Industry White Paper:

The Complete Guide to Building or Buying an ML Monitoring Solution

Introduction

What this guide is about

Artificial intelligence-powered applications (underpinned by machine learning models) are disrupting markets from ad tech to agriculture. In the past 3 years, countless organizations have launched ML models to production. This progress is driven by tools built by cloud companies and vendors. These tools lower the barrier for an organization to successfully take an ML project from an idea to production. Once the model is in production, the main challenge of deriving business value from it begins. But it's not enough to simply build and productionize those models, ML teams need to proactively address AI performance degradation to ensure return on investment for the models they build.

ML monitoring and observability is the key to ensuring the reliability of your team's models. By proactively monitoring models and quickly debugging issues with the machine learning system as they occur, data scientists and machine learning engineers are able to ensure that your models are performing as expected, thus reaping the immense value of machine learning.

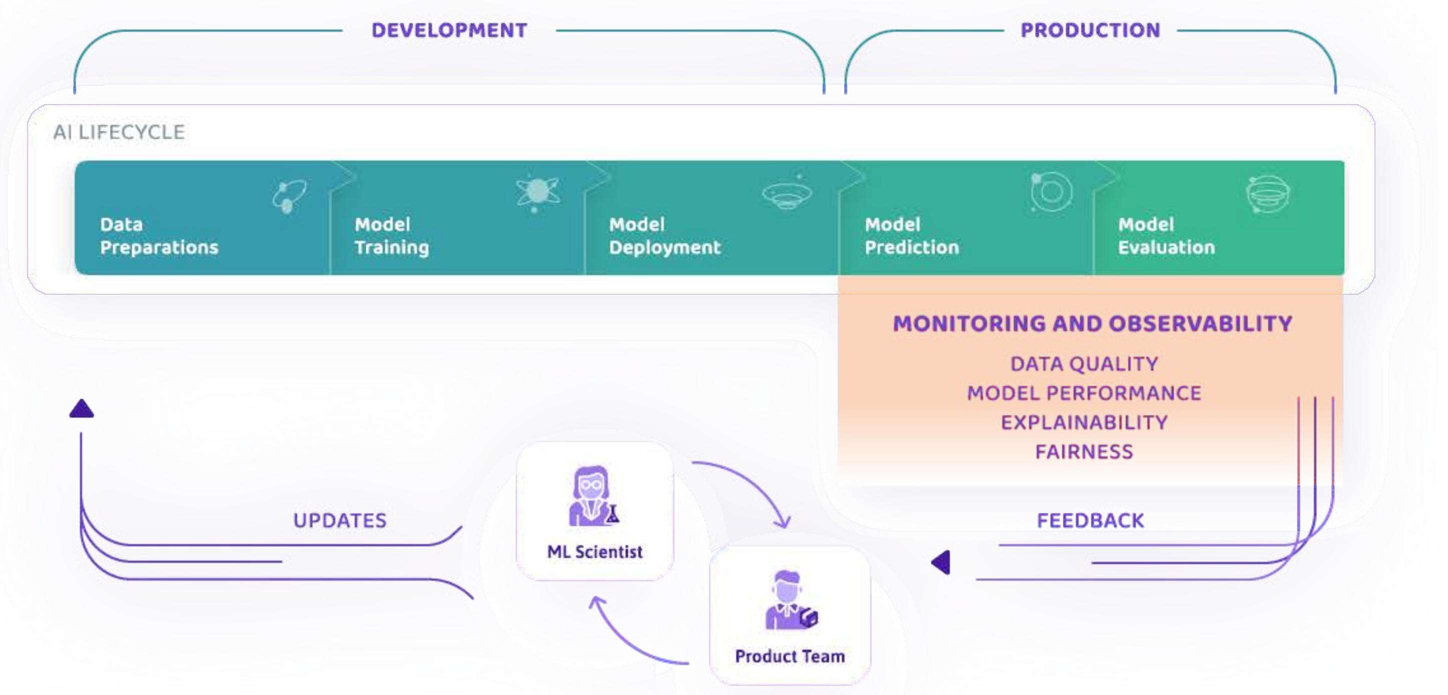
This guide is written as a reference for any team that operates ML applications, focusing on the automation of monitoring capabilities.

The goal of this guide is to thoroughly answer a single question:

> *Should you build your own ML monitoring solution from scratch or buy a solution from an MLOps vendor?*

What happens without monitoring and observability

Once a model is in production, ensuring that it's consistently making highly accurate predictions becomes critical. Monitoring is the proactive, automated process which generates alerts when there are degradations in model performance. Observability is the complement of monitoring, which gives users visibility into how the machine learning system as a whole is performing, allowing them to debug the problems that monitoring surfaces.



Without monitoring and observability, two main issues arise: **AI failure** and **ROI failure**.

AI failure

AI failures occur when models fail to make predictions or when they make predictions that are so bad that they may as well not have made a prediction at all. Without monitoring, model operators only learn about failures after they have already negatively impacted users. Monitoring allows businesses to be proactive rather than reactive when it comes to AI failures.

Without monitoring, model failures can go unnoticed for days, weeks, or even months. This happens because model failures sometimes occur only in a segment of predictions, say in predictions that affect a specific customer type or geography. The result is that until this segment either becomes big enough or affected end users are vocal enough, the model operators will not notice the problem.

What are model failures like? When the model is receiving inputs that contain missing data or out-of-distribution data, it doesn't throw an error, it generates an inaccurate prediction. These inaccurate predictions can be an ad served to the wrong audience, a wrong product being recommended, or a wrong price being predicted for a bid. When predictions degrade in accuracy, the downstream impact carries a financial component - the ads do not convert, the recommendations do not bring engagement, and the high bids affect the margin.

On a small scale, these incorrect predictions affect the return on investment of the model - meaning that the predictions the model is making are not making positive impacts on revenue to justify the model costs. On a large scale? In Q2 2022, Unity released a statement in the quarterly report that an ads model caused $110M in revenue loss due to bad data.[1]

ROI failure

Not only do models fail to live up to expectations if they aren't being monitored, but the costs of maintaining the model in production can also get out of hand. Without observability and automation, machine learning practitioners will be forced to make costly, manual adjustments to the model. This costs the business both time (negatively impacting its ability to quickly capture market opportunities) and money (negatively impacting the business's bottom line).

Once a model is in production, it requires continuous maintenance. This maintenance might involve periodic retraining, fixing bugs in the data pipeline, and making improvements to the ML system as a whole. The more manual this maintenance is, the more expensive the total cost of the model, and the higher the likelihood that the model will fail to achieve the expected ROI. For example, if the model in production requires 2 full time employees to continuously maintain it, this will add >$500K of annual cost to the model, which will offset the ROI.

In Deloitte's "State of AI in the Enterprise"[2], 82% of respondents said that they were seeing a positive ROI from their AI initiatives. The other 18%? Well, they might be companies like Zillow, which famously reported $300M in losses[3] on their house flipping business. One of the causes of ROI failure in this situation were data quality issues.

How monitoring helps

Monitoring ML systems for data quality, data drift, model performance, and other key characteristics prevents both AI and ROI failures. By monitoring their ML systems, model operators are able to proactively identify when model performance will begin to degrade. And by automating the processes around model maintenance, they will be able to ensure that models produce the return on investment that their business expects.

How does ML monitoring fit into your stack?

MLOps stack or ML Platform:

To solve the problem of AI failures and ROI failures, ML teams implement monitoring and observability as part of their MLOps strategy. An MLOps strategy is a set of practices which will enable the organization to deploy and maintain ML/AI applications in production reliably and efficiently.

Organizations may refer to the tools that underpin the MLOps strategy as the ML Platform or MLOps Platform.

Typical components

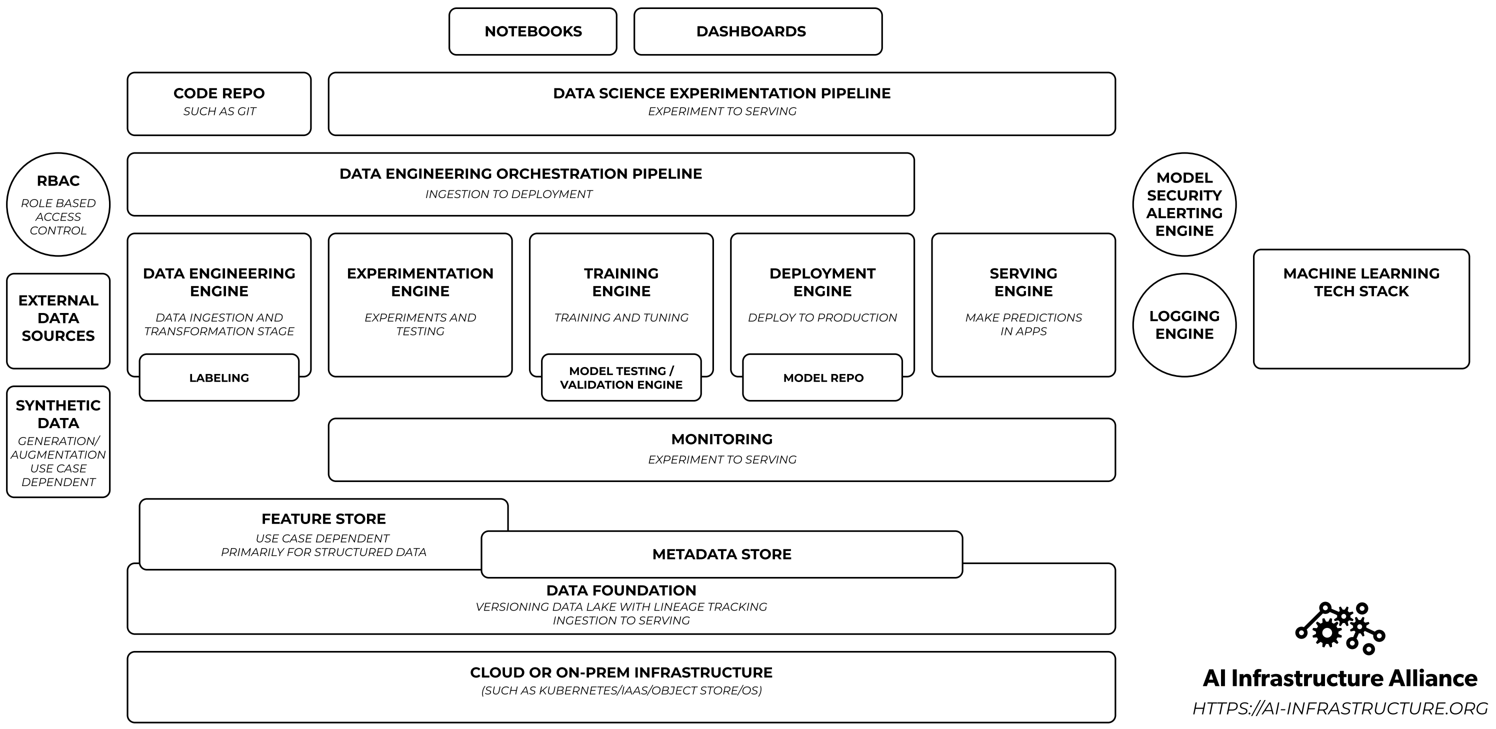
The AI Infrastructure Alliance[4] is an organization that aims to help the AI industry converge on a

1. https://twitter.com/Brian_Stoffel_/status/1524396516420562944
2. https://www2.deloitte.com/content/dam/insights/us/articles/4780_State-of-AI-in-the-enterprise/DI_State-of-AI-in-the-enterprise-2nd-ed.pdf
3. https://www.npr.org/2021/11/03/1051941654/zillow-will-stop-buying-and-renovating-homes-and-cut-25-of-its-workforce
4. https://ai-infrastructure.org/

WHY LABS

common understanding of the building blocks and best practices for doing AI/ML at enterprise scale.

Following the collaboration with architects from dozens of cloud companies and MLOps vendors, AIIA has issued a blueprint of a canonical MLOps Platform stack:


NOTEBOOKS
DASHBOARDS
CODE REPO
SUCH AS GIT
DATA SCIENCE EXPERIMENTATION PIPELINE
EXPERIMENT TO SERVING
RBAC
ROLE BASED ACCESS CONTROL
DATA ENGINEERING ORCHESTRATION PIPELINE
INGESTION TO DEPLOYMENT
MODEL SECURITY ALERTING ENGINE
EXTERNAL DATA SOURCES
DATA ENGINEERING ENGINE
DATA INGESTION AND TRANSFORMATION STAGE
LABELING
EXPERIMENTATION ENGINE
EXPERIMENTS AND TESTING
TRAINING ENGINE
TRAINING AND TUNING
MODEL TESTING / VALIDATION ENGINE
DEPLOYMENT ENGINE
DEPLOY TO PRODUCTION
MODEL REPO
SERVING ENGINE
MAKE PREDICTIONS IN APPS
LOGGING ENGINE
MACHINE LEARNING TECH STACK
SYNTHETIC DATA
GENERATION/ AUGMENTATION USE CASE DEPENDENT
MONITORING
EXPERIMENT TO SERVING
FEATURE STORE
USE CASE DEPENDENT
PRIMARILY FOR STRUCTURED DATA
METADATA STORE
DATA FOUNDATION
VERSIONING DATA LAKE WITH LINEAGE TRACKING
INGESTION TO SERVING
CLOUD OR ON-PREM INFRASTRUCTURE
(SUCH AS KUBERNETES/IAAS/OBJECT STORE/OS)
AI Infrastructure Alliance
HTTPS://AI-INFRASTRUCTURE.ORG

What are the most complex components and why?

To build and operate AI applications in production, the team will need to analyze their requirements for each component in this stack, evaluate what solutions exist on the market–in the open source, as cloud services, or as vendor services–and fit them together to build an MLOps Platform.

Embarking on the journey of establishing an MLOps strategy and an MLOps Platform in an organization can be overwhelming because building each of these components can require significant engineering hours. There is a full spectrum of approaches organizations take to create an MLOps platform, but they can be boiled down to three general approaches:

- Build and host in-house using home-grown and open-source libraries
- Compose an MLOps platform from a mixture of cloud services, open source libraries, and vendor tools
- Procure a monolithic MLOps platform from a vendor

Depending on an organization and its ML use-cases, each of these approaches can be the right way to establish an MLOps Platform.

There are many guides that help AI/ML leaders think through the MLOps Platform holistically, if you are at the beginning of the journey, consider also reading through a few guides we link in the reference section.

This guide focuses on one of the most challenging components of the MLOps Platform. There are a few reasons why it's challenging:

- Monitoring provides the indicator of whether the entire MLOps pipeline is working
- Monitoring provides the indicator of whether the models are delivering the impact they were designed to deliver
- Monitoring touches all stages of the model lifecycle, so it has to be flexible enough to integrate across these stages
- Monitoring has to be the most robust component in the system - if monitoring is unreliable, it's impossible to trust the models operated by the MLOps Platform

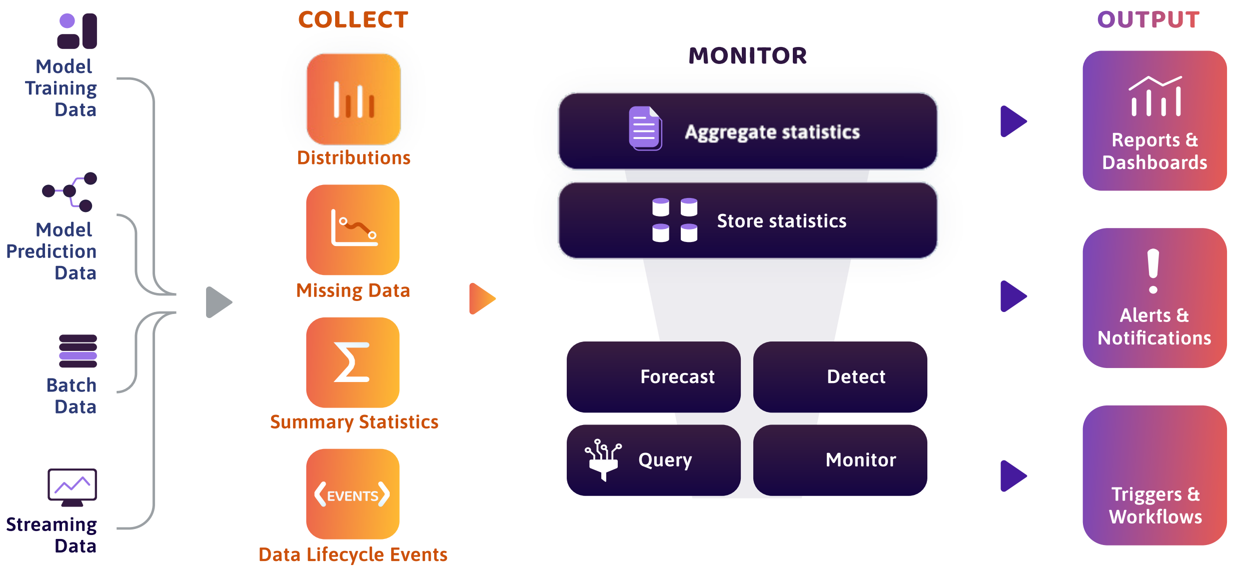
How does ML monitoring work

Monitoring is a critical piece to a well built MLOps system as it enables data scientists, product owners, and executives to trust that critical systems are functioning correctly. To enable monitoring in an ML pipeline, a systematic collection of logs, traces, and metadata must be carried out at each of its stages. In addition to infrastructure health and model performance, logs need to describe statistical properties of the data that runs through the pipeline.

A monitoring solution consists of two core activities:

Collect: The process of collecting vitals from the ML application. These vitals represent model inputs, model output, model performance, ground truth, and other relevant signals about the application health (i.e. latency, load, conversion).

Monitor: The process of automatically identifying anomalies in the vitals and surfacing them to the stakeholders. To accomplish this, the vitals from the previous process are aggregated and analyzed for deviations from the expectations. When a deviation is identified, the alerts notifications are sent and dashboards surface deviations to power debugging.



Architecturally, a monitoring solution has the following components:

1. Telemetry collection
2. Telemetry ingestion and pre-processing
3. Telemetry analysis and anomaly detection
4. Visual result and anomaly display
5. Alert and notification generation

With these minimum components in place, an ML monitoring and observability solution can provide the value of ML monitoring previously discussed: building stakeholder trust in the reliability of ML models in production.

Beyond these table stakes components, monitoring solutions will likely require other specific features. To learn more about the breadth of features that a monitoring solution may require, skip ahead to the Deeper Dive: Features section.

Build or buy

This brings us to the main subject of this guide: how can you decide whether to build or buy the ML monitoring solution that you need in order to reliably deploy models to production? There is no single answer that works best for everyone, so we propose a framework based on the core considerations that you must go through.

This framework is based on answering three fundamental questions:

1. What is needed to solve this problem?
2. How much will it cost?
3. Do you have the resources to solve it?

Each of these questions is important to analyze in order to come to a decision about whether to build or buy an ML monitoring solution. Each of the questions also has multiple components, which is why we propose the following framework to evaluate how to answer each of those three questions:

1. What is needed to solve this problem?
   - **Feature completeness:** Integrations, anomaly detection, alerting, dashboards, interactivity, etc, all of these feature requirements need to be evaluated
   - **Ease of use:** If a solution isn't easy to use, users won't use it, in which case it's as bad as the solution not existing
   - **Security considerations:** This is more often a consideration for deciding between tools that you'd buy, but it's worth considering for building your own solution as well
2. How much will it cost?
   - **Sticker price:** The negotiated price of a good or service in a signed contract
   - **Labor cost:** The cost associated with either building a monitoring system from scratch or integrating and setting up an existing monitoring system
   - **Infrastructure cost:** The cost of hosting a software (be it one you built yourself or one you procured and are running in your environment) in your cloud environment or data center

- **Maintenance cost:** After the system is built and deployed, how much will it cost to keep it running well?
- **Opportunity cost**: In the time that it takes to build or set up the system, how much market opportunity has been lost

3. Do you have the resources to solve it?
    - **Human resources:** If you decide to build, do you have the necessary headcount to develop this solution?
    - **Budgetary discretion:** If you decide to buy, do you have the signing authority to procure this solution?

This comprehensive list of considerations should be used to evaluate options when it comes to building or buying an ML monitoring solution. Below, we elaborate on each of the components of this framework, so that you can more easily evaluate your particular situation, with all of its priorities and constraints.

CONSIDERATIONS FOR BUILDING

What is needed to solve this problem?

- What features will I need to build?
- How easy will it need to be to use?

How much will it cost?

- What is the labor cost of building it from scratch?
- What is the infrastructure cost of hosting it?
- What are the costs of maintaining it?
- What could my team have built if they hadn't been building this?

Do you have the resources to solve it?

- Do you have the freely available engineering talent to build this solution?

CONSIDERATIONS FOR BUYING

What is needed to solve this problem?

- What features will a tool I buy need to have?
- How easy will it be to use?
- Does it meet my security requirements?

How much will it cost?

- What is the contract price of the tool?
- What are the labor costs of integrating the tool?
- Are there infrastructure costs for hosting the tool?
- Are there maintenance costs for using the tool?

Do you have the resources to solve it?

- Do you have the budgetary discretion to procure this tool?

What is needed to solve this problem?

First, it's important to scope the problem that needs to be solved. Some problems can't be solved by buying a tool and require in-house expertise to build a unique solution. This may be due to feature requirements, ease of use, or security concerns. Other times, vendors provide solutions beyond what you could build in house.

By scoping the requirements for your solution, you can begin to understand how much it will cost to build or which vendor to select if you decide to buy.

Feature completeness

In the ML Monitoring Overview section, we defined the key features of an ML monitoring tool:

1. Telemetry collection
2. Telemetry ingestion and pre-processing
3. Telemetry analysis and anomaly detection
4. Visual result and anomaly display
5. Alert and notification generation

While these features are necessary for any ML monitoring solution, they may not be sufficient. For instance, a monitoring solution needs to integrate with your particular training and deployment tools and platforms.

When comparing different vendor solutions against each other and against a solution that you could build yourself, the first thing to evaluate is whether the solution has the features needed to solve your problems. There are many ways to build an ML monitoring product, and vendors make tradeoffs about which features and use cases to prioritize, so make sure that the solution you pick is the right one for you.

For instance, if you are running computer vision models, being able to monitor image data will be table stakes. Similarly, if your team is deploying your models using a managed deployment service like Amazon SageMaker or an open source solution like BentoML, you will need to make sure that the solution you pick or build integrates with those tools.

To better understand the features that may be relevant for your ML monitoring solution, take a look at the Deeper Dive: Features section.

Ease of use

But even if a perfect system is built, there's no guarantee that it will get used. If MLOps engineers aren't careful to design the monitoring solution such that users can easily integrate them with their existing and future models, building the monitoring solution is simply a waste of time. After all, a monitoring system that doesn't get used is as good as a monitoring system that wasn't built at all.

When making the choice about whether to build or buy, be sure to take into account not just the effort it would take to build a functional tool but a usable tool. Similarly, examine the solutions offered by vendors to understand whether their tools will be easy for your data scientists and machine learning engineers to use.

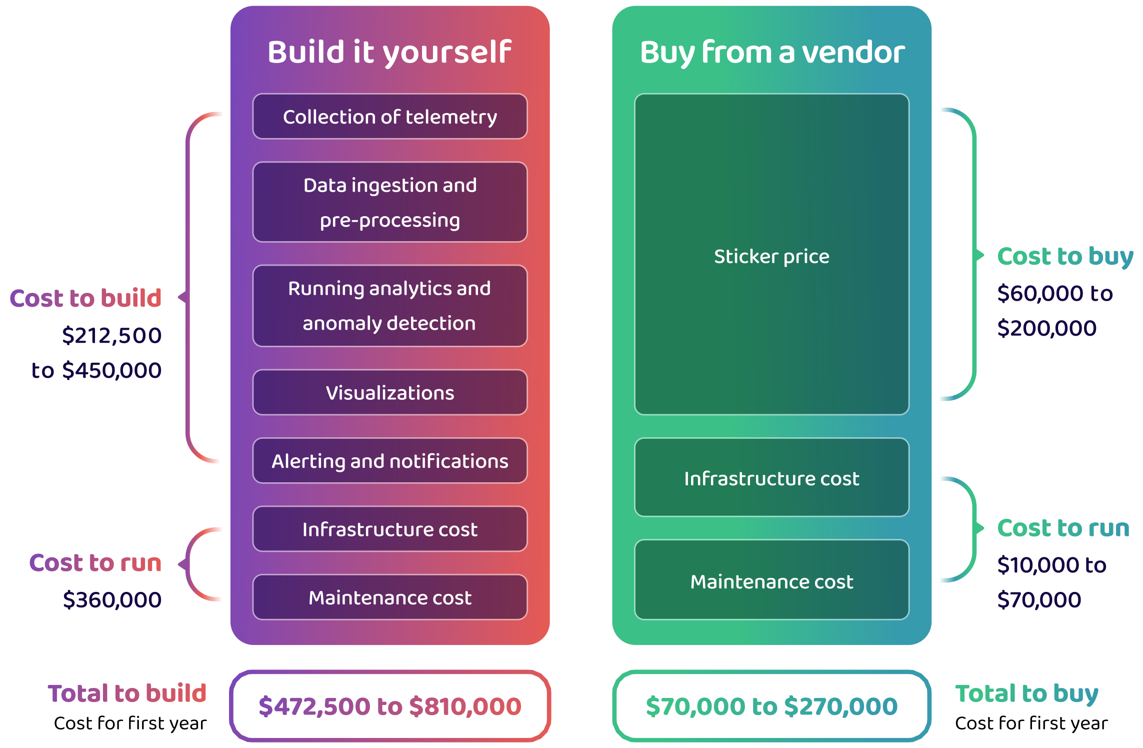
Security considerations

Finally, it's important to consider whether it's feasible to use a vendor solution, given the security considerations of using external software. Many monitoring solutions require access to your raw data, which can be a non-starter for security and compliance teams. Having proprietary vendor software interact with raw data opens a data science team up to potential liability. Therefore, part of your evaluation of building or buying an ML monitoring tool should be whether your security and compliance teams will allow you to use an outside vendor at all.

How much will it cost?

Evaluating the question of cost is generally more straightforward for vendor solutions than for building a monitoring solution yourself. This is because most of the cost is clearly articulated in the sticker price of the solution. Still, there are other costs which need to be evaluated beyond the sticker price, such as labor infrastructure costs, and maintenance costs.

Let's evaluate them one by one.



Sticker price

The sticker price of a model monitoring solution is the price that a company pays to monitor their models using a solution. Although there are many pricing models and structures for vendor solutions, especially in the enterprise space, they mostly fall into two buckets:

1. Model-based pricing: Model-based pricing is the most transparent pricing model for ML monitoring solutions. Companies pay a fixed amount for each model that they monitor. Some

variations to model-based pricing might charge more for higher-granularity monitoring (e.g. hourly monitoring may cost more than daily) or segmented monitoring, but the fundamental SKU in these pricing models is still an ML model.

2. Volume-based pricing: Volume-based pricing is less transparent and more variable, based on the number of predictions being made by the model. Since the number of inferences may vary from model to model or even day to day for a model, this pricing model may end up costing customers more than they expect. Companies may also limit the number of features that a model may have in the volume-based pricing model.

Many companies offer "custom pricing" options, especially for enterprise customers. However, since these pricing models are generally based on the Cost of Goods Sold for the vendor, this custom pricing is generally in some way based on one of these two pricing structures.

Naturally, there is no sticker price for solutions that your team builds.

To better understand the types of pricing offered by vendors, take a look at the Deeper Dive: Pricing section.

Labor cost

Building

However, the sticker price advantage of building is often offset by the labor cost disadvantage. Even when relying on open source tools as the base for building an internal system (and currently there is no complete solution for ML monitoring purely in the open source), it requires a significant amount of engineer labor to build a system that satisfies even the most basic requirements scoped in the "Will it solve the problem?" section.

Going from zero to one in the house-built monitoring system is fairly straightforward, the simple prototype can be built in a few sprints by one or two engineers. However, going from this prototype to a solution that will reliably monitor production models and produce actionable alerts is a big endeavor. The time to build a monitoring solution is also affected by the scale of the data that flows through the model inference or data pipelines.

Here's a breakdown of the estimated number of developer months that it will take to build a bare-bones ML monitoring solution, with no consideration for special features that your use case may require:

Activity	**Under 100 GB/day**	**Over 100 GB/day**
Collection of data/telemetry	0.5	2
Data ingestion and pre-processing	1	2
Running analytics/anomaly detection	3	6
Visualizations of the data and anomalies	3	6
Alerting/Notifications	1	2
Total developer months	8.5	18

Buying

Buying a tool has its own associated labor costs, but they are a fraction of the labor costs involved in building a solution from scratch. The technical labor costs associated with buying a tool are primarily the work it takes to install it (in the case of a managed service or enterprise software solution) or sign up for it (in the case of a software-as-a-service solution).

In addition to the technical labor cost, there is also a non-technical labor cost for procuring an external solution. The non-technical labor cost of choosing to buy a monitoring solution is the cost associated with procurement activities by non-engineering teams such as:

- Procurement team
- Legal team
- Information security team

These teams are often involved in the purchase of any software tool, especially tools that deal with potentially sensitive data. Involving these teams accrues a certain cost which must be included in the calculations.

Infrastructure cost

In addition to the sticker price and labor cost, after a tool is built or bought, set up, and configured, there becomes a recurring cost involved in running the tool. Except for in the case of a software-as-a-service solution, this infrastructure cost is borne by the MLOps platform. Depending on how the system is architected and how much data is being run through it, this infrastructure cost can be quite large for a built system.

Similarly, SaaS ML monitoring services have little associated infrastructure cost, while non-SaaS services will require hosting the entire monitoring solution on your own infrastructure.

Since SaaS solutions only require hosting the telemetry collection part of the solution in the customer's environment, the only infrastructure cost is the overhead associated with capturing that telemetry.

Maintenance cost

Monitoring solutions must be reliable, so maintenance is not to be taken lightly. And, for many teams, long term maintenance of house-built tools is the most often overlooked consideration. The simplest rule of thumb to calculate maintenance needs: for every 1 hour spent building the software, 10 hours will be required to maintain it.

Building

With monitoring, one aspect of ongoing maintenance is configuration. ML and data systems are very dynamic: features change, distributions change, models get retrained so baselines change. One of the most common complaints from teams about in-house monitoring systems is how time consuming and error-prone the configuration/re-configuration maintenance is.

Buying

There are also maintenance costs associated with a procured solution, but they are relevant when an upgrade is required because either:

- Your infrastructure (or other components of your MLOps platform) change or
- The solution gets updated

These costs are generally lower than the costs associated with maintaining software that you've developed yourself, but maybe significant, especially for PaaS or on-prem solutions.

Opportunity cost

This cost is the hardest to quantify, but is important to keep in mind. If a software solution can be procured in a month and set up within another month, the opportunity cost of getting the solution up and running is two months. This means that over those two months, any AI or ROI failures won't be detected and will potentially result in harm to the business.

Conversely, if a monitoring solution requires 12+ months to be built, tested, and deployed, there is a more than 6x chance of AI or ROI failure negatively impacting the business. This delay can be enough to make stakeholders lose confidence in the ML models being deployed, thereby hamstringing their ability to positively impact the business.

To understand how to estimate the impact of the opportunity cost of building a solution, take a look at the Deeper Dive: Timing section.

Do you have the resources to solve it?

Even if a solution can theoretically be built or bought, and even if the benefits of that approach are calculated to outweigh the costs, there remains one final roadblock to choosing to build a solution: resourcing. Without the correct resources, even if the cost story makes sense, nothing can get done.

Building

In the case of building a solution, the primary resources are human resources. Simply put, do you have the talent and are you willing to pay their associated labor cost for building this tool yourself?

Although in-house monitoring often begins as somebody's side project, it inevitably grows into a team of 3+ people. The more widely used the monitoring solution is, the larger the team grows. For teams that are choosing to build a monitoring solution in-house should make a case for 3-5 engineers to be dedicated to the task in perpetuity.

Because talented model operators are in high demand, the available talent to fill these roles is limited. Enterprises should be careful not to rely too heavily on a few technical employees, which can expose them to the risk of significant losses if those employees leave. Compounding the challenges of recruiting and retaining a talent pool, ML teams often require members with highly-specialized skill sets, in topics like scalable infrastructure and AI explainability.

Buying

The bigger resource challenge for buying a solution is budgetary discretion. While it takes fewer human resources to evaluate and implement a solution, because of the higher sticker price, organizations may limit your ability to procure a solution. You might have to work with a dedicated procurement team or make a procurement request from your management chain to purchase a new tool.

Fortunately, ML monitoring vendors are experienced with helping data scientists and machine learning engineers navigate the complexities of organizational hierarchy and make sure that you have access to the tools that you need to be successful. By working with a vendor, you have a partner who is committed to your success.

Deeper Dive: Features

As we alluded to in the Build or Buy section above, scoping a home-brewed monitoring tool or deciding which tool to purchase starts with identifying and cataloging the features that would be required from such a tool. Although virtually all monitoring tools share a number of core features, different organizations may have different needs for specific features. Understanding your unique set of requirements is crucial for understanding whether to build your own solution or which solution to procure.

We've broken down specific feature requirements into a few categories:

1. Integrations
2. Support for types of data
3. Anomaly detection

Although the list of features that we enumerate isn't comprehensive, we have found that they do account for the vast majority of differences between different homebrewed and vendor-provided ML monitoring solutions.

Integrations

Integrations are key to the usefulness and usability of an ML monitoring tool. As highlighted in the MLOps Stack section, ML monitoring is just one piece of the puzzle for a machine learning platform. Being able to integrate with other components of the stack is, therefore, crucial for an ML monitoring service to be successful. Similarly, integrating with notification (e.g. email, Slack) or workflow (e.g. PagerDuty, OpsGenie) tools is crucial for ensuring that the insights generated by the automated monitoring service are acted on.

Model deployment services

Model deployment services are managed services (generally either SaaS or PaaS) which allow users to deploy a trained machine learning model, so that the model can generate predictions in response to new data. This often involves creating a REST API endpoint that the services connected to the ML model can pass data to and get a prediction.

Model deployment services are generally offered as either part of an end-to-end ML platform (such as GCP Vertex or AWS SageMaker) or as point solutions (such as Valohai or TeachableHub). In either case, integrating your ML monitoring solution with your model deployment service is key to ensuring that the data feeding your ML model and the predictions that it's making get monitored for data drift, data quality issues, performance degradation, etc.

At larger organizations, teams may use a variety of different model deployment services. In this situation, it's important to be able to integrate with all of them, so that all data scientists and machine learning engineers can easily monitor their models in production.

Model deployment tools

In addition to managed services for model deployment, there are also open source model deployment tools. These tools require a user to manage the deployment themselves, but provide a convenient wrapper around the machine learning model to make it easier for outside services to interact with.

Historically, data scientists used generic Python tools such as Flask and FastAPI to deploy models. Now, many are moving to dedicated ML deployment tools such as BentoML and Ray Serve, which offer ML-specific features beyond simply creating a web service or API endpoint.

Similar to model deployment services, monitoring solutions need to integrate with each model deployment tool in use at a company.

Data pipelining tools

But to truly understand an ML system, it's not enough to only monitor the model at the end of the system. Data flows through pipelines before reaching the model in question, and monitoring that data in those pipelines is crucial to debugging problems that may be created upstream of the model itself.

Data pipelining tools include DAG-managers such as Airflow and Prefect. In addition to these DAG-managers, it is also useful to integrate with the computation engines which are actually moving or transporting data before it gets to the machine learning model. For batch systems, this may mean integrating with Apache Spark or data warehouses such as Snowflake. For streaming systems, the message buses and stream processors such as Apache Kafka and Apache Flink are commonly used.

Investigate which data pipelining tools are in use at your company and whether the tool you are thinking of building or buying can integrate with those.

Notification tools

Once an anomaly is detected, an alert needs to be sent to the machine learning engineer or data scientist in charge of the model, so that they can debug the problem with their model and begin to remediate the issue. To ensure that the user will receive the message, the ML monitoring system needs to integrate with notification systems (such as email or Slack) to alert the user about anomalies.

Trigger integrations

A step beyond notifications, trigger integrations are integrations with managed workflow tools (such as PagerDuty and OpsGenie), which allow users to create automated if-then responses to anomalies. An example of this may be automatically retraining an ML model once a certain level of drift and performance degradation have been detected. Doing this automates the ML lifecycle and makes maintaining model performance in production easier.

Support for types of data

In addition to key integrations, support for key types of data is crucial for an ML system's usability. If your organization is using streaming data, unstructured data, or distributed data, ensure that the tool you build or buy can support that type of data.

Streaming data

Streaming (or real time) data is data that gets extracted, processed, moved, and inferenced on as it gets generated. This is in contrast with systems which rely on "batch data", where data would be allowed to accumulate into batches (hourly, daily, weekly, or even monthly) before being interacted with.

Support for streaming data means more than just integrating with streaming tools, such as Apache Kafka and Apache Flink. A platform has to be architected with real-time capabilities in order to provide valuable insight about how a model is performing on streaming data.

Unstructured data

While most monitoring systems are built to monitor structured data (i.e. data that can be represented in tabular terms, with each data point sitting at the intersection of a row and a column), not every system can support unstructured data types. Many great advances in ML systems in the last decade have come in supporting unstructured data, from computer vision models supporting image data to natural language models supporting text data.

Some vendors claim to support unstructured data by offering visualizations of embeddings in their UI. Most models for unstructured data do not use embeddings and therefore can't be monitored by a system that relies only on embeddings data for monitoring. Even for models which do use embeddings, insights from monitoring embeddings are much harder to understand than insights from monitoring the unstructured data itself, so it's worth investigating with data scientists on your team whether they will even be able to productively use an ML monitoring system which relies entirely on embeddings.

Verify what types of data your organization uses, and whether the ML monitoring solution you're building or buying can support that type of data.

Distributed data

Distributed data (data which sits on more than one machine) is notoriously hard to work with. The evolution of "big data" technologies (from MPP to Hadoop to Spark to Presto/Trino) have shown how much value there is in distributed systems, and how hard they are to effectively manage. Many ML systems at large scale rely on distributed data technologies, such as Apache Spark, to handle data at the scale that's needed.

Monitoring systems need to scale to match the data and machine learning models that they're monitoring. While it can be easy to construct a system that monitors a single model at a small scale, many vendor-built systems flail when it comes to monitoring large, distributed datasets. They rely on techniques like sampling, which worsen the performance of the monitoring system. If you are dealing with data at large scale, build or look for vendors who have built solutions with large scale in mind.

Anomaly detection

One of the key elements of a monitoring system is anomaly detection: the ability to pick up on changes in data which indicate potential AI or ROI failure. While nearly all monitoring systems have some form of anomaly detection, not all anomaly detection algorithms are equally effective. Inferior anomaly detection algorithms result in a lower accuracy rate, resulting in more false negatives (missed AI or ROI failures) or more false positives (resulting in alert fatigue, in turn makes users less likely to respond to true positive issues). It's important to evaluate anomaly detection algorithms when comparing a system you could build yourself in-house vs different vendor-provided systems.

In addition to the anomaly detection algorithm, a related feature to explore is the metrics used to evaluate data drift. Data drift (when data changes over time) causes AI and ROI failure by decreasing the performance of a model over time. It can be subtle but insidious, continuously degrading the performance of the model until it is no better than a coin toss. To evaluate for data drift, there are a number of different drift detection metrics that can be used and fed into the anomaly detection algorithm. Making sure that your monitoring solution has support for the right one is an important piece of the puzzle for evaluating which monitoring system to pick.

Anomaly detection algorithms

There are three broad categories of anomaly detection algorithms:

1. Static threshold
2. Variance-based threshold
3. Learned threshold

These three categories of algorithms encompass nearly every anomaly detection algorithm deployed in ML monitoring systems today. They are ordered from least sophisticated to most sophisticated.

Static threshold

The simplest tools that could be used for ML monitoring rely on "anomaly detection algorithms" that are actually merely static thresholds. These static thresholds require a user to look at their historical data and make a decision about what an acceptable threshold is for change in each metric. This requires a lot of manual work on the part of the user setting up the system, because for each metric about each feature in the model, an acceptable threshold must be defined. This type of system is easy to implement, but hard to use.

Variance-based threshold

A more sophisticated anomaly detection technique relies on calculating the variance or standard deviation in a dataset. This allows the user to simply set a sensitivity threshold, rather than needing to analyze the raw dataset metrics to determine an appropriate threshold. This technique is still somewhat prone to false negatives and false positives, however, because even a finely-tuned variance sensitivity might still miss important anomalies or alert on non-anomalies.

Learned threshold

The most sophisticated monitoring systems rely on learned thresholds. They take into account aspects of your data such as seasonality (especially important in fields where time series data is the norm, including verticals such as retail) and allow users to continuously improve the anomaly detection algorithm by providing feedback to the model about whether an anomaly was relevant or not. This type of algorithm is hardest to implement because it requires effectively implementing a machine learning model to monitor your machine learning models, but it provides the best possible accuracy.

Deeper Dive: Pricing

When considering purchasing an ML monitoring solution, it's important to compare the different pricing structures offered by vendors. As was briefly discussed in the Sticker Price section, there are two broad categories of pricing structures:

1. Model-based pricing
2. Inference-based pricing

Model-based pricing charges customers per model that they deploy in production, while inference-based pricing charges per prediction that the model makes. As you can imagine, model-based pricing is almost always the cheaper option for models with high throughputs. But even for lower-throughput models, inference-based pricing can get expensive quite quickly.

In addition to these fundamental differences in pricing models, vendors may also charge for different add-on features, such as:

1. Number of users allowed to access the platform
2. Premium support
3. API access
4. Number of model features
5. Particular platform features (algorithms, dashboards, etc)

Nonetheless, the majority of cost for the sticker price of a vendor solution is driven by the fundamental SKU, which is either model-based or inference-based.

Model-based pricing

Model-based pricing charges per model that the user deploys and monitors, regardless of the size of the model itself or its throughput. This type of pricing is clearly the best choice for models with lots of throughput, but may also be preferable for models that process fewer inferences. This is because systems that can offer model-based pricing often relies on more advanced telemetry storage and monitoring techniques. This results in a lower COGS (cost of goods sold) to the vendor, which are savings that they can pass along to the customer.

Inference-based pricing

Inference-based pricing charges per prediction that the user makes. This pricing model is generally predicated on the fact that the monitoring system stores raw data, resulting in higher storage and computation fees for the vendor which they must pass along to their customers.

This type of pricing often results in sampling, where users only send a subset of their data to the ML monitoring provider, rather than representing the entire dataset. Sampling results in decreased accuracy, which means that the system is less useful because it results in a higher number of both false negatives and false positives.

Deeper Dive: Timing

One often under-discussed aspect of picking an ML monitoring solution is figuring out how long it will take before the solution is in place. This timing can be hard to think about, because it requires probabilistic thinking about AI and ROI failure. However, by using a simple expected value calculation, we can determine what the potential cost of being slow to implement model monitoring will be for your business.

Say that there is a 1 in 100 chance every month of an AI failure which would result in a loss of $10 million to your business. If it takes one month to implement a monitoring solution, this has an expected value cost of $100,000, because there is a 1 in 100 chance of you losing $10 million. However, if it takes 10 months to implement such a solution, this has a $1 million cost, because there is a 1 in 10 chance of losing out on those $10 million.

This opportunity cost of implementing a monitoring solution should be taken into consideration when deciding whether to build or buy, or which vendor to buy from. Implementation times aren't just long and annoying, they have a direct impact on the ROI of a model monitoring solution.

Case Study: Airspace

Airspace is a high-tech logistics company which uses machine learning as a key part of their platform. With many business critical models in production, they explored building out a custom monitoring solution but decided that it would be hard to resource and a distraction from their core priorities. They use WhyLabs to monitor all of their productionized machine learning models, monitoring hundreds of metrics with zero configuration.

Airspace had three primary considerations which caused them to select WhyLabs as their ML monitoring vendor:

1. Support for complex deployment architectures
2. Data privacy preservation
3. Ease of setup and use

Airspace built a sophisticated deployment infrastructure which requires monitoring to ensure the performance of their proprietary models doesn't degrade. WhyLabs provides a flexible monitoring solution compatible with the requirements of Airspace's sophisticated and unique deployment infrastructure. By allowing Airspace to monitor for issues such as feature drift, training-serving skew, and missing features, which were some of the biggest drivers of poor model performance, WhyLabs satisfied Airspace's feature requirements.

Airspace also considered data privacy and security considerations when selecting between building their own ML monitoring solution or picking among different vendors. Because WhyLabs uses statistical summaries of data, rather than raw data itself, it's able to ensure data privacy and security. In this way, WhyLabs satisfied Airspace's security considerations.

Finally, Airspace considered the labor costs associated with not only building their own solution, but managing, maintaining, and running it in production. Because WhyLabs runs as a wholly managed service, WhyLabs' customers don't need to worry about managing or maintaining the software. Additionally, with one-click and programmatic setup options for monitoring with WhyLabs, it's trivial for users to onboard new models.

Airspace carefully considered their options about how to monitor their mission-critical machine learning models in production. Ultimately, they decided to use WhyLabs, because they found that it was the fastest and least expensive way to get the monitoring that their ML models needed.

Conclusion

There is no one-size-fits-all solution when it comes to monitoring ML models. For some, requirements may be so complex and unique that no tool on the market will satisfy their needs, so building a solution is their only choice. For others, monitoring needs are so minimal that they can rely on the minimal monitoring solutions provided by end-to-end ML platforms. For the rest, best-in-breed ML monitoring point solutions offer a cost-effective way to meet all of their monitoring requirements.

Rather than try to present a single recommendation for a one-size-fits-all solution, we've written this build vs buy guide as a way to help our readers evaluate the best option for them. We've worked with dozens of companies who have gone through this process, some of whom opted to use the WhyLabs AI Observatory or our open source data logging library whylogs for their ML monitoring needs.

Want to learn more about WhyLabs or brainstorm on ways to improve your organization's ML monitoring processes?

Contact our experts

For out-of-the-box model and data monitoring, sign up for a free starter account.

Get started at whylabs.ai/free