arize

BUILD VERSUS BUY

Tradeoffs for Observability In a World Dominated By Generative AI

Contents

Foreword

By Gabe Barcelos,
Founding Engineer at Arize AI

The instinct to build is intrinsic to most engineers. You see it in our love of open source. You see it at hackathons, where developers pull all-nighters building novel large language model (LLM) applications. You feel it in the air in San Francisco's "Cerebral Valley" and "The Arena" or in dozens of technical hotspots near universities all over the world. There's something about the independence and pride that comes from doing something yourself in a community of other hackers. It's something that I understand deeply after spending years working as an engineer at Adobe, Saildrone, and TubeMogul.

Propelled by this build culture, a revolution in generative AI is underway. According to a recent survey, over two-thirds (66.9%) of developers and machine learning teams are planning production deployments of LLM apps in the next 12 months or "as fast as possible" – and 14.1% are already in production!

This new era calls for new approaches. Given constant change, teams need to be agile – reinventing everything on the fly as new research, foundation models, orchestration frameworks, and methods upend established techniques.

In this world, the tradeoffs about what to spend time building are more stark. A team that spends a month building a piece of infrastructure that only connects to one foundation model (i.e. GPT-4) or orchestration framework (i.e. LangChain) may quickly find their work – or even entire business strategy – rendered obsolete.

At Arize, we work with dozens of enterprises and companies that have both traditional ML models and LLM apps live in production. Some opt to build their own MLOps or LLMOps infrastructure (particularly around orchestration) and lean on open source solutions, while others are tapping an array of third-party tools like major vector databases and Arize for observability.

This paper outlines some approaches, considerations, and perspectives on the build-versus-buy debate around ML and LLM observability.

Overall Considerations

For many teams, considerations about whether to build or buy any piece of technical software are more pragmatic than technical. While not exhaustive, typical tradeoffs appear below and may serve as a good starting point in planning.

Build

- Total control over product direction
- Prioritization of capabilities/ development
- Internal approval and procurement often easier (caveat: once headcount and resources are secured)
- Built for existing stack
- Lower external fees, just infrastructure costs and the opportunity cost of labor
- Can become a proprietary competitive advantage if in the company's domain of expertise

Buy

- Everything in a best of breed platform
- Evolves with new integrations (i.e. a new foundation model) and updates without additional cost
- Support - extended team, more resources
- Scale - infrastructure is built to support thousands of customers instead of a single deployment in a customer
- Speed - buy/install happens in minutes rather than a slow, years-long buildout
- Better Performance - freed-up time for team translates to higher-performing models and LLM apps
- Thorough product documentation and support

Often, these decisions come down to what is better for the organization. ROI is determined by quantifying the impact of model accuracy, model improvements, speed and flexibility in terms of lift to key business metrics. Quantifying AI project ROI will enable easy calculations of the value of observability and individual insights for each project and model.

In practice, all but a few enterprise teams at large technology giants end up in the "buy" category for at least part of their MLOps and LLMOps infrastructure. This is due not to a dearth of technical talent, but rather the effort involved and opportunity costs of spending time on infrastructure rather than on the models and applications that will boost AI ROI in the near term.

Essential Elements For a Modern AI Observability Platform

In determining whether to build or buy an LLM or ML observability platform, it is important to identify essential capabilities. While these will be unique to every team, the following lists – informed by experience working with hundreds of practitioners across dozens of large enterprises and technology companies with everything from computer vision models to LLM apps in production – may prove helpful in whittling down a list of key building blocks to aid in estimating the man-hours and timetables required to build them and associated opportunity costs.

LLM Observability	**Needed?**
LLM System Evaluations	
Modifiable and Runs Across Environments Seamlessly	
Fast Throughput	
Support for Common Pre-Tested Templates	
Metrics Beyond Average Accuracy (i.e. Precision, Recall)	
Ability To Run Evals On the Span and Chain Level	
Reproducible	
LLM Traces and Spans	
Specification for Capturing and Storing All Relevant LLM Application Executions	
Support for LLM Traces	
Comprehensive Ability To Log Span Kinds and Attributes	

LLM Observability (cont'd)	Needed?
Prompt Engineering	
Run On Any Major LLM Provider	
Support for Prompt Variables	
Performance Analysis Across Templates	
Compare Across Prompts and Templates	
Retrieval Augmented Generation	
Support for RAG Metrics	
Embeddings and UMAP Cluster Analysis	
Purpose-Built Workflows for Optimizing and Improving	
Fine-Tuning	
Workflows for Finding Priority Areas for Fine-Tuning	
Data Export Controls	
Embeddings Analysis	
Automatically Cluster Data for Anomaly Detection	
2-D and 3-D Embeddings Projector	
Sort Problematic Clusters by Performance Metrics	
Interactively View Data, Select Data, Colorize Data and Export	
General Platform Support	
Full-Suite Support for ML Observability Beyond LLMs	
Self-Servability	
Customer Support with ML and Data Science Expertise	

Additional capabilities may prove useful to other model types and managing deployed AI generally.

Other Model Monitoring & ML Observability Capabilities	Needed?
Drift Monitoring & Troubleshooting	
Overall Production Drift Detection	
Drift Tracing	
Automatic Thresholding on Drift	
Drift Metric Support & In-flight Application: PSI/JS Divergence/KS Metric/ Embedding Drift	
Automatic Binning Support	
Programmatic Support for Monitor Setup & Configuration	
Troubleshooting Model Drift by Drilling Into Feature Drift	
Compare Training Versus Production Distributions	
Drift on Any Flexible Dataset	
Drift Detection Across Any Cohort	
Configure Baseline Setup	
Performance Monitoring and Troubleshooting	
Monitor ground truth by combining predictions with delayed response label data	
Performance Tracing	
Performance by Sup-Model	
Ranking and Recommendation Model Support	
Custom Performance Metrics	
Production A|B Comparison of Models	
Configurable Baselines That Support Both Pre-Production and Production	

Other Model Monitoring & ML Observability Capabilities (cont'd)	Needed?
Ability To Compare Model Performance Metrics (such as ROC-AUC, PR-AUC, accuracy, precision, recall, r-squared, MSE, MAE) From Trained Model To Production Model (or two other periods of time)	
Monitor Production Models Using Constant Thresholds and Dynamic Thresholds	
Automatically Surface Performance Problems By Feature, Value or Cohort Without a User Needing To Write SQL Queries	
Ability To Perform Dynamic Cohort Analysis and Segmentation Of Predictions	
Dashboards That Non-Technical Stakeholders Can Understand	
Explainability	
Ability To View the Feature Importance For the Top N Features	
Visibility Into Cohort, Model and Local Explainability	
Model Lineage, Validation & Comparison	
Model Versioning and Lineage Support	
Pre-Launch Model Validation	
Data Quality Monitoring & Troubleshooting	
Monitor Production Model For Bad Inputs	
Configurable Real-Time Statistics On Features & Predictions (Min, Max, Median, Mean, Standard Deviation) In Aggregate and By Cohorts	
Outlier Detection On Predictions	
Configurable Baseline Setup	

Other Model Monitoring & ML Observability Capabilities (cont'd)	Needed?
Integration Functionality and Experience	
Agnostic of Model Types/Libraries	
Support SaaS, On-Prem and Hybrid Deployments	
Ability To Set Up Alerts That Integrate with PagerDuty Or Preferred Incident Response Platform	
Automatically Infers the Model Type and Calculates the Appropriate Metrics For Monitoring	
Ability To Easily Import Data From and Export to External Data Sources	
Ability To Handle Analytic Workloads	

Quantifying ROI

To help compare to a potential in-house solution, this section provides an overview of the value teams are seeing with the Arize platform. Based on an analysis of 50 teams with at least one model in production (many of whom have multiple models), the study spans 500+ models with varying use cases across companies of various sizes.

- **Model insight prevalence:** 95% of teams can find a valuable insight when first exploring their data in Arize.
- **Time to first insight:** Users uncover an initial insight within the first 24 hours of exploring their model data in Arize.
- **Confirmed model insights:** We see teams detecting confirmed model issues with Arize monitors once a month.
- **Time to detect:** Detection is immediate with Arize monitoring.
- **Time to root cause:** Users can root cause a monitor alarm within the first 24 hours through exploring their model data in Arize. Without ML observability, this can take days to weeks.
- **Time to resolve:** Resolution depends on the remedial step required. Often, automated model retraining will resolve many model issues. Some issues will require additional data collection or labeling, and model experimentation to resolve.

Often, model insights and improvements can and should be correlated back to business metrics to show ROI and cost savings and observability initiatives. This breakdown shows the value of ML observability per model, based on the estimated cost of productivity and business value of catching model problems in production.

180k* *(lost time)*	$100k*	$280k*
Cost of Team Productivity	**Cost of Issues in Production**	**Total Value of ML Observability**
3 ML Team Practitioners time 30% time spent on MLOps	3 days to resolution without Observability $50k per major model problem in production	Year 1 Alone

*These assumptions are based on representative users and industry-standard salaries

Calculating the Tradeoffs

Estimates around time, and other factors will vary by industry, team, and use case. Hopefully this overview provides a framework through which to make these decisions.

Here is one approach for calculating these tradeoffs at a high level that may be useful to customize:

			Year 1	
	Development	**Maintenance**	**Build**	**Buy**
Compensation Costs (development & administration)				
Infrastructure for Development and Maintenance				
Software Licensing (development, collaboration, etc)				
App Monitoring				
Staff and User training				
ML/LLM Observability Platform Licensing Fees				
Infrastructure Costs for MLOps (Depends on SAAS vs On-Prem)				
Total				

Perspectives

Of course, estimates only go so far – real-world experience is also important. Here are some perspectives from around the industry on how different ML teams approach build versus buy decisions around AI observability.

Mihail Douhaniaris
Data Scientist

Steven Mi
Senior MLOps Engineer
GetYourGuide

"While our in-house platform achieved better standardization of ML model monitoring across projects, it still came with a relatively high cost of maintenance in terms of time and resources. On top of that, developing an observability platform is not our core area of expertise. We quickly realized that an enterprise solution would meet our long-term monitoring needs. As we looked for a better alternative to monitor our ML models, we explored various enterprise platforms available in the market. We met with a few different vendors to get an overview of their products, and after comparing functionality, scalability, and features, our team decided to build a proof-of-concept (POC) using Arize as it seemed to fit all our requirements for an ML observability platform including support for multiple data ingestion, model types, as well as automatic monitor and alert creation."

Kyle Gallatin
Senior Machine Learning Engineer,
Etsy

"At Etsy, we have a wide mix of in-house, open-source, cloud and third-party solutions for different portions of the ML lifecycle. That tends to make third-party tools spanning the entire ML lifecycle a bad fit for us - the more points of integration, the more difficult it is to fit the tool into our existing workflow. However, as complex as ML observability is - it only has a single point of integration. We were already collecting prediction logs containing the features, prediction, and ground truth for our traffic. We could either build out an entire team to handle the frontend, backend and infrastructure of an in-house tool, or upload this data to a third-party service and get all of these things "for free" without any disruption to our existing ML lifecycle. ML observability itself is very hard - but it's not super difficult to slap existing ML observability tooling onto a system that's already operating and collecting data as it should. This made ML observability an ideal buy decision for our use case, and we sought out vendors that fit our wide range of requirements...We settled on working with a third-party vendor to handle our observability as a scalable SaaS solution."

Habib Baluwala, PhD.
Domain Chapter Lead - Commercial Data,
Spark NZ

"Initially we did consider an internal solution. We focused on trying to understand what kind of time investment and team expertise would be required, while also considering how quickly the field might change. We asked ourselves if this would be a one-year setup with a team building a product or if we needed a team on a more continuous basis to develop the product to meet the changing needs. In looking at all those different components, it wasn't worth the cost for us to hire a data scientist, an ML engineer, and front-end developers to build a customizable solution in-house. It's not part of our core business and does not make sense when there are already good solutions available."

Yunshi Zhao
Machine Learning Engineer,
Liftoff Mobile

"Most of the product we're using right now is built in-house because the company wanted to move fast. A lot of our systems are really lean and were built for a specific use case. For example, we have an experiment tracking tool that you can go on and see some of the matches of each performance. It's really simple and can't really do a lot of fancy things that experiment tracking tools in the market right now can do, but it does the job. Right now we do have a push to try and move towards a more standardized tooling because expansion can be a bit of a pain point. Before, I think our ML was focused more on the conversion models, but now we have so many other ML applications. For example, pacing the budget and market price. But then every time we try to build a new model, because of the narrow cases, it's a bit hard. It's also really hard to onboard people to use an in-house product or narrow cases. So because of that, we're also investigating the other tools that might be more flexible and will apply to other ML applications in our company."

Conclusion

As the generative AI field continues to evolve, teams face a critical task in deciding whether to construct or procure their ML and LLM observability infrastructure. While building in-house offers control and potential competitive advantage, for many teams the speed, scalability, and evolving integrations offered by buying third-party solutions may prove more practical and immediate in terms of ROI. Regardless, clearly weighing practicality against technical aspirations in the dynamic landscape of AI development with regard to LLM observability is a critical task.