THE DEFINITIVE

Machine Learning Observability Checklist

What to Look for When Assessing ML Monitoring & Observability Platforms

New Edition

arize

Table of contents

What to look for when assessing ML monitoring observability platforms

Introduction

Effective model performance management requires more than “red light, green light” monitoring of key metrics. Machine learning (ML) teams across all industries and companies invest heavily in building, testing, and experimenting with ML models to carry out mission and operations-critical tasks for their business and customers — yet according to a **recent survey**, over 84% of data scientists and ML engineers say the time it takes to detect and diagnose problems with a model remains a pain point for their teams.

Machine learning observability is a **critical element for any modern ML infrastructure stack**. The right model observability platform enables teams to not only detect when an issue emerges, but provides capabilities that allow for deeper and more proactive introspection to diagnose the root cause of problems before they significantly impact a business or its customers. Armed with the right solution, ML teams can quickly visualize where and why problems are emerging — clicking into slices directly, without added burdens like writing queries.”

This checklist covers the essential elements to consider when evaluating an ML observability platform.

SECTION 01

Drift Monitoring & Troubleshooting

Models are not static. They are highly dependent on the data they are trained on. Drift is a change in distribution over time and can be measured for model inputs, outputs, and actuals. Drift can occur because your models have grown stale, bad data is flowing into your model, or even because of adversarial inputs. Tracking drift in your models amounts to keeping tabs on what had changed between your reference distribution, like when you were training the model or from a prior time period, and your current production distribution.

Overall production drift detection (concept, data, model)

Production drift is a great proxy metric when performance metrics can't be gleaned in real-time (or near-real-time) due to delayed actuals. Ideally, your solution allows for varying levels of drift detection, including:

- **Concept drift:** drift of the actuals. This covers drift of the ground truth from the past — such as from a prior time window, or a training dataset — to now.
- **Data drift:** drift of the features or inputs, or covariate shift.
- **Model drift:** drift of the predictions. This covers differences between what the model is predicting today versus the past. Also known as prior probability shift or prediction drift.

Drift Tracing

The hardest part of tracking down model drift is the ability to trace what features are causing the movement (drift) in the output of the model. Drift tracing allows teams to immediately trace the cause of prediction model drift to the feature causing that drift. Drift tracing combines both model drift analysis, feature drift, and feature importance scores into an impact score by feature, allowing teams to quickly sort and find the root cause of output prediction drift.

Automatic Thresholding on Drift

ML model monitors have a real challenge of being useful and minimizing noise since they need to be deployed on hundreds or thousands of features multiplied by every model deployed. The ability to automatically create and automatically update thresholds based on data changes is imperative to keeping monitors useful and noise minimized. Arize applies a controllable continuous auto-thresholding that enables stable, quiet and useful monitoring for ML drift metrics.

Drift Metric Support & In-flight Application: PSI/JS Divergence/KS Metric/Embedding Drift

Many teams want the ability to set metrics on the fly, in product, without reconfiguring or re-uploading data. Teams need full support for in flight changes and support for any of the common drift metrics without having to update or rewrite ingestion code.

Automatic Binning Support

Drift metrics are driven by the binning approach used by a platform's data engine. The ability to automatically set binning approaches for drift analysis based on the type of data allows teams to automatically monitor features with no manual changes to binning required to monitor. This is critical to have monitors that actually give teams a red light when an issue actually occurs.

Programmatic Support for Monitor Setup & Configuration

Teams increasingly want the ability to setup monitors as code and use the ability to configure their ML monitoring pipelines with APIs and code.

Troubleshooting Model Drift by Drilling Into Feature Drift

Change in the inputs to a model is inevitable when dealing with an online production environment. While models can be resilient to minor changes in input distribution, when distributions stray further from what the model encountered in training the performance on the task at hand eventually suffers. However, drift isn't a binary indication of model health; just because a feature drifts doesn't mean model performance will regress. Therefore, it's important to have an ML observability solution that not only readily surfaces feature changes over time but can surface related metrics such as data quality metrics and performance metrics in the same view so you can better determine the potential impact of a drift.

Compare training versus production distributions

Taking a model from research to production is hard, in part because how a model responds to inputs in a contained training environment won't necessarily translate perfectly to the production environment where the data feeding your model tends to be more dynamic. Understanding when the overall distribution of your model's outputs is shifting from what you expected when training is an important signal for tracking the efficacy of your models.

Drift between training and production may indicate that retraining or updating the model is needed in order to address challenges such as concept drift or model sensitivity.

Drift on any flexible dataset

Your ML observability solution should enable drift monitoring against any baseline that you want to benchmark against. This includes training, validation, different versions of datasets, and earlier windows from production.

Drift monitoring against different versions of datasets is helpful in understanding how your model responds when it encounters new or deviant inputs. And tracking against earlier production windows is especially relevant in time series or forecasting cases.

Drift detection across any cohort

There are situations where a specific cohort is the primary focus of your model monitoring or troubleshooting, such as when expanding your forecasting models to new locations or inventory types.

In those instances, the ability to detect concept, data and model drift at the cohort level provides more tangible value to your analysis than looking at aggregate predictions.

Configure baseline setup

Again, teams need flexible ways to set up a baseline as a reference point for monitoring. An ML observability solution should be able to support baselines coming from training, validation and production.

SECTION 02

Performance Monitoring & Troubleshooting

Once an ML model is deployed into production, consistently monitoring performance metrics is a core element of maintaining overall model health. While monitoring is traditionally a reactive approach to model performance management, the best tools can help you proactively catch regressions before they impact your business or customers and even recognize when it's time to retire or retrain.

Monitor ground truth by combining predictions with delayed response label data

In an ideal scenario, ground truth or actuals are surfaced in real-time (or near-real-time) for every prediction, allowing you to immediately analyze how your model is performing in production. While some industries are lucky enough to encounter this scenario — with food delivery estimates, for example, you'll know the accuracy of an ETA prediction once the meal is delivered to the customer — the latency period for other use cases may be much longer.

Your model monitoring solution should be able to support joining predictions with delayed actuals, regardless of the latency duration, so you can reference and analyze performance for any given time period.

Performance Tracing

One of the most time consuming jobs in data science is tracking down performance issues in production. The ability to go from a drop in aggregate performance to looking at the actual data inputs causing the problem is one of the most important product features required from an ML observability platform. Arize's performance tracing product, for example, is powered by patented technology that enables the platform to analyze performance across millions of cohorts of data, sort up the data causing the biggest problems and highlight what needs to be fixed in seconds. The performance tracing technology also allows teams to quickly compare production to training to surface instantly what data values are new or performing poorly and what to send back to development teams.

Performance by Sub-Model

Many models are designed to have a single model architecture with a sub-model build approach, such as having a model built by city or by product line. In order to support this, you need flexibility to ingest easily, possibly with slight difference in actual input features, then in product have the ability to break out performance by sub-model (say by city).

Ranking and Recommendation Model Support (NDCG, Recall @ K, Precision @ K, MAP @ K, and custom)

The ability to support ranking models requires a complex set of metrics and data formats. ML teams want to dynamically set K for analysis, apply different ranking metrics based on the model view and change as needed without reformatting data. Additionally, calculating ranking metrics at scale is extremely hard, Arize is designed for scaled ranking analysis.

Custom Performance Metrics

Many teams need the ability to write custom performance metrics on their models. Arize supports custom metrics defined on scaled data ingested into the platform. The custom performance metrics are usable throughout the performance sections of the platform such as performance tracing, dashboards, and monitors.

Production A|B comparison of models

Even if you employ an experimentation platform during the model building and testing phases, the ability to see how different versions perform in the real world is the ultimate evaluation of the efficacy of your optimizations and retraining efforts. This is especially advantageous with shadow or canary deploys, where you are gauging performance of a model version on a subset of production data before full deployment.

Your ML observability solution should enable easy, intuitive views (read: purpose-built, side-by-side analysis in the same window) of how each version is performing. And because new model versions often emerge to service a new need or improvement — such as expanding a delivery service to new neighborhoods — the ability to compare performance against specific cohorts of predictions and features will help yield more fruitful analysis.

Configurable baselines that support both production and pre-production

At the most basic level of monitoring, you need the ability to select parts of your production data as the reference point to analyze performance to answer questions like: “Is my model’s accuracy as good this month as last month, or has it regressed?” Look for the flexibility to use *fixed* (i.e. previous month) or *moving* (i.e. trailing 30 days) time ranges.

The ability to set validation or training datasets as the reference point for monitoring also helps you troubleshoot regressions or issues that emerge in production. For example, if you’re seeing more predictions of positive class “fraud” in production than when training the model, you would want to detect this quickly to diagnose the root cause — whether it’s an uptick in actual fraud or an issue with model sensitivity.

- [ ] **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)**

  This helps with more granular troubleshooting when an issue is detected.

- [ ] **Monitor production models using constant thresholds and dynamic thresholds**

  In some instances, you may want to simply detect whether a performance metric exceeds or dips below a constant value — e.g. accuracy below 90%. Other times, it's more relevant to monitor performance from a rolling baseline, such as if accuracy is +/- 2 standard deviations from the previous 72 hour window.

- [ ] **Automatically surface up performance problems by feature, value or cohort without a user needing to write SQL queries**

  A good observability solution should help surface up the unknown-unknowns, identifying the exact inputs or slices that are causing the performance to drop. Ultimately, you want to find problems quickly and directly — not to spend time digging into SQL to find the issues.

  Your ML observability platform should also help you create default monitors in bulk without needing to manually configure each one with the right performance metrics and thresholds based. The monitors should also offer tracking at the prediction, feature, and cohort (subset of dimensions) levels to provide depth of coverage against model health.

- [ ] **Ability to perform dynamic cohort analysis/ segmentation of predictions**

  Look for interactive functionality to select the cohorts you want to analyze on the fly by grouping features, predictions or actuals into facets or slices for analysis. All performance or data for product metrics should roll up based on the slices you want to analyze without requiring pre-established segments.

- [ ] **Dashboards that non-technical stakeholders can understand**

  Once ML models are being relied on and tracked by the business to measure things like product health or customer outcomes, an ML observability solution should help you provide easily-interpretable, non-technical health metrics that any stakeholder can check to determine if models have changed in a meaningful manner.

SECTION 03

Embeddings (Unstructured LLMs/CV/NLP)

According to multiple estimates, 80% of data generated is unstructured audio, images, text, or video. As computer vision (CV) and natural language processing (NLP) models proliferate, ML teams need to stay a step ahead of new patterns in this data causing model performance degradation in production.

- [ ]

Embedding Drift Monitoring for Generative, NLP, CV

The challenge with measuring data drift in deep learning models (CV, NLP, LLMs, Unstructured) is that you need to understand the change in relationships inside the unstructured data itself. These models are extremely hard to troubleshoot and understand how to improve performance. One paradigm shift is to leverage embeddings to analyze deep learning models. Embeddings are everywhere in modern deep learning and by detecting movement in the embeddings. Arize's patent pending approach, for example, is to track drift between embeddings to highlight issues quickly and surface the root cause of the problem in the data.

- [ ]

Automatically Cluster Data for Anomaly Detection

Troubleshooting embeddings is incredibly hard. Teams require the ability to find groups of problems as clusters, and analyze those clusters relative to performance metrics or drift. The ability to automatically find clusters of problems and integrate workflows for fixing the data is required by any team using complex models.

- [ ]

2-D and 3-D Embeddings Projector

Visualizing your data in a low dimensional space is critical to the process of finding the root cause of problems and analyzing them. Projecting embeddings to 2-D or 3-D space is imperative to understand the groups of data that might be causing issues, selecting them, filtering them and exporting them.

- [ ]

Sort Problematic Clusters by Performance Metrics

The ability to sort clusters of problems by performance metrics is critical to understanding what groups of clusters are causing problems and sorting what teams should look at first.

- [ ]

Interactively View Data, Select Data, Colorize Data and Export

The ability to interactively view data, select data, and colorize by performance/features allows teams to work through interactive root cause workflows. Once you find the group of data causing the issue you can export or save the cluster of data for use in monitoring or fine tuning flows.

- [ ]

Auto-Embeddings Generation

In many cases teams do not have access to the internal embeddings within the model itself in production use cases. The platform should have auto-embedding generation for automatically generating embeddings for use within embedding workflows.

- [ ] **UMAP visualization to better troubleshoot CV and NLP models**

  Once drift is identified, you need to click a level deeper to visualize what may be happening with the data in order to troubleshoot. By visualizing embeddings on a UMAP plot with production embeddings overlaid over training embeddings, you can identify new patterns in production not seen in training.

- [ ] **Identify and export what what data to label next**

  What works in training may not work in production when identifying what to label next. By identifying new patterns to focus on for labeling and retraining, teams can save significant overhead.

SECTION 04

LLM Observability

Monitor GPT-4/Anthropic/General LLMs

The latest explosion in LLM growth has caused a large set of GPT-4 use cases to be pushed into production with little visibility in outcomes and no ability to root cause problems in production. An ML Observability solution should support the modern foundational model prompt and response pairs collecting data on use, problems, and interactions.

Performance Evaluation of Prompt and Response Pairs

The ability to understand how each task for each prompt template is performing and where it is not working is critical to supporting production use cases. Supporting evaluation metrics from OpenAI Evals that are integrated into the SDK and supporting analysis methods that break these down by prompt template is important to managing production use cases.

Cluster Prompt and Response Pairs

The ability to cluster groups of prompts and responses that are problematic is one of the main approaches to troubleshooting production issues. Once you find problematic groups, teams can either fix using prompt engineering or fix using fine tuning datasets built from similar samples.

Prompt Engineering Workflows

The ability to iterate on a prompt quickly once you have found a problem in a template is critical. Sometimes the best way to fix a prompt is to ask an LLM for a recommendation. The ability to call back to an LLM for insights, iterate in product on the problem data and suggest a prompt fix is critical to the workflow.

- [ ] **Monitor Token Usage and Delays**

The simple stuff is sometimes most important. The ability to break out usage by tokens and API delays is important to track ad monitor. The ability to add custom metrics on top of the counts is important to teams tracking actual costs. The ability to use tags to segment out tasks and usage patterns is important for tracking.

- [ ] **Fine Tuning Workflows**

The fine tuning approach is key to improving long term model performance, and the key to fine tuning is collecting the right data. As teams discover problem clusters, saving those clusters then applying them to export the right fine tuning datasets is a pivotal part of the workflow needed to improve the core datasets.

SECTION 05

Explainability

Explainability isn't a leading indicator of model performance and therefore shouldn't be the primary functionality you evaluate when selecting an ML observability solution. However, model explainability can be a useful tool when performing deeper root cause analysis after identifying an issue, or exploring ways to improve a model to deliver better outcomes in future iterations. The ability to introspect and understand why a model made a particular prediction is also important for AI ethics and broader organizational concerns.

- [ ] **Ability to view the feature importance for the top n features**

  Embeddings – vector (mathematical) representations of data where linear distances capture structure in the original datasets – are everywhere in modern deep learning, from deep neural networks to transformers. By monitoring embeddings of their unstructured data, teams can proactively identify when their unstructured data is drifting in and improve their CV and NLP models in production.

  Since the metrics typically used for drift in structured data – such as Kullback-Leibler divergence (KL divergence) – only allow for statistical analysis on structured labels and do not extend to unstructured data, an ML observability solution needs to support a metric that is proven to be sensitive and stable like Euclidean distance to quantify embedding drift.

There are several levels of explainability to consider, each with potential to assist in various stages of model training, validation, and production

- **Global explainability:** Model explainability across all predictions, attributing which features contributed the most to the model's decisions. Global explainability is useful to convey how a model makes decisions on average and can help you detect problematic logic before a model ships into production.
- **Cohort explainability:** Model explainability for a particular subset of data, also known as a cohort. Cohort explainability is useful when attempting to convey why a model isn't performing as well for a set of inputs and in discovering model bias. This is particularly useful in validation, when you're testing your model's ability to generalize performance across your dataset.
- **Local explainability:** Model explainability for an individual prediction. This level of explainability conveys why, for a specific example, did the model make that particular decision. Local explainability is indispensable for root causing a particular issue in production — for example, to answer customer support questions.

SECTION 06

Business Impact Analysis

Business metrics go hand in hand with model performance metrics and — when properly defined — they should be linked. At the end of the day, you aren't shipping a F1 score to your customers, so it's important to keep in touch with how your models are affecting how each customer is experiencing your product.

Since these metrics are not easy for a model to optimize for, the optimization problem will be set up using a parallel metric.

Custom user defined function (UDF) to tie model performance back to business metrics

The cost of model mistakes to a business and its consumers generally aren't equal. With credit card transactions, for example, a false positive for fraud typically just means a minor inconvenience to the customer who might need to verify over email/text that a transaction was legitimate — whereas a false negative is an immediate financial loss for the credit card provider.

Look for an ML observability solution that doesn't simply flag when there's a potential performance issue, but enables you to analyze the potential risk to the business. This can be in the form of UDF or formulas that tie true positive, false negative, false positive, true negative thresholds back to your target business metrics.

Dynamically analyze thresholds for probability-based decision models

Once you've defined your business metrics, you'll want the ability to analyze the thresholds in which your probability-based decision models are able to operate before regressions have a meaningful impact on your business goals. If a model can drift 10% before any measurable impact is seen, for example, you can configure your monitors accordingly.

Compare pre-production models to current production models — champion and challenger

This is especially useful when approximating the business risk or incremental performance lift of deploying a new model or model version into production.

SECTION 07

Model Lineage, Validation & Comparison

Model versioning and lineage support

The following can be helpful in comparative analysis:

- Ability to send predictions for different versions of the model.
- Ability to determine which model versions belong to the same model.
- Ability to compare performance of two different model versions and automatically surface which model is better-performing and why.
- Ability to set up customizable multi-model analytics for different model versions and models on the same dashboard.

Pre-Launch model validation

To avoid situations where ML teams are shipping models into production “blind,” it’s highly recommended that any model observability solution you employ provides a number of pre-launch validation capabilities, including:

- Ability to track training datasets and product predictions data across model versions.
- Ability to track multiple validation datasets as “batches.”
- Ability to set up customizable dashboards analyzing pre-production datasets.
- Ability to compare business impact of a new not-launched model versus production-deployed model.

Think of these like CI/CD checks for ML models. The more readily you are able to map changes in model version, datasets used, and expected results, the better able you are to focus your monitoring and troubleshooting efforts.

SECTION 08

Data Quality Monitoring & Troubleshooting

A model is often only as good as the data it is trained on. Data quality doesn't stop being important after the model is trained, but continues to remain important as the model is deployed in production. The quality of the model's predictions is highly dependent on the quality of the data sources powering the model's features.

Monitor production model for bad inputs

The ability to triangulate whether the production issues you are encountering are due to a faulty model or a problem with the data source is fundamental to guiding your troubleshooting and remediation efforts.

Because the relationship between the data feeding a model and its outcomes are inextricably linked, look for ML observability solutions that can help you monitor and troubleshoot both inputs and outputs. Data quality monitoring should include: data schema change, data malformation, new ground truth class, bad/missing values in columns expected to be clean, among others.

Configurable real-time statistics on features & predictions (min, max, median, mean, standard deviation) in aggregate and by cohorts

Think of this as your first line of defense against performance regressions. Minor fluctuations in data will not likely result in drastic model behavior or indicate a point of concern, but seeing feature values that are, for example, two standard deviations from what you'd generally expect is probably a red flag.

Because you are best equipped to understand the parameters in which your model was meant to thrive and its limitations, look for ML observability solutions that offer pre-set as well as fully configurable thresholds for data quality monitoring.

Ability to detect anomalous behavior (outlier detection) on predictions

When the cost of model mistakes are high, as is often the case in industries such as healthcare or insurance, the ability to flag extreme or out-of-place behavior is incredibly important.

A handful of outliers may simply warrant human intervention to review the model decision, whereas a large or increasing number of outliers could indicate a problem with the model or data source itself.

Configurable baseline setup

Flexible ways to set up a baseline are also important in data quality monitoring. Baselines can come from training, validation, and production. An ML observability platform's inference store should have a history of data to reference from training sets or historical production data to set intelligent baselines and thresholds to balance the frequency of alerts and give power back to ML teams to ensure high-performing models in production.

SECTION 9

Integration Functionality

Agnostic of model types/libraries

Your ML observability solution should connect seamlessly into your existing ML stack and model building framework, not the other way around.

Support SaaS, on-prem and hybrid deployments

While SaaS comes with many benefits of dynamic scale and reduced operational overhead, there are simply instances where a project, team, line of business, or entire organization needs to operate in a private installation. Look for vendors that are able to reasonably accommodate these needs now and in the future.

Specializes in model monitoring and observability instead of providing an end-to-end hosting and serving system

Machine learning systems and models are incredibly complex and require substantial investments in resources from a financial and talent perspective. We highly encourage ML teams to look for purpose-built solutions designed with this complexity and domain expertise in mind for a variety of reasons, namely:

- **Depth of use cases supported:** solutions that are built for a particular purpose (i.e. production observability) often provide more comprehensive value for that specific area.
- **Flexibility to customize your own stack:** connect the best solutions across your ML lifecycle rather than settling for what's "included in the box."
- **Maintain autonomy in choice and control:** it's easy to get locked into building your stack around the pricing models of end-to-end hosting and serving platforms. Instead, consider the specific value being extracted from the tools and their limitations — just because something is there doesn't mean it's worth using.

- [ ] **Ability to set up alerts that integrate with PagerDuty or your preferred incident response platform**

  As mentioned, the ease of connecting into your existing ML stack and systems is paramount.

- [ ] **Automatically infers the model type and calculates the appropriate metrics for monitoring**

  When you're dealing with hundreds or thousands of models, efficiency of automation is key to scaling production monitoring and observability. Look for platforms that can automatically detect the schema of models and data being sent through to save yourself set-up time in the future.

- [ ] **Ability to easily import data from and export to external data sources**

  Your observability solution shouldn't impede the flow of data across your systems or workflows. Look for solutions that allow you not only to easily import data, but extract easy-to-interpret insights to share out to stakeholders (in the form of custom dashboards or otherwise) and in rawer formers such as a Jupyter notebook that can be shared back to the data science or model-building team.

SECTION 10

UI / UX Experience

While *function* is paramount for your ML observability solution, *form* shouldn't have to be a tradeoff!

☐

Flexible, customizable dashboards that technical and non-technical stakeholders can check to determine if models have changed

Your analysis is only as useful as your ability to easily convey the findings across your team and business. The ideal ML observability solution should complement your efforts with the ability to present rich, interpretable insights that are useful to any type of stakeholder you work with. Look for:

- Fully customizable dashboards.
- Ease of sharing (links that save filters, PDF export, etc).
- Intuitive data visualizations via heatmaps, charts, and other at-a-glance formats.

☐

Dark mode (optional)

Monitoring and troubleshooting models is hard work, but it doesn't need to be hard on your eyes, too.

SECTION 11

Scalability To Meet Current and Future Analytics Complexity

A platform's architecture matters, particularly when supporting enterprise scale; without it, ML teams might be delayed or lose insights at the worst possible time.

Ability to handle analytic workloads

Knowing where a platform is using online transaction processing (OLTP) versus online analytics processing (OLAP) is important since each is optimized for different workloads. Ideally, the analytic data from which numeric reports are built should be housed in an OLAP, columnar system (i.e. Druid). OLTP or row-based engines will often fail at analytic workloads once the quantity of datapoints in each query becomes large since these systems are not optimized for fast scans of millions to billions of rows and are generally not able to support interactive queries (i.e. subsecond latencies). It's also important to be able to scale out that workload beyond a single server instance — generally, you need something that can expand query semantics to terabytes or petabytes of data.

Ability to support load testing

As the old proverb says: trust, but verify. In addition to asking for a reference customer, a load test can be helpful in proving ability to handle current analytics complexity. A few example parameters:

- Test > 500 features.
- Test > 50M predictions a day with a total of over 500M to 1B predictions in total over a testing period (10+ days or more).