Experimenting with AI-driven Systems

SREcon EMEA 2025

Jay Lees
Production Engineer

Javier Martin Montull
Production Engineering Manager

Meta

MEET THE SPEAKERS

Jay Lees

Production Engineer
Monetization Infra & Ranking

Javier Martin Montull

Production Engineering Manager
Monetization Infra & Ranking


EMERGENCY

Agenda

01 Ads Experimentation at Meta

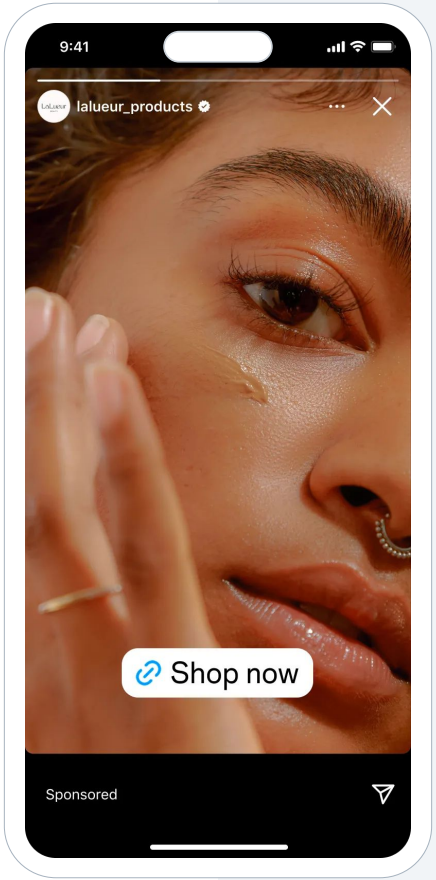
02 Experimentation as the Default

03 Experiment Safety

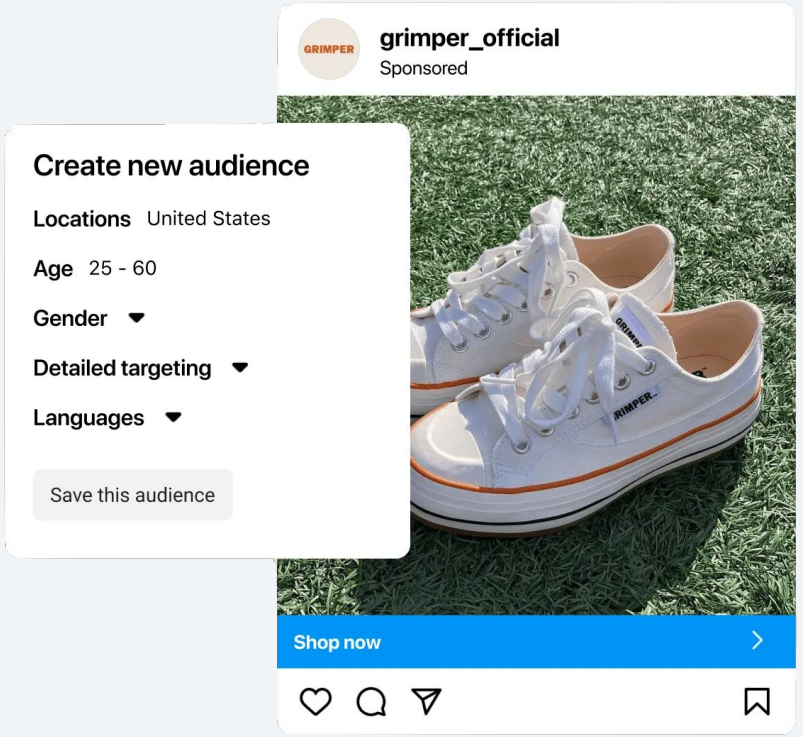
04 Lessons & Learnings

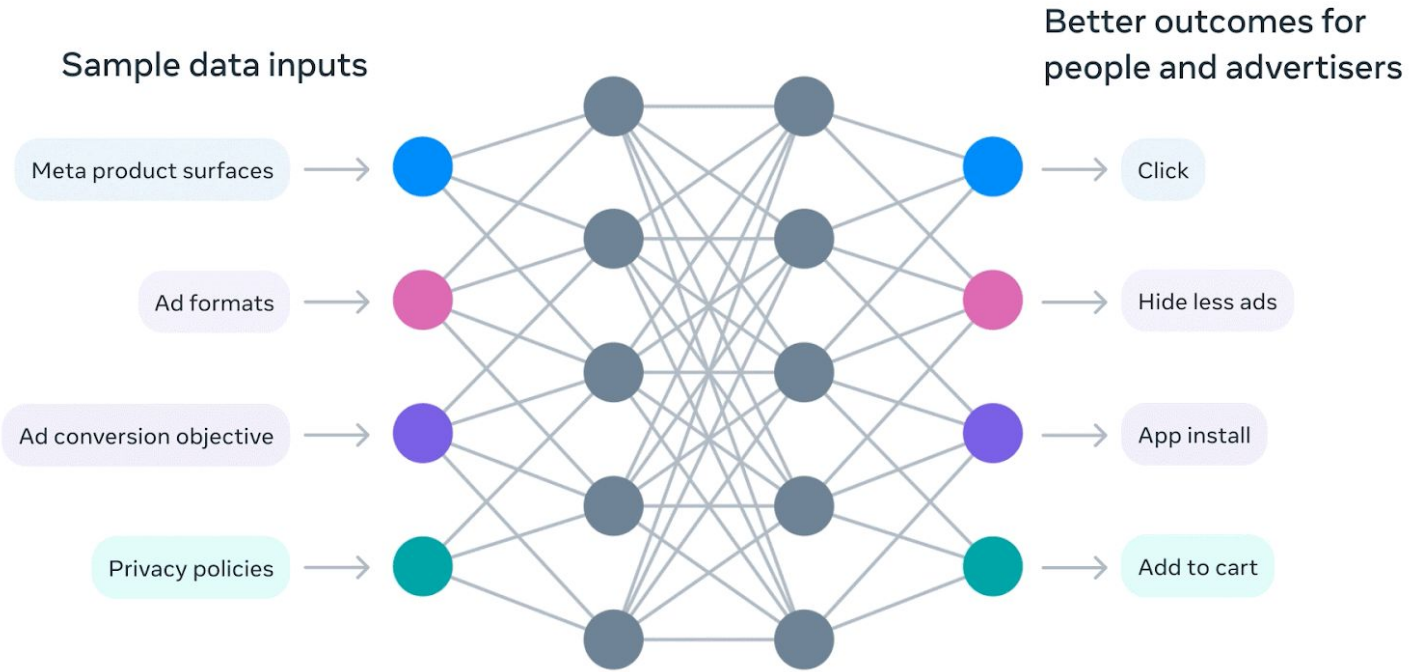
Ads Experimentation at Meta

Personalized Ads

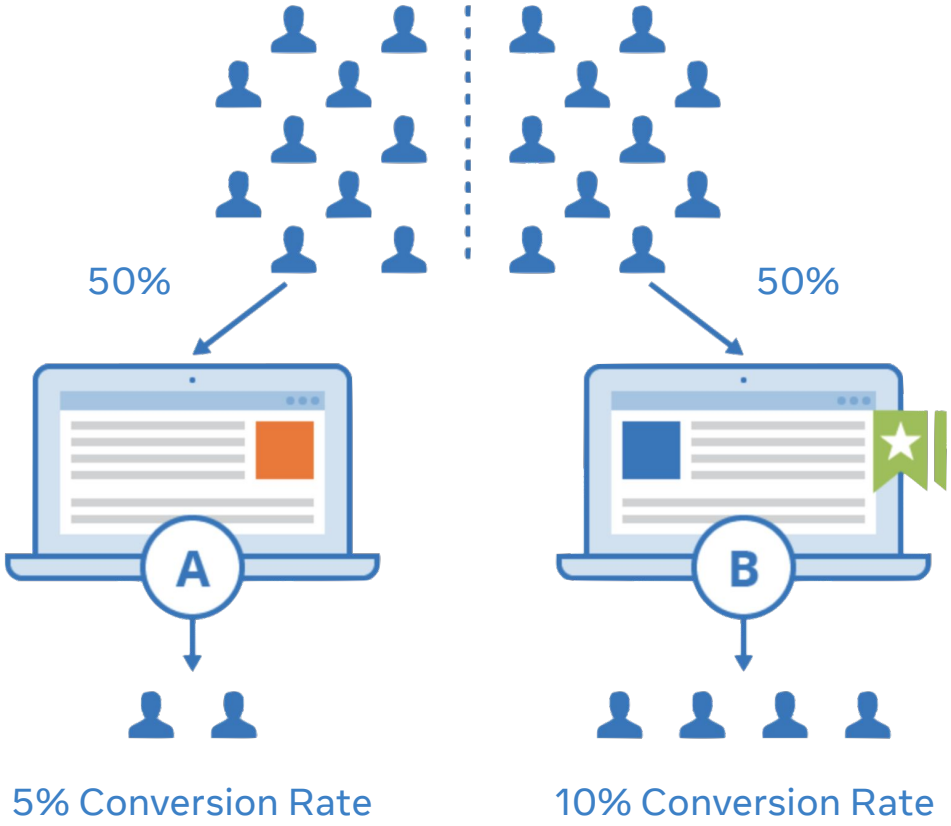


Advertiser Constraints




Sample data inputs
Meta product surfaces
Ad formats
Ad conversion objective
Privacy policies
Better outcomes for people and advertisers
Click
Hide less ads
App install
Add to cart

Experimentation


50%
50%
A
B
5% Conversion Rate
10% Conversion Rate

1000s

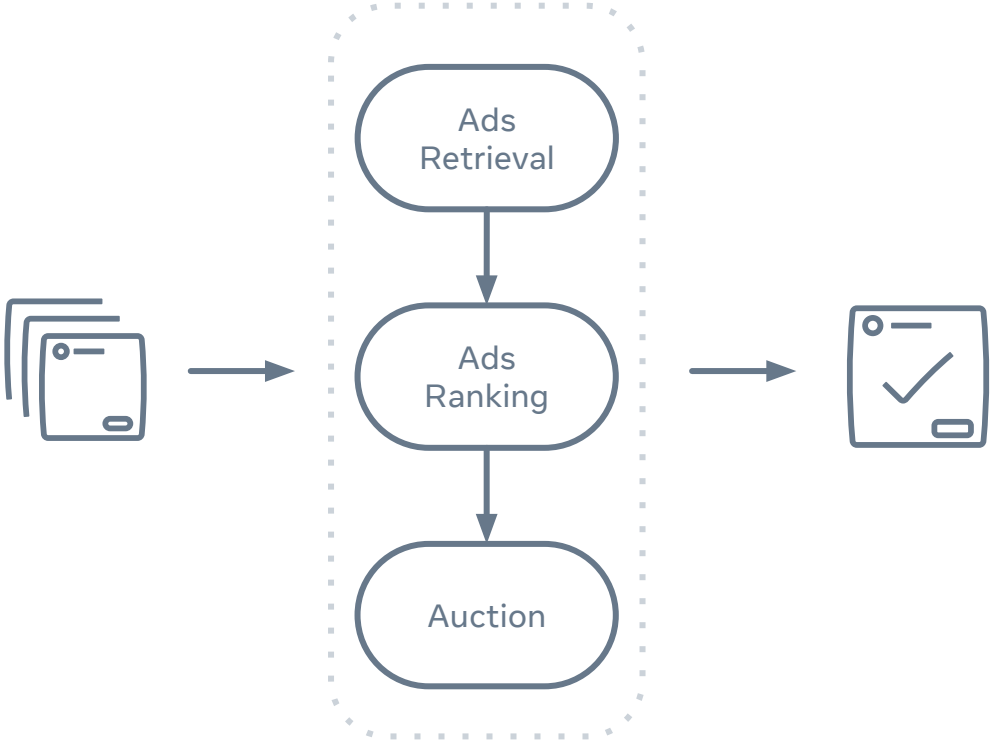
Concurrent Experiments

1000s

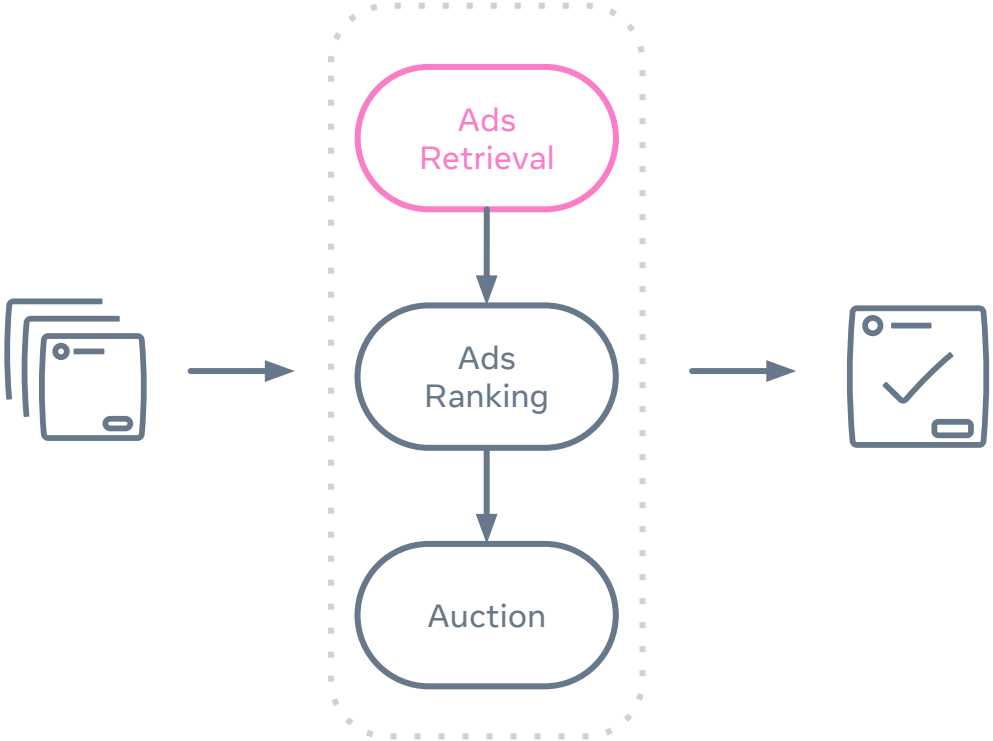
Engineers

Experimentation as the Default

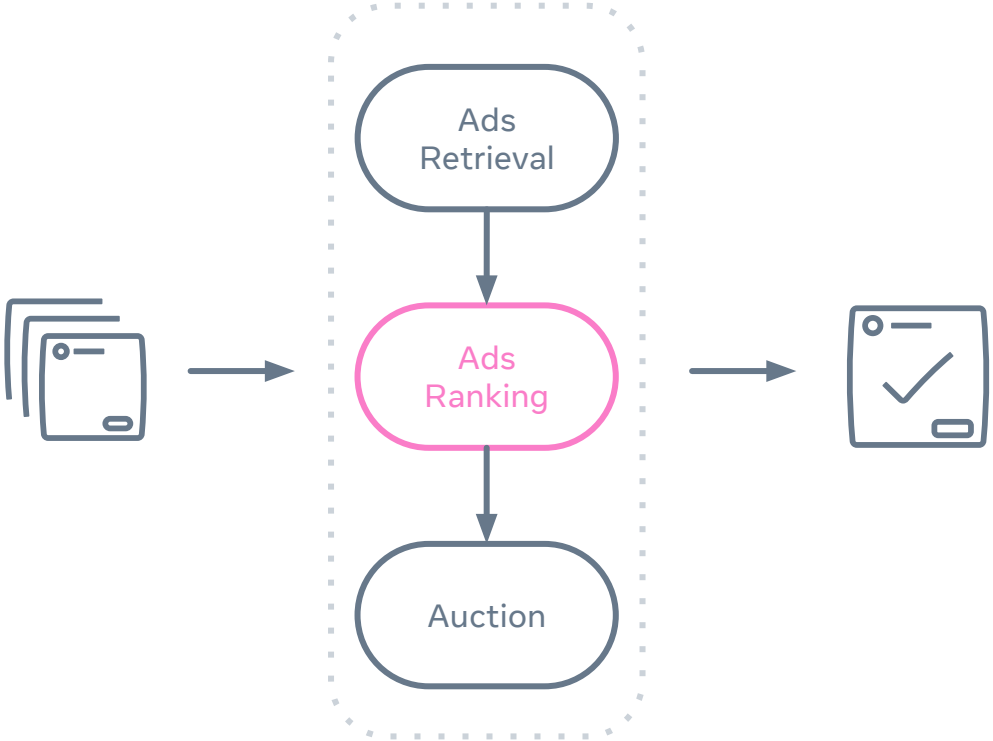
EXPERIMENTATION AS THE DEFAULT



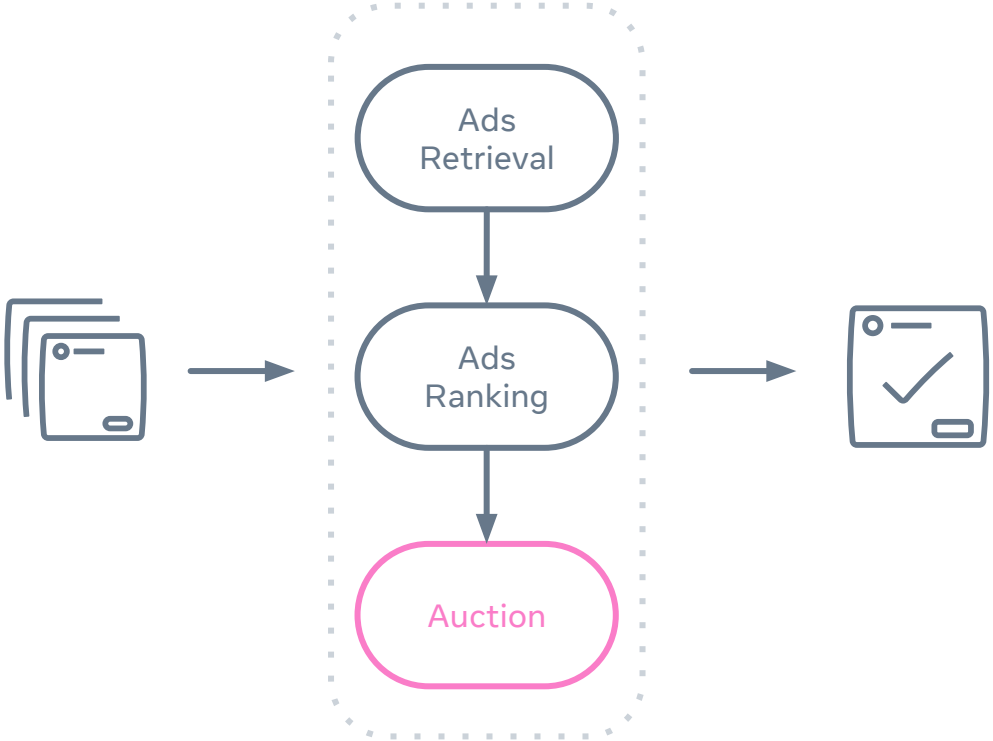
EXPERIMENTATION AS THE DEFAULT



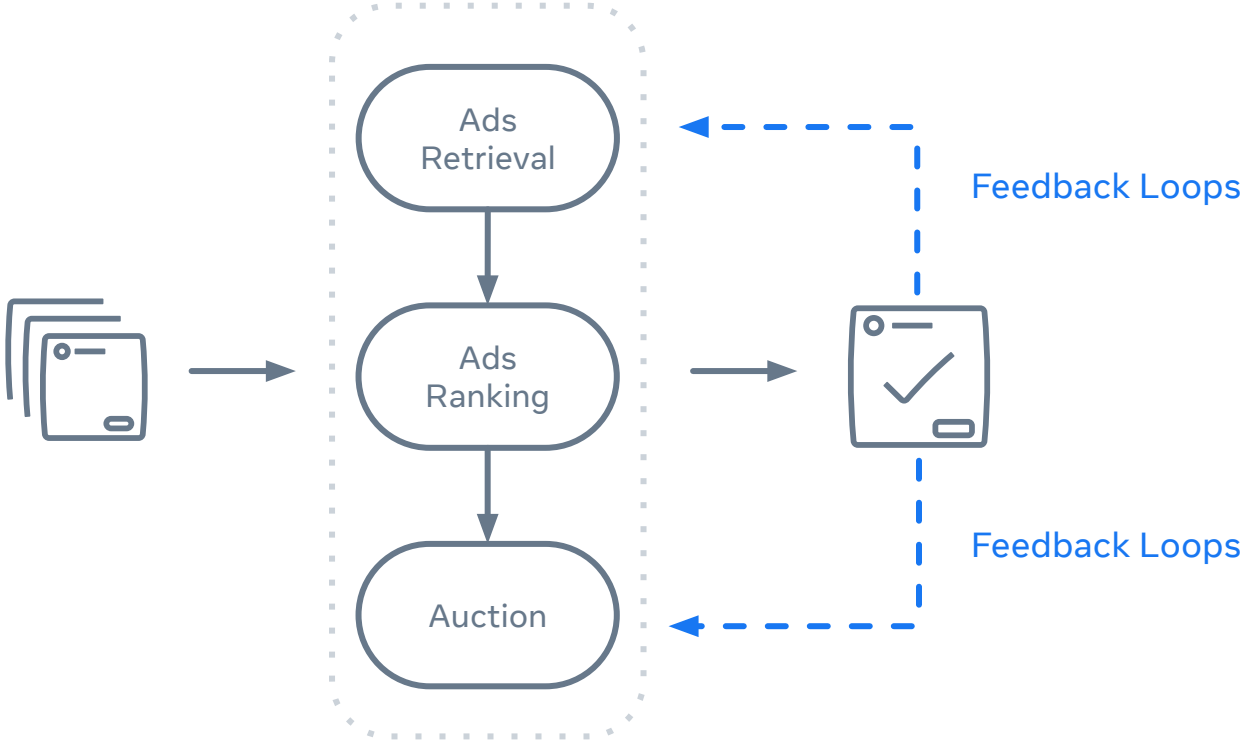
EXPERIMENTATION AS THE DEFAULT



EXPERIMENTATION AS THE DEFAULT

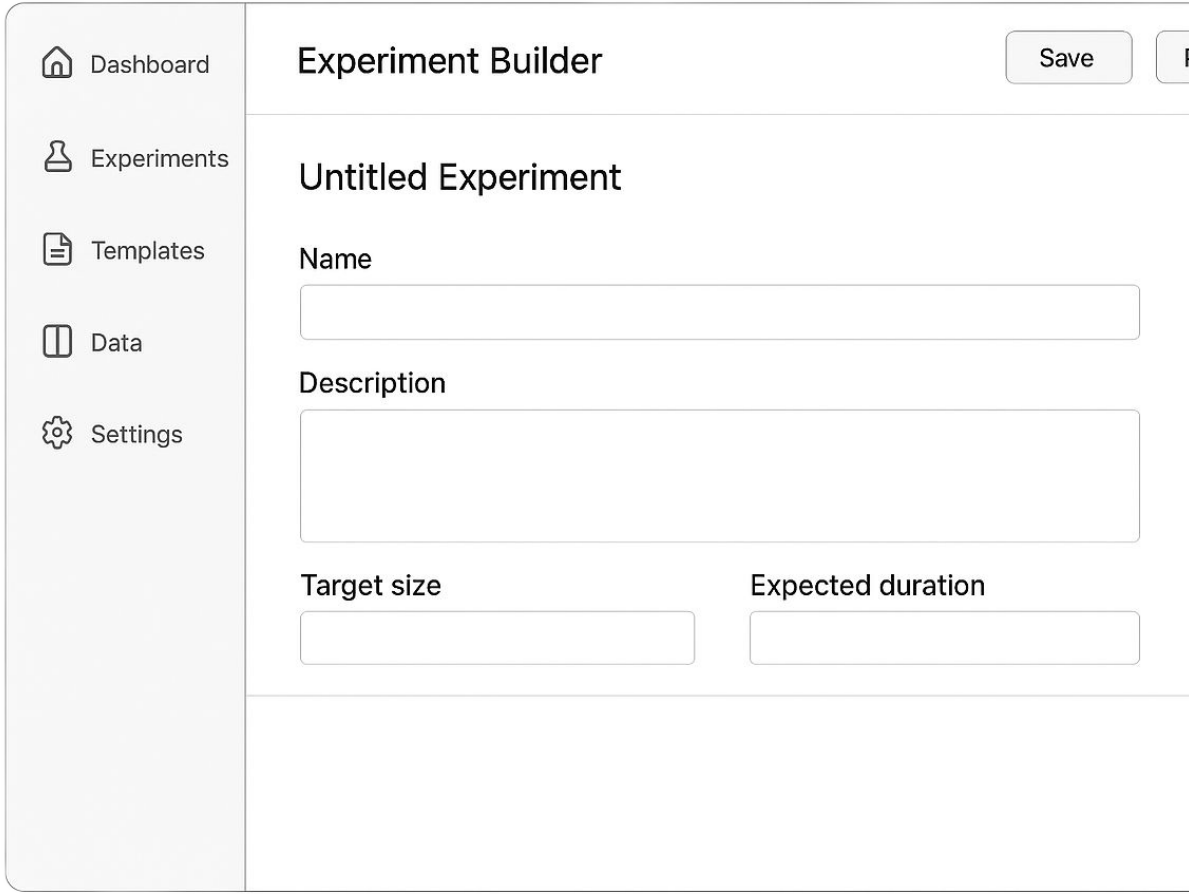


EXPERIMENTATION AS THE DEFAULT



What does an Experiment look like?

Experiment Set Up

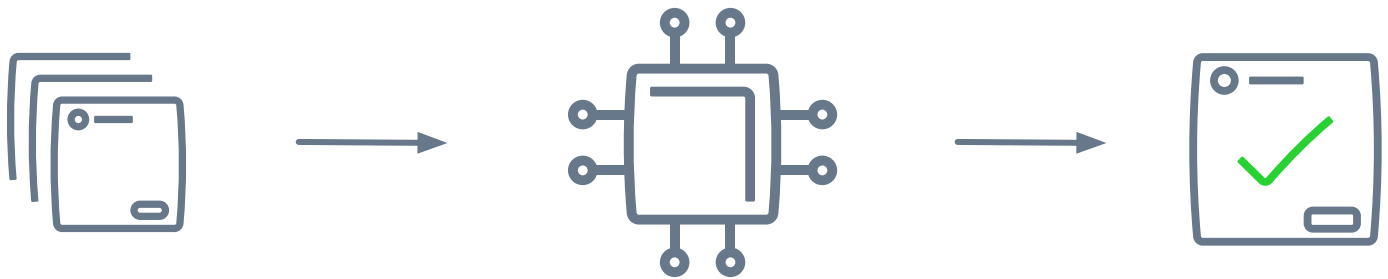


High-Level API

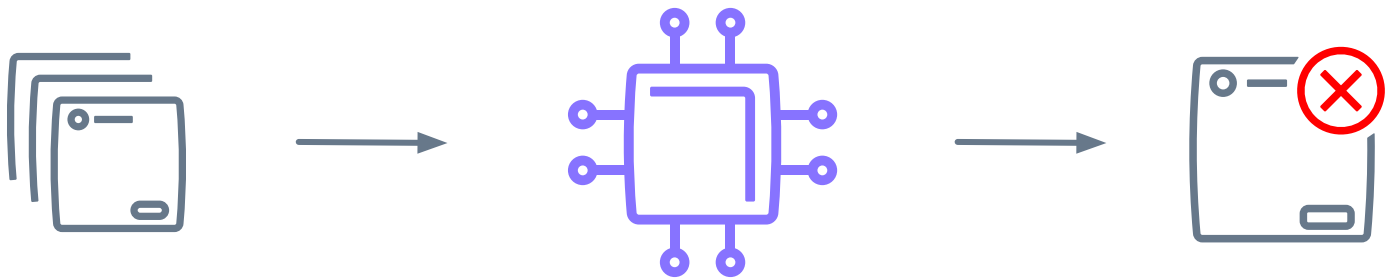
```
if (Experiment::isEnabled(request, experimentId)) {
    newBehaviour();
} else {
    existingBehaviour();
}
```

Failure Modes

EXPERIMENTATION AS THE DEFAULT

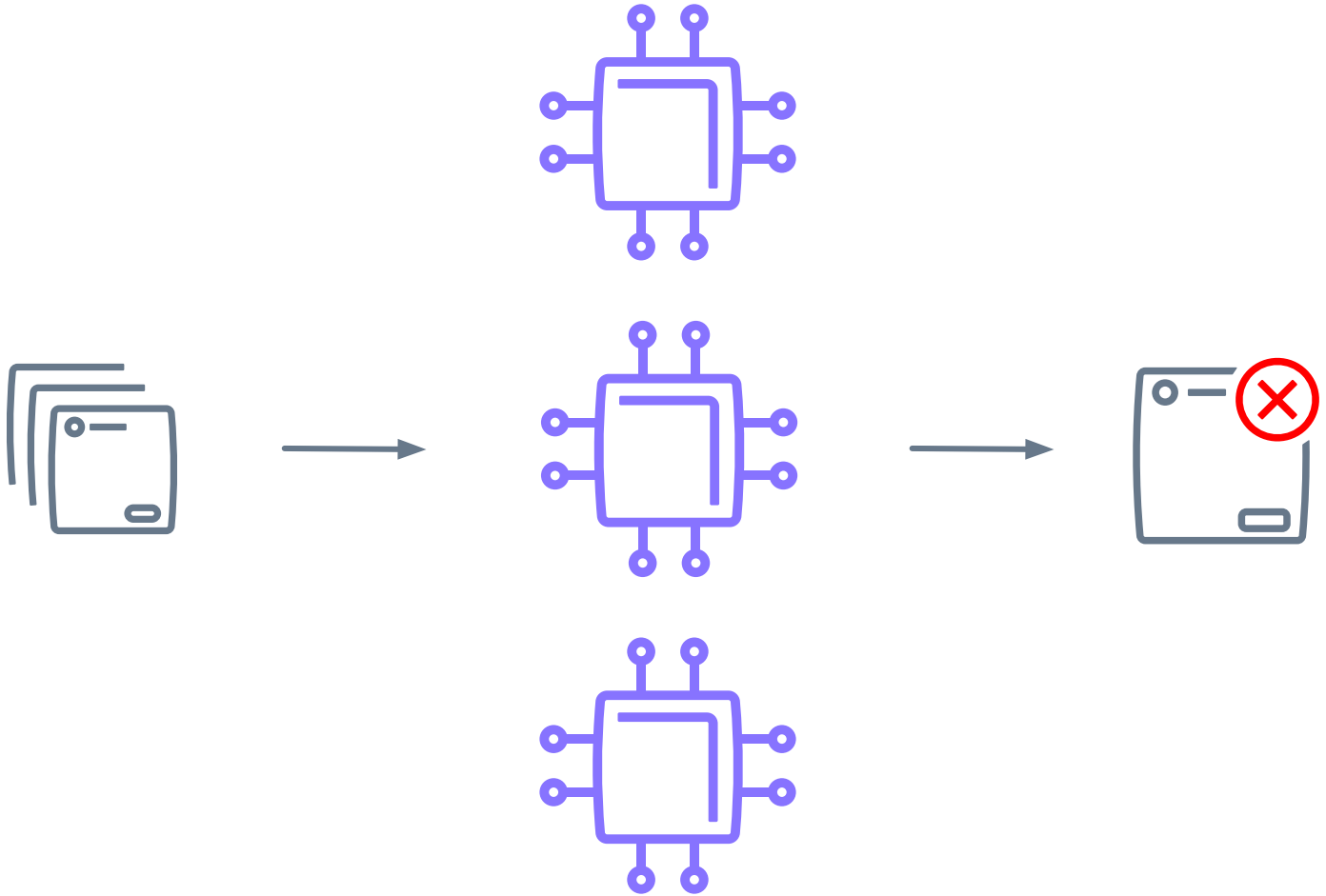


EXPERIMENTATION AS THE DEFAULT

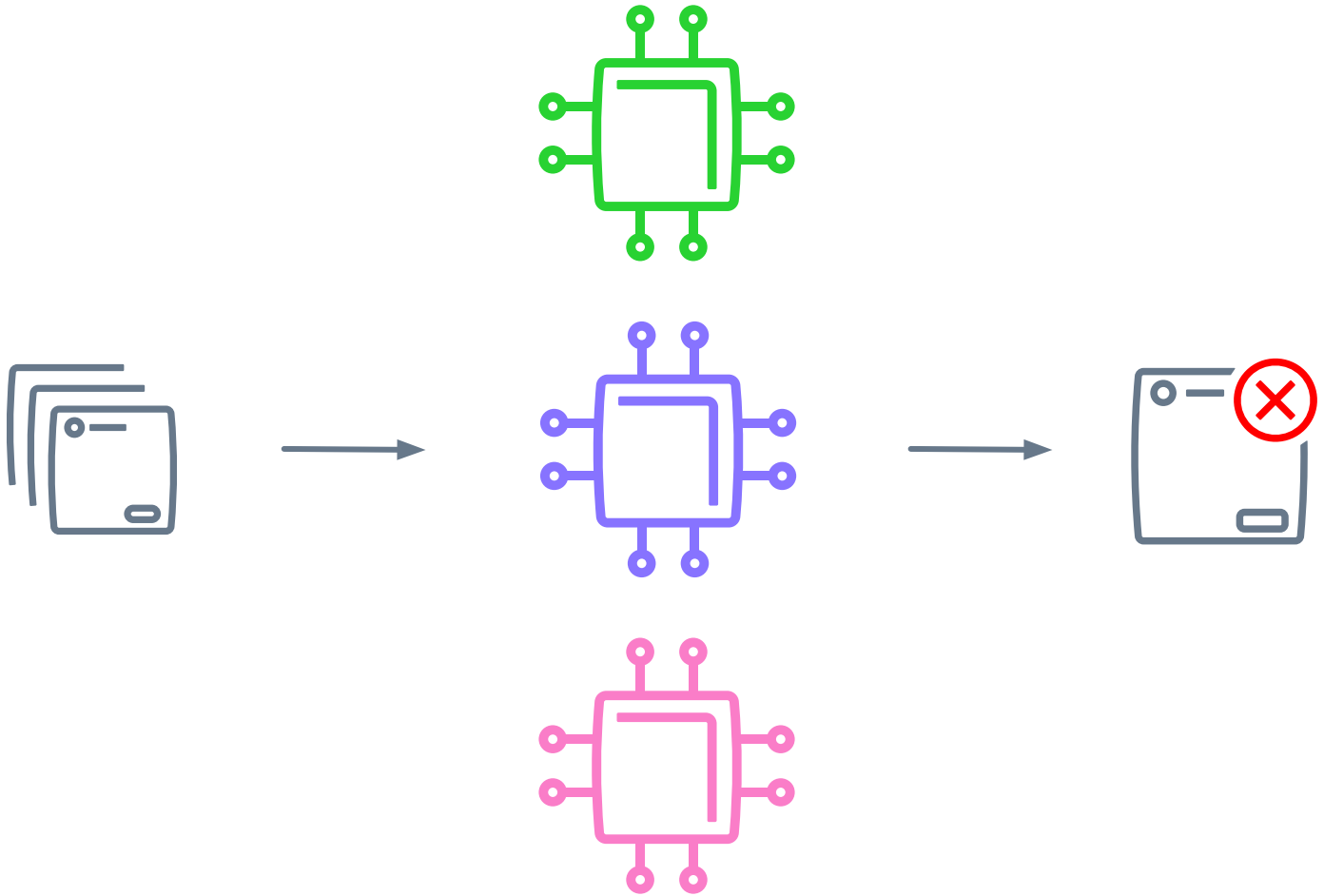



EMERGENCY

EXPERIMENTATION AS THE DEFAULT



EXPERIMENTATION AS THE DEFAULT



Failure Scenarios

01 ML Model Mispredictions

02 Infrastructure Outages

03 Experiment Allocation & Measurement

Business & People Challenges

ADS EXPERIMENTATION AT META

10s Bn

Meta's yearly Ad Revenue

3.4Bn

People across Meta Platform's core products

1000s

Engineers relying on Experimentation day-to-day

“**Everyone**’s dog becomes **no one's** responsibility to feed”

ANCIENT SRE PROVERB

Experiment Safety

Investment Areas

01 Change Safety & Isolation

02 Monitoring & Observability

03 Rapid Mitigation

04 People & Processes

Investment Areas

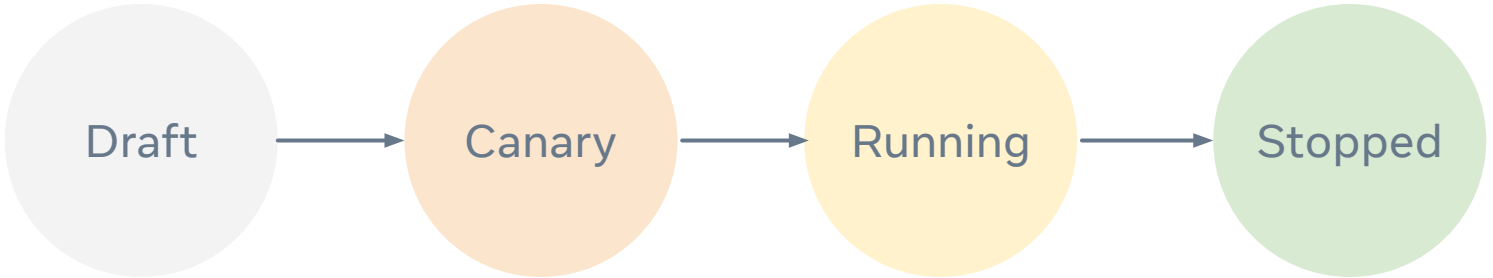
01 **Change Safety & Isolation**

02 Monitoring & Observability

03 Rapid Mitigation

04 People & Processes

Experiment Lifecycle



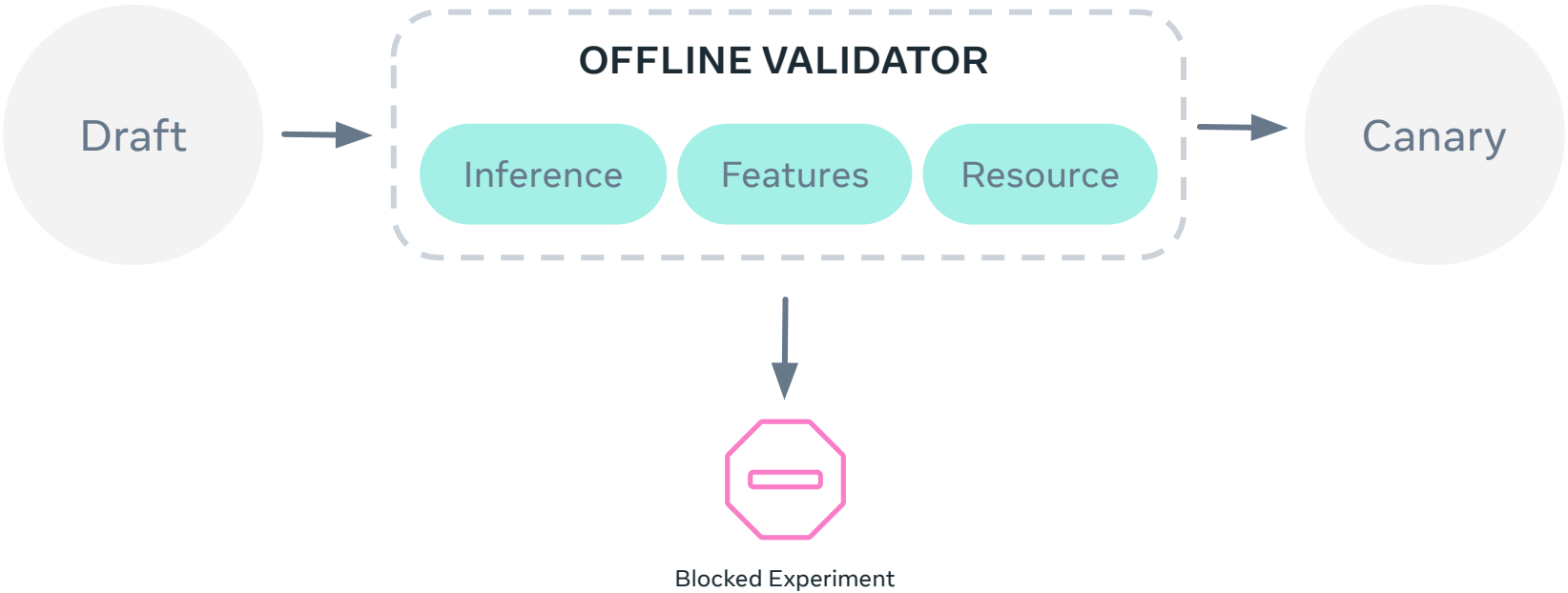
Automatic Experiment Configuration

01 UI-driven Model Configuration

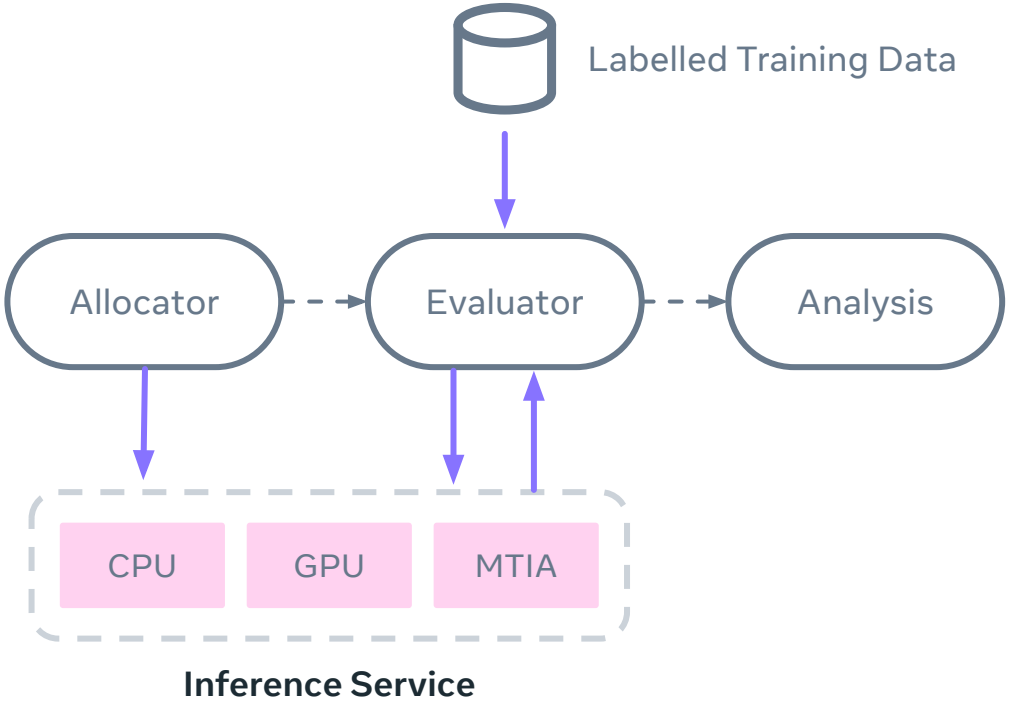
02 Automatic Model Loading & Scaling

03 Property-level Validations

Offline Testing

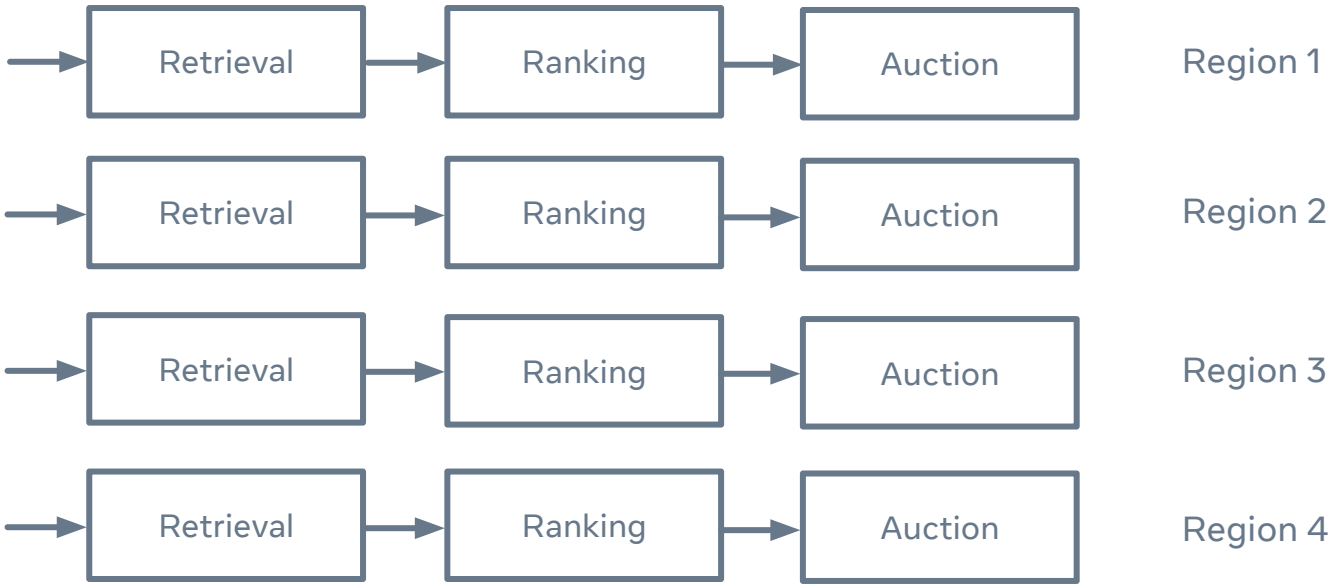


Offline Testing: Model Inference

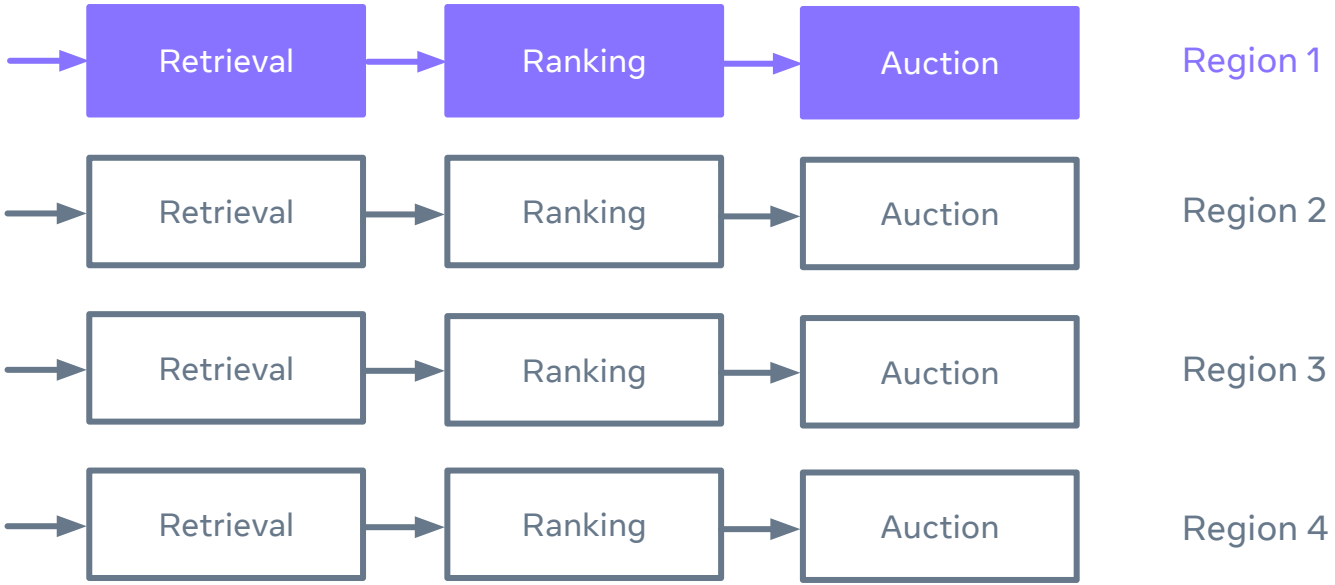


Offline Testing: Resource Usage Estimation

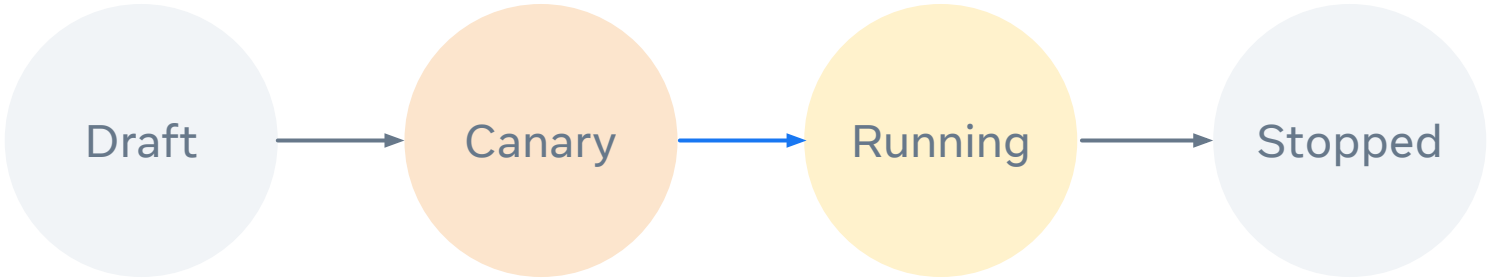
Isolated Environments



Isolated Environments

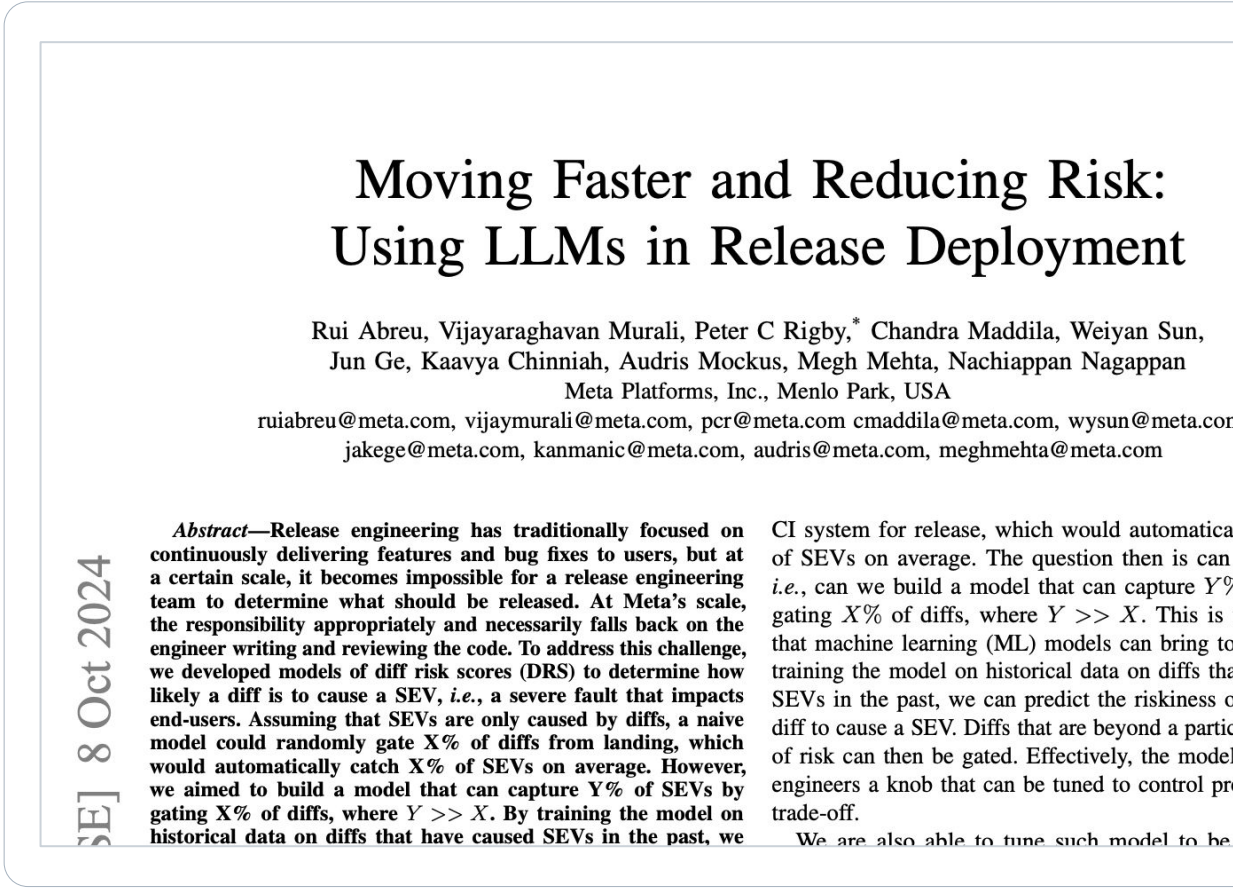


Start Experiment?



Risk Scoring

https://fb.me/risk-score-paper



Moving Faster and Reducing Risk:
Using LLMs in Release Deployment

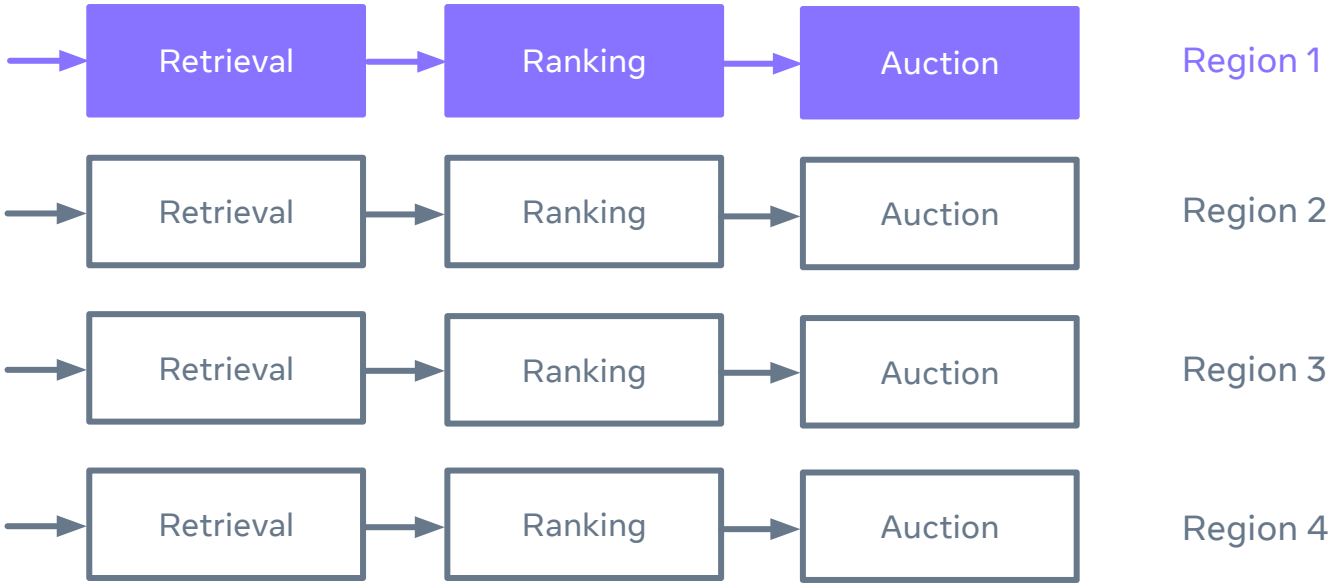
Rui Abreu, Vijayaraghavan Murali, Peter C Rigby,* Chandra Maddila, Weiyan Sun,
Jun Ge, Kaavya Chinniah, Audris Mockus, Megh Mehta, Nachiappan Nagappan
Meta Platforms, Inc., Menlo Park, USA
ruiabreu@meta.com, vijaymurali@meta.com, pcr@meta.com cmaddila@meta.com, wysun@meta.com
jakege@meta.com, kanmanic@meta.com, audris@meta.com, meghmehta@meta.com

***Abstract*—Release engineering has traditionally focused on continuously delivering features and bug fixes to users, but at a certain scale, it becomes impossible for a release engineering team to determine what should be released. At Meta's scale, the responsibility appropriately and necessarily falls back on the engineer writing and reviewing the code. To address this challenge, we developed models of diff risk scores (DRS) to determine how likely a diff is to cause a SEV, *i.e.*, a severe fault that impacts end-users. Assuming that SEVs are only caused by diffs, a naive model could randomly gate X% of diffs from landing, which would automatically catch X% of SEVs on average. However, we aimed to build a model that can capture Y% of SEVs by gating X% of diffs, where $Y >> X$. By training the model on historical data on diffs that have caused SEVs in the past, we**

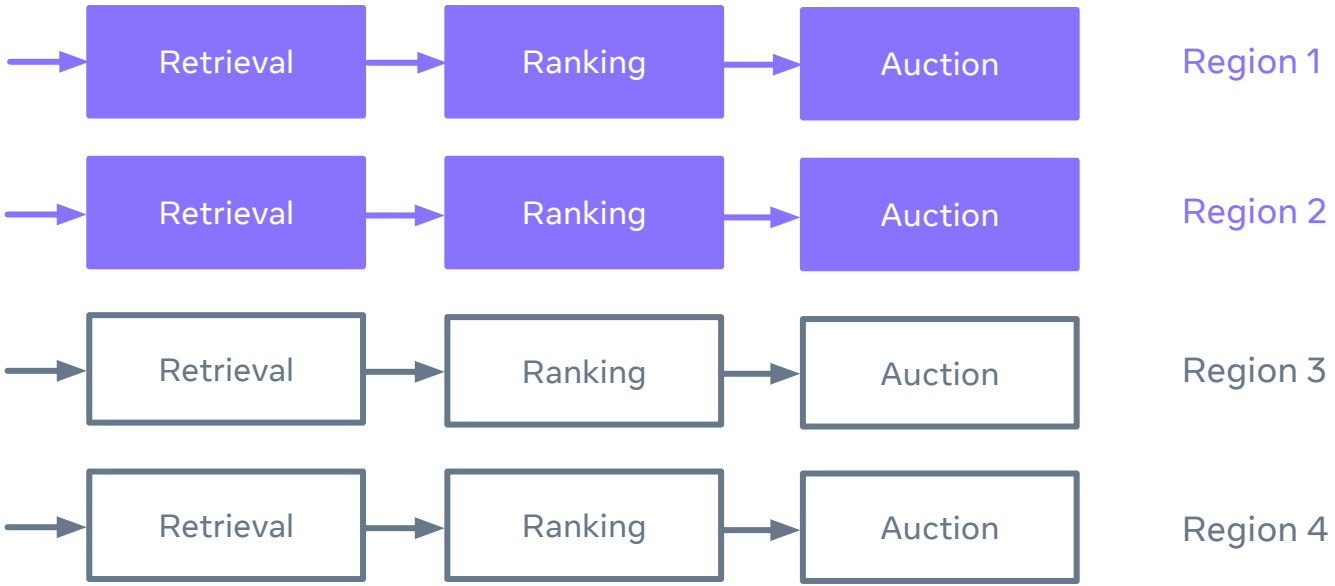
CI system for release, which would automatica
of SEVs on average. The question then is can
i.e., can we build a model that can capture $Y\%$
gating $X\%$ of diffs, where $Y >> X$. This is
that machine learning (ML) models can bring to
training the model on historical data on diffs tha
SEVs in the past, we can predict the riskiness o
diff to cause a SEV. Diffs that are beyond a partic
of risk can then be gated. Effectively, the model
engineers a knob that can be tuned to control pr
trade-off.

We are also able to tune such model to be

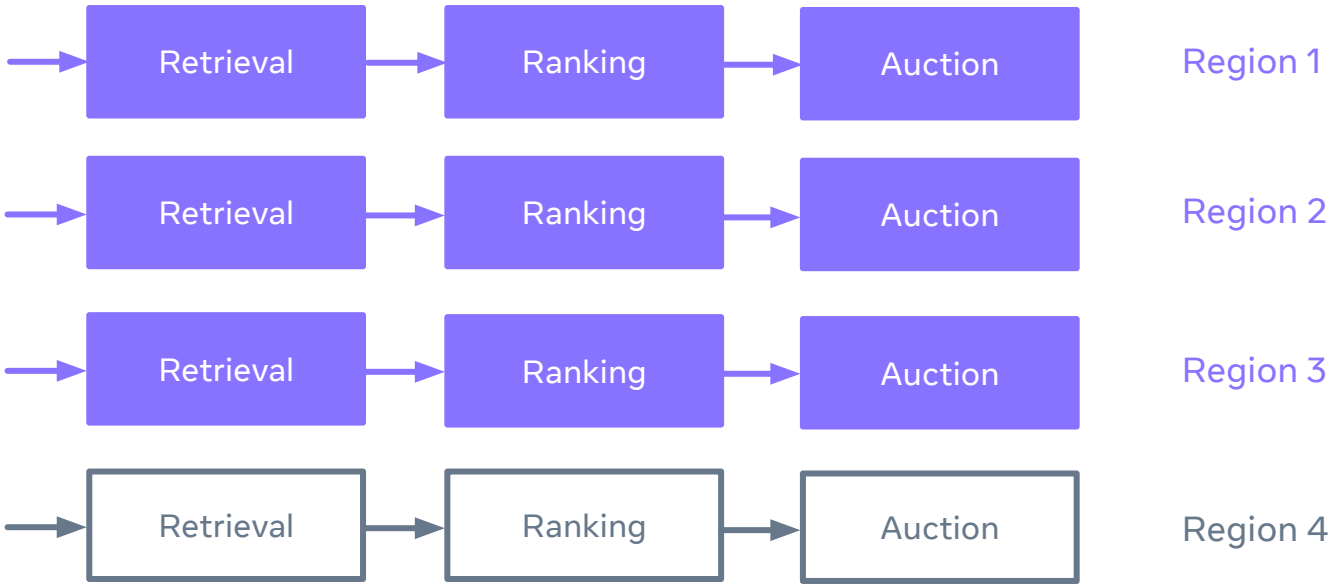
Gradual Rollout & Health Checks



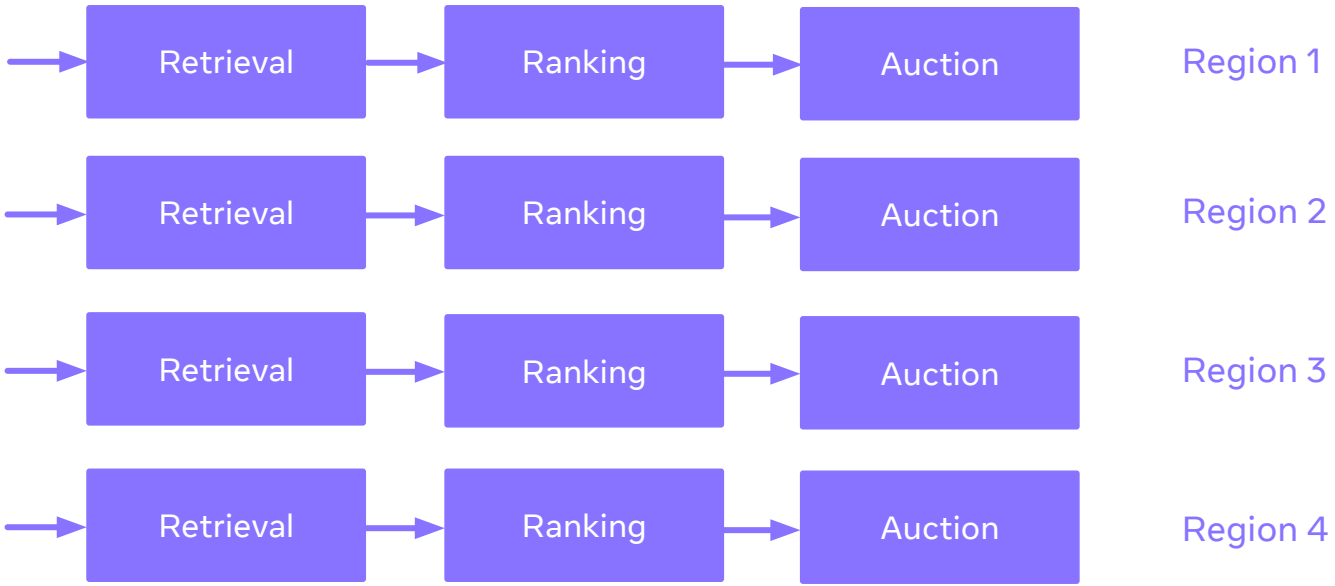
Gradual Rollout & Health Checks



Gradual Rollout & Health Checks



Gradual Rollout & Health Checks



Investment Areas

01 Change Safety & Isolation

02 Monitoring & Observability

03 Rapid Mitigation

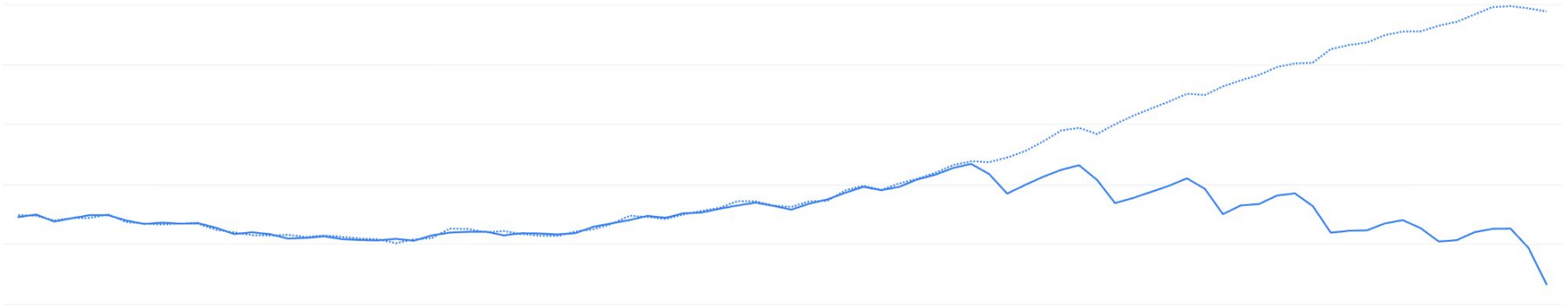
04 People & Processes

Experiment Monitoring

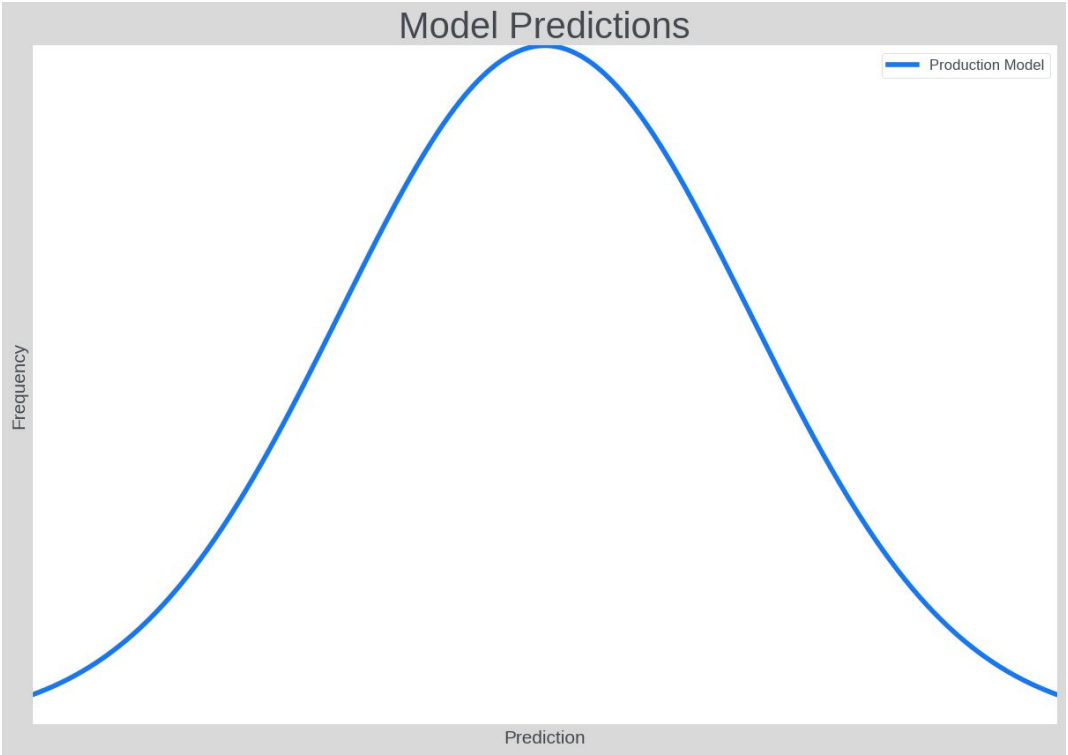
01 Automatic Experiment Onboarding

02 Leveraging Breakdowns of Global Business Metrics

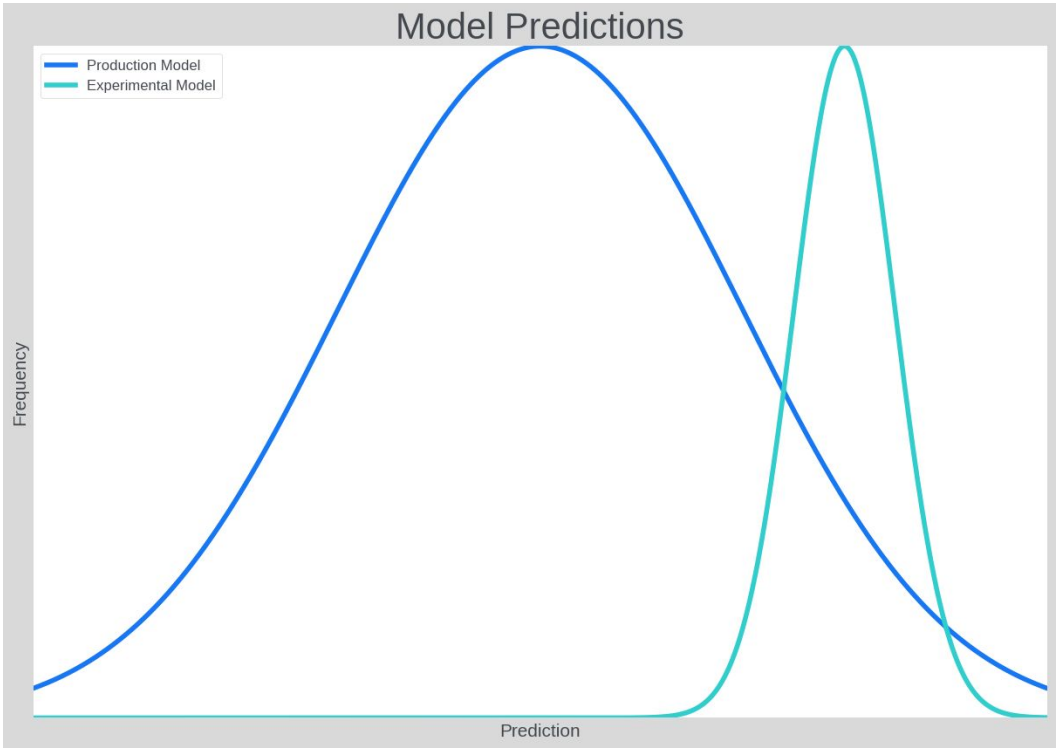
Experiment Monitoring



ML Prediction Monitoring



ML Prediction Monitoring



Investment Areas

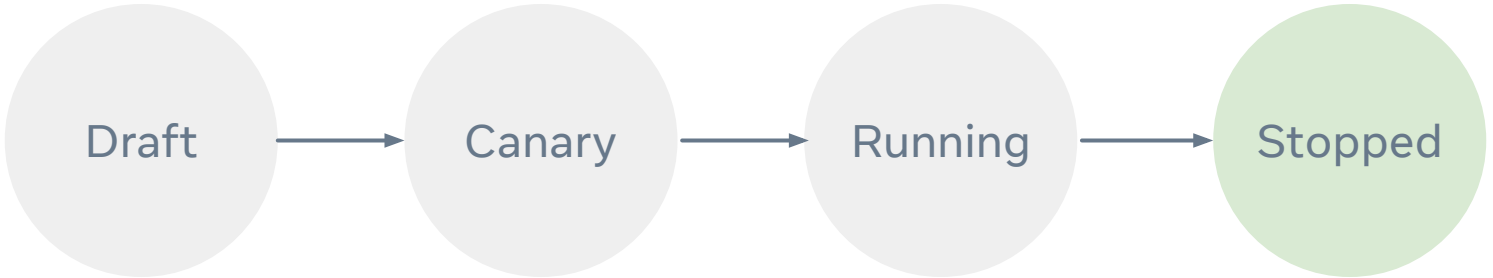
01 Change Safety & Isolation

02 Monitoring & Observability

03 Rapid Mitigation

04 People & Processes

Mitigation Workflows



LLM-assisted Mitigation Workflows

After investigating the revenue regression starting at 08:32 in **experiment 879** ("Ranking Model Improvements"), I have concluded that:

- ✅ No issues were found with model prediction
- ✅ No issues were found with model calibration
- ✅ No issues were found with training normalized entropy
- ✅ No issues were found with model staleness
- ❌ **Issue identified with increased model fallback**, see **here** for more details

Big Red Button

Investment Areas

01 Change Safety & Isolation

02 Monitoring & Observability

03 Rapid Mitigation

04 People & Processes

Mechanisms

01 Accountability & Ownership

02 Collaboration across Teams

03 Methodical Analysis

Lessons & Learnings

LESSON 1

Solve Complex Reliability Problems with Data-Driven Insights

LESSON 2

Balance Reliability with Developer Velocity

LESSON 3

High-Quality Monitoring and Early Guardrails


Meta