TrackIT — AI-Powered Notebook Lineage & Experiment Summaries

Role: Product · Research · UX · System Design · Applied ML

Why This Problem Matters Now

Experimentation Volume Rising

AI teams are scaling =more experiments than ever.

Notebook-First Workflows Dominate

Feature engineering decisions drive model outcomes.

Preprocessing Impacts Model Quality

Most DS workflows happen in Jupyter/Colab.

No Tool Captures Preprocessing Lineage

Notebook transformations remain hard to trace

Reproducibility Problems

Struggling to keep track of experiments

Problem Statement

Data scientists perform rapid preprocessing and feature engineering inside notebooks, but a lack reliable record of how data was transformed across experiments.

Losing Context Across Sessions

Painful handoffs when sharing notebooks

Hard to track or reproduce experiments

Poor reproducibility for audits / debugging

“Model metrics are tracked. Data transformations are not.”

User Research & Evidence

Research Approach

- Mom Test–inspired qualitative discovery
- Focused on behaviors, not solution validation

Consistent Patterns Across 40+ Comments

- **~70%** reported losing track of preprocessing logic
- **2+ hours** per experiment commonly lost retracing steps
- Preprocessing changes rarely documented consistently
- Reproducibility breaks when notebooks evolve or teams collaborate
- Nearly all users rely on fragile, manual workarounds

Research Channels*

- Reddit (r/MLQuestions, r/AskDataScience)
- Hacker News (Ask HN / discussion threads)
- Slack DS & ML communities
- 2 in-depth 1:1 interviews with practicing data scientists
- Multiple channels reduced sampling bias and strengthened signal confidence.

Observed Workarounds

- Custom logging inside functions
- MLflow notes used as pseudo-lineage
- “final_v3 / final_final” script folders
- Spreadsheets tracking experiments
- SQL temp tables
- YAML-based MLTable configs

Users have tools for models and metrics — but not for preprocessing lineage.

*Sources: Reddit (r/MLQuestions, r/AskDataScience), Hacker News threads, Slack DS communities, Links in appendix.

Key Insights & Problem Themes

Insight 1 — Preprocessing logic is routinely lost
During rapid notebook experimentation, users frequently lose track of filtering, feature engineering, and transformation steps.

Insight 2 — Experimentation is fragmented and poorly documented
Notebook cells are overwritten, experiments re-run without context, and rationale for changes is rarely preserved.

Insight 3 — Reproducibility breaks down in collaborative settings
When notebooks evolve or multiple people contribute, teams struggle to reconstruct how datasets were produced.

Insight 4 — Existing tools leave a critical workflow gap
Tools like MLflow, Git, DVC, and Airflow track models or data versions, but not transformation-level lineage inside notebooks.

The core problem is not model tracking —
it's invisible data transformation during notebook experimentation.

Market Gap / Existing Analysis

Existing tools optimize for production and governance — not for messy, exploratory notebook workflows where preprocessing decisions are made.

Category	Examples	What They Track	What They Don’t Track	Gap Track IT Fills
Experiment Tracking	**MLflow, W&B**	**Models, metrics**	**Preprocessing lineage**	**Notebook-first lineage**
Data Versioning	**DVC, LakeFS**	**Dataset snapshots**	**Step-by-step transformations**	**Transformation evolution**
Orchestration	**Airflow, Prefect**	**Pipelines, DAGs**	**Ad-hoc experiments**	**Early-stage exploration**
Enterprise Lineage	**DataHub**	**System flows**	**Notebook logic**	**Micro-level lineage**

Solution Exploration

Evaluation Criteria: User Friction | Value Delivery | Technical Feasibility | Scalability

Option A — Python Decorator Library

Wrap preprocessing functions with decorators to log transformations.

❌ High friction, incomplete lineage

Option B — Custom Notebook IDE

Build a new notebook environment with built-in lineage + LLM summaries.

❌ High effort, low adoption

Option C — Local Background Agent

Monitor notebook execution locally without workflow changes.

Best balance of value, feasibility, and adoption

Option D — Browser Extension

Intercept notebook UI events in the browser.

❌ Brittle, shallow lineage, Difficult to implement.

Why Option C Was Chosen (MVP Decision)

Option	User Friction	Value	Feasibility	Scalability	Verdict
A: Decorators	High	Low	High	Medium	✗
B: Custom IDE	High	Medium	Very Low	High	✗
C: Background Agent	Low	High	Medium	High	MVP
D: Browser Extension	Medium	Low	Low	Low	✗

Option C uniquely satisfies all four criteria:

- Zero workflow change (critical for adoption)
- Captures true cell-level preprocessing lineage
- Feasible for a solo builder in 4–8 weeks
- Forms a foundation for RAG, Q&A, and team features

MVP Definition

MVP Value Hypothesis

Automatic lineage tracking and LLM summaries reduce cognitive load and help users reconstruct experiments faster — without workflow changes."

MVP Definition

Notebook Discovery UI

- Lists local notebooks
- One-click tracking activation

Why: Low friction onboarding

Passive Notebook Tracking

- Monitors execution and saves
- Captures cell-level lineage automatically

Why: Zero workflow change (critical for adoption)

Local-First Log Storage

- Structured lineage stored locally
- No data leaves the machine

Why: Trust, privacy

LLM-Generated Summaries

- Converts raw logs into readable narratives

Why: Validates insight + time savings

What the MVP Explicitly Does NOT Include

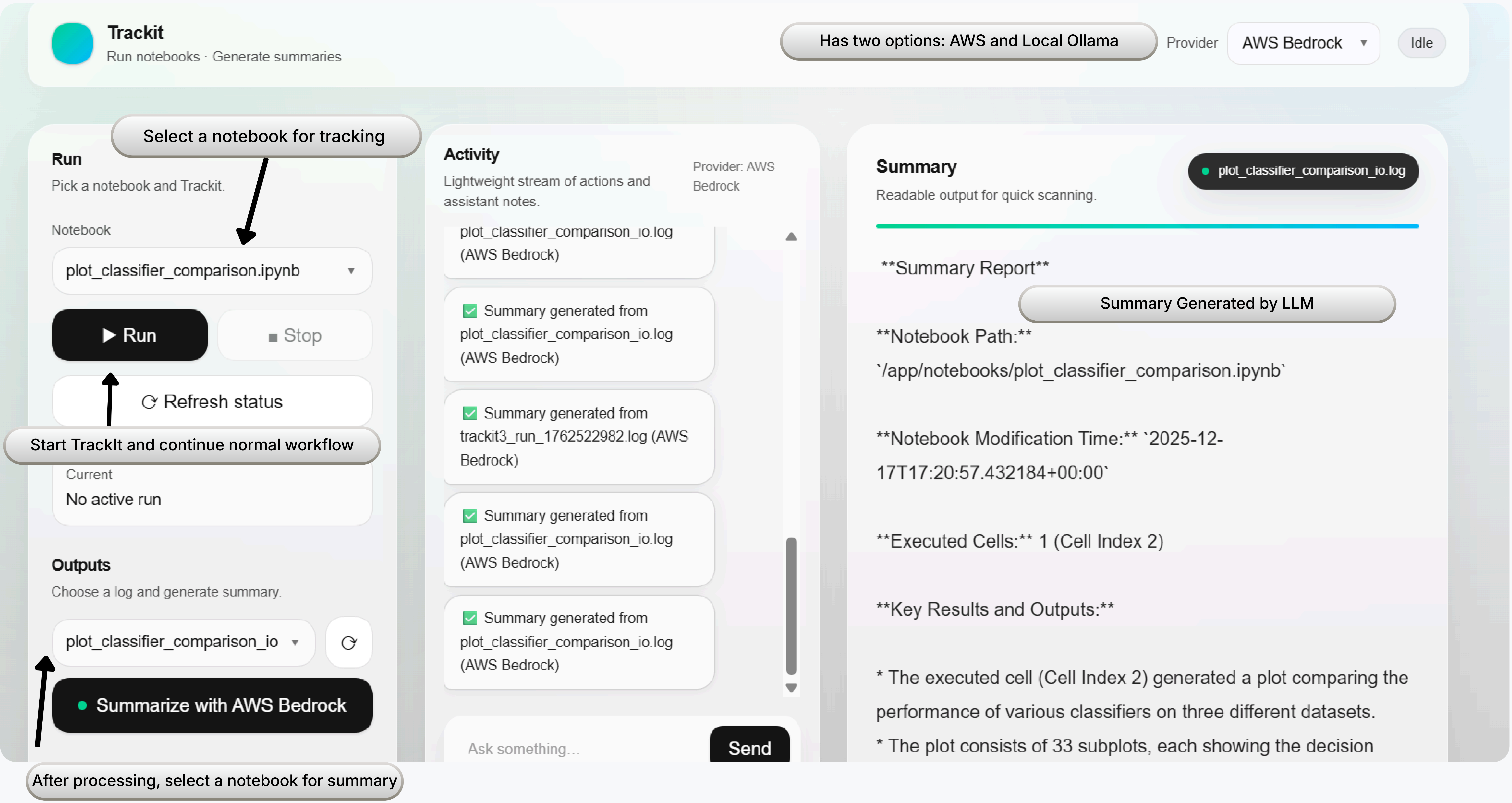
Excluded from MVP

- Chat / Q&A
- RAG pipelines
- Multi-LLM routing
- Vector databases
- Collaboration / accounts
- Report or PPT generation
- Cloud sync
- Guardrails & safety layers

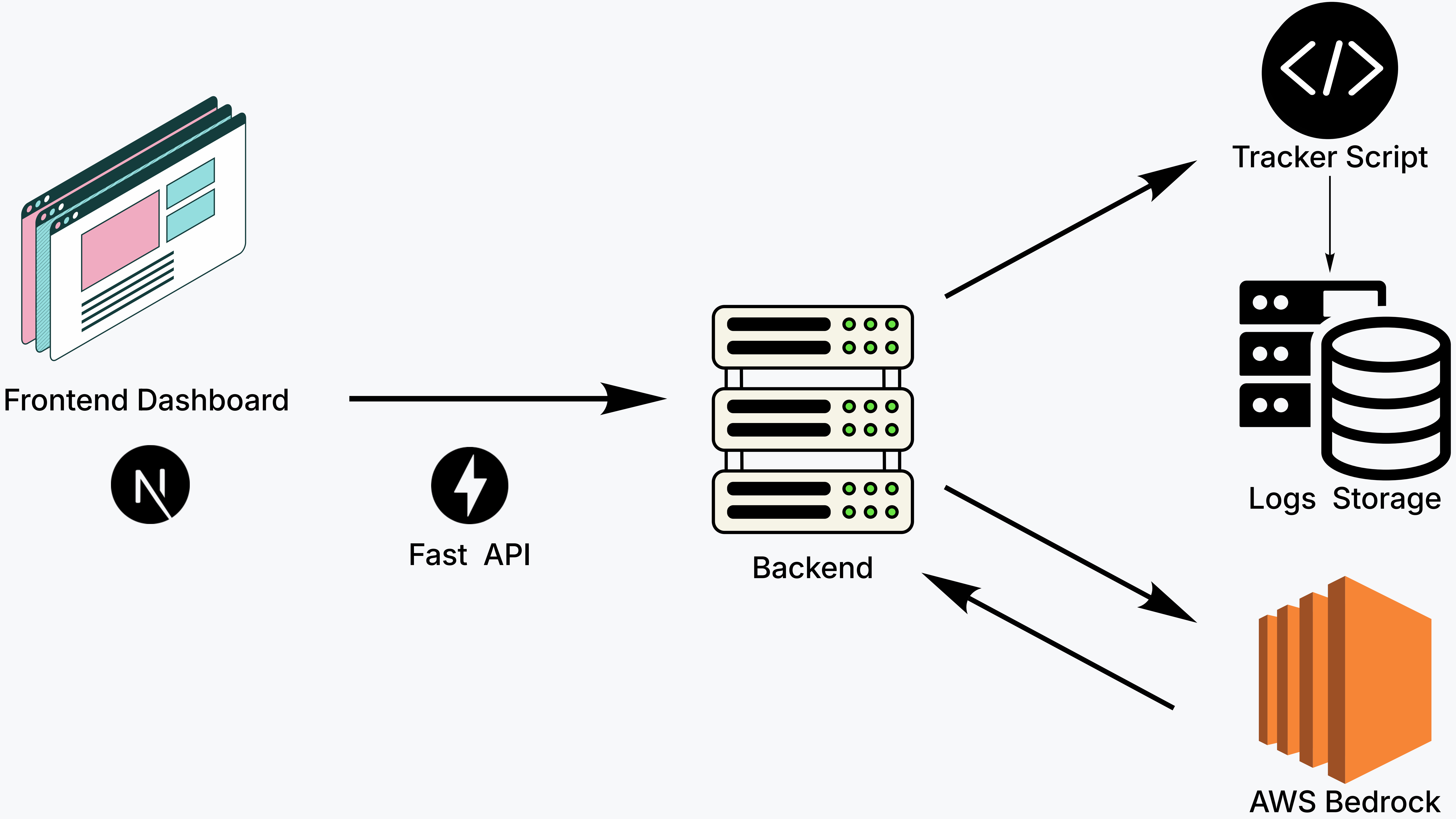
Why This Scope Is Right

- Validates desirability before scaling complexity
- Minimizes adoption friction and engineering risk
- Builds foundations for future AI features

Demo


Trackit
Run notebooks · Generate summaries
Has two options: AWS and Local Ollama
Provider
AWS Bedrock
Idle
Select a notebook for tracking
Run
Pick a notebook and Trackit.
Notebook
plot_classifier_comparison.ipynb
Run
Stop
Refresh status
Start TrackIt and continue normal workflow
Current
No active run
Outputs
Choose a log and generate summary.
plot_classifier_comparison_io
Summarize with AWS Bedrock
After processing, select a notebook for summary
Activity
Lightweight stream of actions and assistant notes.
Provider: AWS Bedrock
plot_classifier_comparison_io.log (AWS Bedrock)
Summary generated from plot_classifier_comparison_io.log (AWS Bedrock)
Summary generated from trackit3_run_1762522982.log (AWS Bedrock)
Summary generated from plot_classifier_comparison_io.log (AWS Bedrock)
Summary generated from plot_classifier_comparison_io.log (AWS Bedrock)
Ask something...
Send
Summary
Readable output for quick scanning.
plot_classifier_comparison_io.log
Summary Report
Summary Generated by LLM
Notebook Path: `/app/notebooks/plot_classifier_comparison.ipynb`
Notebook Modification Time: `2025-12-17T17:20:57.432184+00:00`
Executed Cells: 1 (Cell Index 2)
Key Results and Outputs:
* The executed cell (Cell Index 2) generated a plot comparing the performance of various classifiers on three different datasets.
* The plot consists of 33 subplots, each showing the decision

Technical Architecture



Success Metrics & Validation Signals

North Star Metric

Time Saved Reconstructing Past Experiments

- Definition: Average reduction in time required for a user to understand or reproduce a previous preprocessing workflow.
- Why this metric: Directly measures whether TrackIT reduces cognitive load — the core value hypothesis.

1. Time-to-Reconstruction

- Time required to explain or reproduce a past experiment
- Measured before vs. after using TrackIT
- Signals direct productivity gains.

2. Summary Usefulness Score

- User-rated clarity and completeness of LLM-generated summaries
- Simple 1–5 rating after viewing a summary
- Validates whether the LLM adds real insight.

3. Lineage Coverage Rate

- % of preprocessing steps automatically captured per notebook session
- Indicates quality and completeness of tracking.

4. Repeat Usage

- Do users generate summaries multiple times per notebook?
- Indicates perceived ongoing value
- Proxy for retention in an early MVP.

Risk & Assumptions

Key Assumptions to Validate

Bucket 1: Problem Severity

- Is lineage a burning pain or tolerated friction?
- Frequency vs impact uncertainty

Bucket 2: Market & Monetization

- Willingness to pay
- Individual vs team buyer
- Open-source vs SaaS

Bucket 3: Adoption & Behavior

- Local agent setup friction
- Silent demand vs small market
- Collaboration vs solo workflows

Competitive Risks

- Incumbents expand upstream
- Manual workarounds persist
- Market appears niche before expanding

Reflection and Learnings

Learning 1 — Hidden Pain Is Real Pain
Insight: Preprocessing chaos is normalized, not complained about.
Decision: I optimized for revealed behavior (lost time, workarounds), not loud requests.

Learning 2 — Adoption Beats Feature Richness
Insight: Every extra step (decorators, config, new IDEs) kills adoption.
Decision: I rejected “clean” but intrusive solutions in favor of a background agent.

Learning 3 — Local-First = Instant Trust
Insight: ML practitioners are highly sensitive to data privacy and control.
Decision: I designed TrackIT so no data leaves the machine by default.

Learning 4 — LLMs Win by Removing Cognitive Load
Insight: LLMs aren’t valuable because they’re “smart” — they’re valuable when they compress chaos.
Decision: I used LLMs for summarization, not prediction or automation.

Appendix A — Research Evidence

Hacker News new | past | comments | ask | show | jobs | submit login

Data Scientists and ML Engineers – How do you keep track of what you have tried?

1 point by arma97 63 days ago | hide | past | favorite | 2 comments

Hi all, I'm curious about how data scientists and ML engineers organize their work.

In your recent projects, how did you keep track of what you tried — preprocessing steps, model runs, or errors?

Did you have a process or system to look back at past experiments and learn from them?

Did you use any tools to help with this, like experiment tracking software? How did that work for you?

If you've ever struggled with this, what's been the hardest part?

brutus1213 63 days ago [–]

I am a manager. It is pretty bad in terms of tracking. Wandb looks great but really expensive (small team in a super large corp, pricing we were quoted plus the challenges of no-saas made this a no go for me). Been trying to get team members to mlflow and some adjacent tools but it is too hard to do it right.

arma97 62 days ago | parent [–]

Yeah, I totally relate — I often lose track of filtering logic, feature engineering, and other preprocessing steps. Those seem way harder to version and track than just model runs. I'm curious, how did your team build a workflow for managing that kind of complexity?

Guidelines | FAQ | Lists | API | Security | Legal | Apply to YC | Contact

Search:

https://news.ycombinator.com/item?id=45676265#45676676

r/askdatascience • 2mo ago

arma1997

Data Scientists & ML Engineers — How do you keep track of what you have tried?

Hi everyone! I'm curious about how data scientists and ML engineers organize their work.

1. Can you walk me through the last ML project you worked on? How did you track your preprocessing steps, model runs, and results?
2. How do you usually keep track and share updates with what you have tried with your teammates or managers? Do you have any tools, reports, or processes?
3. What's the hardest part about keeping track of experiments(preprocessing steps) or making sure others understand your work?
4. If you could change one thing about how you document or share experiments, what would it be?

*PS, I was referring more to preprocessing and other steps, which are not tracked by ML Flow and WandB

3 2 Share

https://www.reddit.com/r/askdatascience/comments/1odn05i/data_scientists_ml_engineers_how_do_you_keep/

You've been blocked by network security.

A_random_otter • 2mo ago • Edited 2mo ago

I use the pins package as my model + artifact registry and git for versioning (lots of branches 😅).

My code is pretty modular one script for data prep, one for modeling, one for EDA, one for evaluation/backtesting, and one for output generation. Plus an orchestrator script calling/executing all of the modules, this script makes sure the names of the artefacts get a prefix identify the run/experiment/backtestperiod.

I try to keep functions small and focused so I can swap stuff in and out without breaking everything. Inputs/outputs always have the same schema and naming, which helps a ton when experimenting.

My last project was a survival model predicting when something will happen (time-to-event).

That said... my tracking could be better. I still lose track of filtering logic and feature engineering decisions, those are way harder to version than model runs. MLFlow/W&B don't help much there. Still looking for a clean, low-friction way to keep that part organized.

4 Reply Award Share ···

arma1997 OP • 2mo ago

I still lose track of filtering logic and feature engineering decisions, those are way harder to version than model runs. Yes im interested to know how people handle this as well. What is your process for this ?

2 Reply Award Share 258 ···

A_random_otter • 2mo ago • Edited 2mo ago

Branches with good names, although I am often too lazy to create a new one. And having too many branches can get get pretty overwhelming quickly. This is why I usually only have them locally and do not push them to the remote unless it is a major refactoring. At some point you'll have to simply delete and/or merge them to keep things tidy and choose a single way forward. I then make a note about this decision in my project notes.

https://www.reddit.com/r/MLQuestions/comments/1odn6qp/data_scientists_ml_engineers_how_do_you_keep/

Links & Artifacts

Landing Page (demo + explanation)

https://track-it-land.vercel.app/

GitHub Repo (technical deep dive)

https://github.com/arjunm97/trackIT-Package

Portfolio Website

https://www.arjunportfolio.xyz/

Full Case Study

https://portfolio-assets-arch.s3.eu-west-2.amazonaws.com/trackIt/trackIt+longformat.pdf