop. The model flags and ranks, but a person makes the final call

For PMs, this means you decide where human review is worth the cost and delay. The answer depends on risk, confidence thresholds, and what happens if the model gets it wrong.



Stay Tuned

Up Next

A plain-English guide to the LLM terms showing up on every roadmap — prompts, tokens, RAG, agents, and the cost math your CFO is already asking about. Because "we'll use AI" isn't a plan.