STATSIG

The AI Validation Gap: How to Build, Test, and Monitor AI Features That Deliver

Introduction:
The AI evaluation crisis

Table of contents

The AI boom is well underway. Every day, there seems to be another SaaS product shipping a new summarizer, recommender, or chatbot.

It's an exciting time in tech, but a troubling pattern is emerging across all these AI-powered products. Even if a model performs well on technical benchmarks, there's no guarantee that they will accrue to a business's goal. Without careful testing and validation, these once-exciting new features often fall flat; users bounce, trust erodes, and support tickets spike.

This is no coincidence. This is the gap between offline model validation and real-world product success. The MIT Technology Review recently called this the AI evaluation crisis:

> "Human preference testing has also emerged as an alternative to benchmarks … AI researchers are beginning to realize—and admit—that the status quo of AI testing cannot continue."

Many AI developers and leaders fail to understand that AI features are about more than just the model itself. In the real world, it's about the UX, context, and user goals surrounding them.

In this guide, we'll dive into how product leaders are addressing the AI evaluation crisis in practical steps with a three-layered approach to AI evaluation.

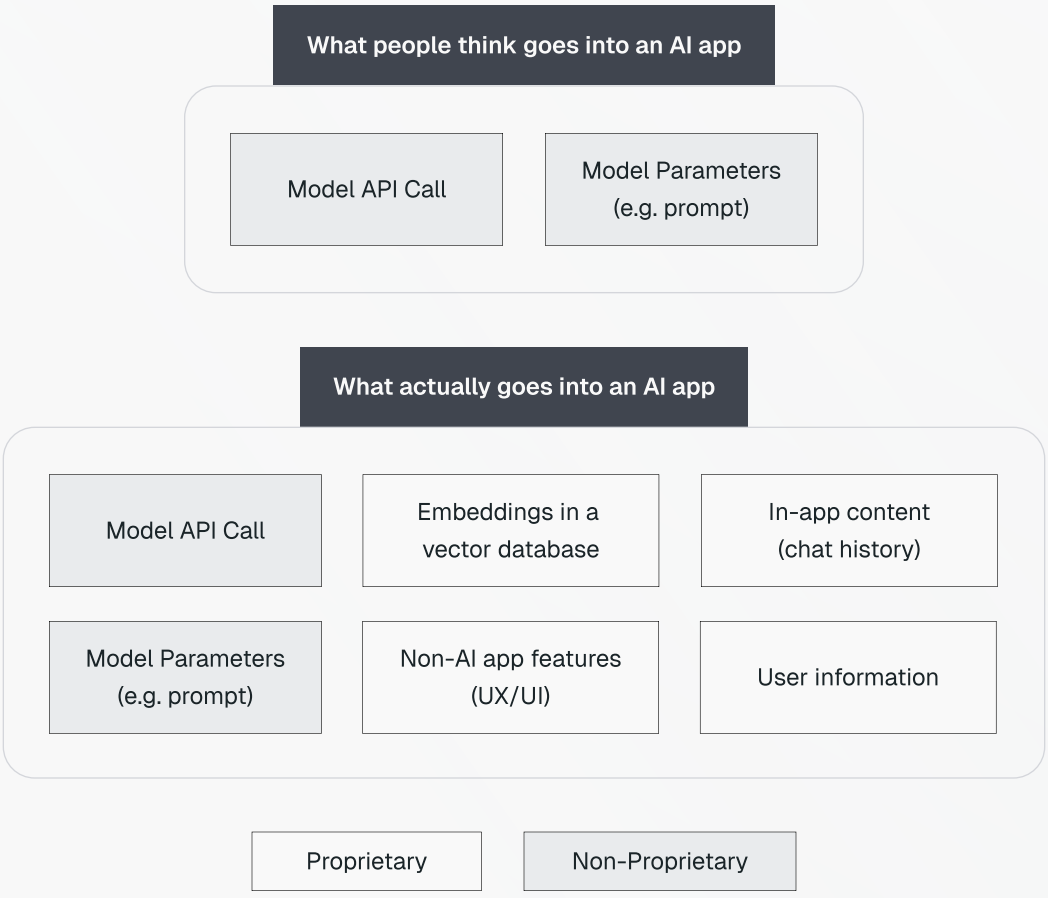
The core problem:
Benchmarks don’t measure reality

For traditional machine learning, metrics like accuracy, AUC or ROUGE were often “good enough.” They worked because the tasks were constrained: rank this list, label that image, predict a number.

In practice, an AI feature is about much more than just model performance. There are entire UX and contexts surrounding it that impact the bottom line of a business. Failing to design and prepare for this can be incredibly damaging.

The evidence is overwhelming:

- **Meta’s Galactica** scored well internally but was pulled within three days for fabricating scientific facts (Source: MIT Technology Review).
- **Air Canada’s AI chatbot** hallucinated fake refund policies, triggering a lawsuit and reputational damage (Source: CBC News).
- **Stanford’s research** showed general-purpose LLM chatbots hallucinating legal facts up to **82% of the time**, causing severe real-world legal repercussions (Source: Bommasani et al.).



The real solution:
Product-level validation

This is the only question that really matters, and there's only one reliable way to answer it: real user data.

What actually works:

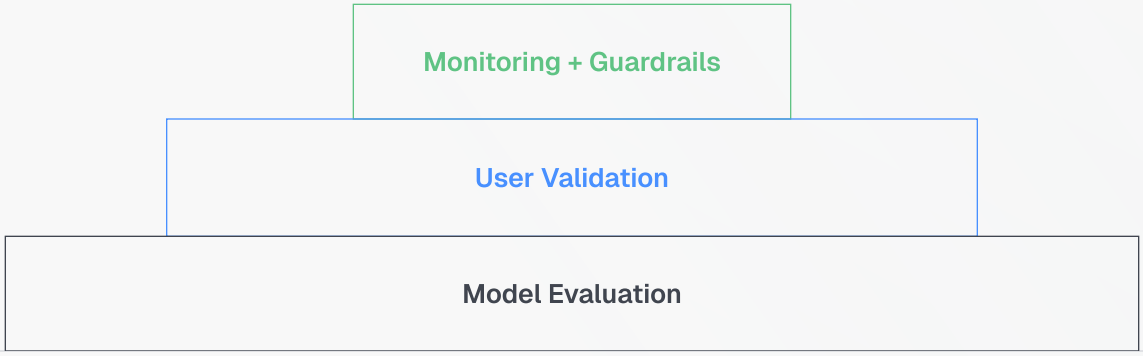
- A/B testing AI-powered features against baselines or different models.
- Holdouts to measure the cumulative impact of new features and catch metric regressions.
- Trust and safety guardrails tied to user behavior and business metrics, not model confidence scores.

This is how fast-growing companies like Notion make new AI features stick—by running the same experiments they would for any major product change.

Product-level case studies:

- Notion's AI succeeded because they didn't just guess what users wanted. They shipped behind flags, ran experiments, and validated that AI improved real user workflows (Source: Vercel, Statsig).
- Cursor, voted Product of the Year 2024 on Product Hunt, thrived by embedding AI deeply into its IDE, emphasizing seamless user experience rather than just benchmark claims (Source: Product Hunt).

The Full AI Testing Stack



The full AI testing stack isn't just about LLM infra; it includes everything that happens between the model and the user.

While many organizations have dedicated solutions, teams, and tools for each of these layers, one of the biggest hurdles in modern development is cross-functional handoff and alignment. Product growth and experimentation tools like Statsig offer solutions for each layer, all in a unified platform for stakeholders to make informed business decisions together.

- **Layer 1: Model Evaluation**

  Check if the model produces coherent, relevant, and safe outputs in a controlled setting.

- **Layer 2: User Validation**

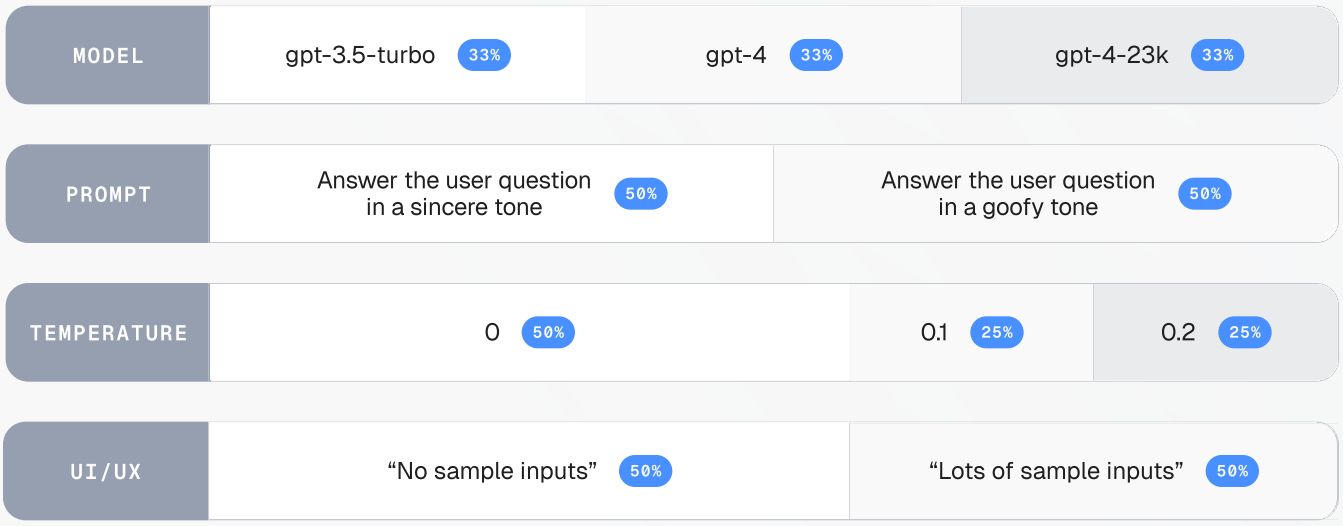
  Test whether the AI experience actually improves user outcomes compared to the baseline.

- **Layer 3: Monitoring & Guardrails**

  Track ongoing performance and user trust to catch silent failures after launch.

Layer 1: Model Evaluation

The first filter in the AI product development process is model evaluation, or testing how the model performs in a controlled (usually offline) environment. This step helps catch functional failures, hallucinations, and quality issues before anything reaches production.



Common model evaluation techniques

Teams typically run predefined evaluation sets, manually review prompt-response pairs, or rely on LLM-as-a-judge techniques where one model scores the outputs of another. Automated tools like toxicity classifiers and hallucination detectors can help catch known pitfalls.

Common methods include offline prompt evaluations, using labeled datasets for expected outputs, and running outputs through rule-based or model-based scoring filters. These techniques help establish a baseline level of model quality before progressing to user-facing experiments.

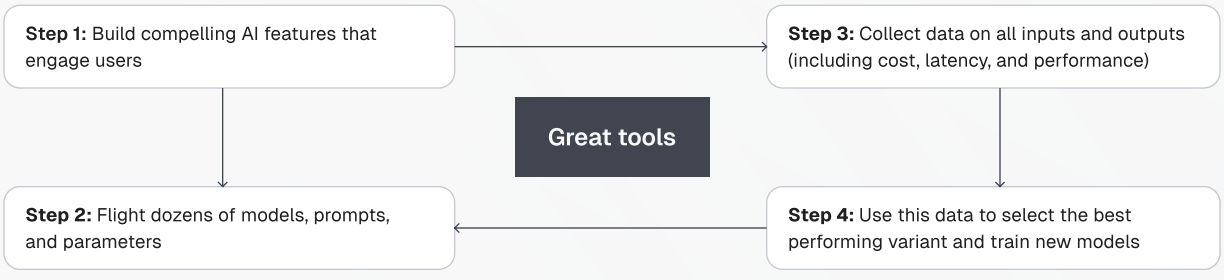
Benchmarks are not enough:

Offline checks only measure isolated output quality and are poor indicators of actual product success.

- Traditional NLP metrics like BLEU and ROUGE show poor correlation with human quality judgments, especially in open-ended tasks (Source: OpenAI).
- InstructGPT (1.3B parameters), tuned with human feedback, significantly outperformed the much larger GPT-3 (175B parameters) in human preference evaluations (Source: Ouyang et al.).

A model can pass every eval and still fail in the real world because it doesn't actually help users get their job done. That's why it's critical to have a strong way to go from this model evaluation layer to the next: user validation.

Layer 2: User Validation



It's critical to see how the AI performs in context—with real users, real use cases, and real stakes. Controlled product experiments and feature flags let you measure whether the AI-powered feature actually improves key outcomes like engagement, task completion, or revenue.

With the high compute costs of AI, early validation at small scales is especially crucial before committing more resources. Analytics platforms with built-in stats engines give teams the confidence to keep, cut, or continue a new AI program without second-guessing ROI.

Experimentation must-haves for user validation

A/B testing AI-powered features

- With AI products, the simplest way is to A/B test between AI or traditional experiences as the baseline. By randomly assigning users between versions of an experience, you can directly compare “Is AI better?” and not just “Is AI working?”
- If your A/B testing tool already does robust statistical testing for interaction effect detection, you can rapidly test multiple AI variants and features at the same time.

Benchmarks are not enough:

- Holdouts are a product experimentation technique where a percentage of users on the non-AI version indefinitely, sort of like a permanent control group.
- These can help you quickly catch silent regressions that new AI features can introduce such as confusion, friction, user churn, or bugs.

The hard truth about AI products is that a feature doesn’t succeed just because the model looks good, or because it was launched successfully. It succeeds if, and only if, real users prefer the AI-powered experience over the baseline. And the only way to know that is through product experimentation.

Layer 3: Guardrails + Monitoring

Even after a successful launch and experiment, AI-powered features are still at risk of silent failures. Unlike traditional software, their outputs fluctuate based on model updates, prompt changes, data drift, and API changes from third-party LLM providers.

As this drift occurs, quality can degrade subtly over time, and issues might only show up in edge cases or downstream business metrics. That’s why ongoing monitoring and automated guardrails are essential.

Methods for monitoring AI success

01	**Feature flags for AI, wrapped in experiments** Launch AI features with flags and wrap them as product experiments permanently. This way, you continuously monitor user behavior and can turn off degraded models or problematic behaviors instantly.
02	**Alerting on trust metrics** Instead of just monitoring system health for crashes and bugs, monitor user trust health: opt-outs, abandonment, negative edits, and spikes in “undo” behaviors.
03	**Built-in rollback tools and plans** Rollback isn’t just for infrastructure risks. Be prepared to revert model versions, prompt versions, and entire AI-driven flows if trust metrics degrade.

AI isn’t “set-it-and-forget-it”. It’s a fundamentally different type of product; a living, probabilistic system that needs permanent guardrails. Monitoring trust signals is as critical as monitoring uptime or error rates.

Conclusion

AI is easy to ship, but hard to get right. The only way to build AI features that actually work is the same way product teams have validated software for decades: **experimentation**.

Product experiments are nothing new. It's been the backbone of how companies like Facebook, Netflix, Amazon, Uber, and Airbnb built products that scaled. These companies didn't rely on intuition. They ran experiments to understand what worked for their users and what didn't.

The difference is that, in the past, engineers shipped deterministic features. You knew exactly how a button, a ranking algorithm, or a recommendation rule behaved, even if the business impact wasn't fully clear.

With AI, that certainty is gone. Foundation models are probabilistic, not deterministic. Outputs vary based on prompts, user inputs, context, and even silent model updates. What looks fine in a demo or benchmark might quietly fail in production, hurting the user experience, degrading trust, or driving churn without anyone noticing.

This is why AI requires a different level of discipline. You need to continuously answer:

- **Layer 1: Model Evaluation**

  Does this AI actually help users?

- **Layer 2: User Validation**

  Do users prefer it over the baseline?

- **Layer 3: Monitoring & Guardrails**

  Is it still working as intended over time?

The only way to answer these questions is through a continuous product loop: **evaluate → experiment → monitor → improve.**

Most tools only solve one part of that loop. Statsig is the leading tool that connects all three layers in one integrated platform.

If you're ready to take the next step, try out our tool for yourself.

SEE HOW IT WORKS

Our Customers

OpenAI · Vercel · character.ai · Notion · webAI

Speak · Buddy.ai · ATLASSIAN · Otter.ai

Supernormal · Read AI · Meshy · ANTHROPIC