The Verta Guide to

The AI/ML Model Lifecycle

Verta

Table of Contents

Introduction

Ask different stakeholders inside of a machine learning-enabled organization to describe the model lifecycle and you'll likely get wildly different responses.

Data scientists might talk about refining the predictive power of a model by fine tuning its hyperparameters.

ML engineers or platform engineers might speak in technical terms about managing containerized applications or integration with CI/CD processes.

Governance or risk management might talk about the importance of tracking and monitoring ML models running in production, to cut off performance degradation or other issues before they can have adverse impacts.

Who's right?

Turns out, every one of the above stakeholders is correct — which demonstrates how multifaceted and complex the model lifecycle actually is. Due to their complexity, machine learning models rely on unique contributions from various highly specialized stakeholders across the model lifecycle. In this ebook, we'll examine these steps and draw a few conclusions about best practices.

For the uninitiated, let's begin by defining a few key terms:

A **machine learning model** is a file that has been trained to recognize certain types of patterns.

Models contain **algorithms**, or procedures that are implemented in code to process data.

A **model** is the output of an algorithm. A model represents what was learned by running an algorithm on data.

Models are **trained** on a known set of data, adjusting algorithms until the models can reason over that data. Once a model has been trained, it can be used to reason over new data sets and make predictions.

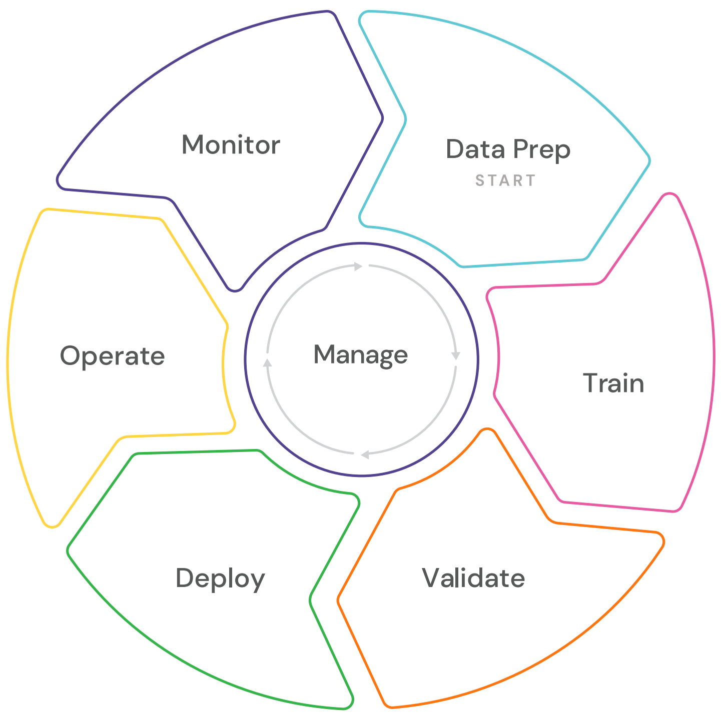
The Verta Perspective

Different people and organizations describe the steps and processes in a model lifecycle in different ways, using different terminology. At Verta, we have a unique perspective on machine learning. We've worked with hundreds of stakeholders across organizations at almost every phase of the model lifecycle — so our view is informed by a fulsome understanding of all parts of the process of building and operationalizing machine learning.

In our view, the full model lifecycle contains six distinct phases, tied together by the overarching goal of management:

1. **Data Preparation**
2. **Train**
3. **Validate**
4. **Deploy**
5. **Operate**
6. **Monitor**

Introducing the AI/ML Lifecycle



For a better understanding of how these phases work together to create functioning ML models, we've created the above visualization.

However, we present it with a caveat. Although the process might look fairly straightforward in this illustration, in practice model lifecycles are rarely simple or linear. Stakeholders often have to move forward and backwards between steps in order to create and maintain a working model. But before we venture into that discussion, let's first examine the different phases in the model lifecycle in a bit more detail.

Step 0

Manage

All phases of the model lifecycle are essential for creating and operationalizing machine learning.

But in isolation, they are like individual spokes on a wheel. Each of these phases is unique and essential in its own right. But to truly add momentum to ML product development, all phases must be organized into a coherent whole.

We believe ML teams need a hub to connect the different phases of the model lifecycle, which is why this overarching phase — Manage — is so important. Model management weaves everything else discussed in this article together, giving organizations visibility as well as control over the model lifecycle and enabling better decision-making and operational flow.

Model management allows a team to:

View the entire model lifecycle and a consistent, unbroken chain of events

When MLOps teams track the entire lineage of where a model came from (including what data was used to train and validate), how the model was trained, and which model was finally deployed, they can find the root of the problems and correct issues in production more quickly.

Manage inventory of models in production

The ad-hoc nature of many model production pipelines and the presence of multiple model deployment environments often hurt visibility. Many companies don't have a single view of what models are currently deployed in production. As a result, resolving production errors and answering governance questions can be challenging, if not impossible. With a single model management system, relevant stakeholders can get a bird's eye view of the models currently in use.

See which models are not behaving as expected

With visibility into monitoring of different individual models, teams can make better decisions about which models to pull from production and when action is required to fix or update running models.

Bring governance to the entire ML process

Often, governance tends to be an afterthought for many business and software processes. But it shouldn't be when it comes to models — which are unique in their vast power, black-box nature, and ubiquitous application. As a result, governance must be baked into the ML product development process from Day-1 and prevent reputational and financial damages later.

Step 1

Data Preparation

If you've read this far, you already understand that machine learning is all about data.

A model is always an output of two things: algorithm and data. So before we can do anything else in the model lifecycle, we have to transform our raw data into a format that's most useful to a model. This process is called Data Preparation or Data Prep in short.

Data Prep typically includes a variety of steps depending on the data source and its complexity. For example, you may need to acquire new data, label existing data, clean data or perform ETL (extract, transform, load) processes. Raw data is messy and often rife with issues that can hurt (or even destroy) model performance. As a result, data must be cleaned up in a variety of ways to make it ready for modeling.

Along with data cleaning, feature engineering is a crucial part of any data prep process. A feature — also referred to as a "variable" or "attribute" — is a characteristic that can be used to build a model. (For example, if a data set contains information about a number of human users, features might include Name, Age, Sex, etc.) Feature engineering is the process of extracting features from raw data based on knowledge of the data set and the purpose of the intended model. As you might suspect, feature engineering is half art and half science — as model effectiveness often depends on engineering the right features.

The Verta Perspective

One of the crucial requirements in Data Prep is making the steps repeatable for model training and model serving scenarios. Investing in infrastructure like feature stores or data warehouses for machine learning can significantly alleviate this pain and assist with productionizing models later in the lifecycle.

Step 2

Train

With useful, relevant data in hand, a data scientist can embark on the process of training a model.

Typically, this step in the model lifecycle involves rapid experimentation, testing and iteration to find a model with the greatest predictive power.

To start, a data scientist will usually evaluate about half a dozen algorithms to identify which one is most accurate and effective. When training neural networks, a data scientist will also experiment with different network architectures. Training is rarely a linear process. It almost always requires the data scientist to go back and forth between training, data prep, and validation (more on that in a minute).

Model training involves **hyperparameter tuning**. Hyperparameters are variables — such as a learning rate — that directly affect the way a model learns or trains. Data scientists “tune” a model by trying different variations and combinations of hyperparameters to optimize performance. In the training phase, speed and creativity are crucial. The more quickly a data scientist can try new combinations of data and ideas, the better the model will perform. Teams should seek to remove as many barriers to iteration as possible.

The Verta Perspective

Management of this phase of the model lifecycle requires that organizations thoroughly document everything that goes into the training process, including scripts, configurations, performance metrics, hardware data, etc. Often, organizations will include additional information (such as feature importance) to help to answer questions about explainability.

Step 3

Validate

This phase is a bit like a pilot's "preflight" check to make sure a model is working correctly before it continues through the model lifecycle.

As you might predict, data scientists validate how well a model performs at this phase (hence the name of this step).

At a high level, validation might seem relatively straightforward — simply test performance of overall model predictions. But that would be a mis-characterization. There's actually quite a bit more to model validation than a simple binary pass/fail test. For example:

- Data scientists frequently conduct backtesting — testing a predictive model on historical data — to see if it makes correct predictions
- Validation testing can also be performed on different subsets of data (or "cohorts"). For example, on a social media platform, new user behavior can be quite different from that of established users. A high quality model will recognize and know what to do with either type of user.
- The validation phase can also look at robustness of a model, e.g., What is the amount of data error that the model can work with without giving erroneous results? How accurately does a model predict on an outlier?
- Increasingly, validation also includes performing explainability and bias checks on a model.

The Verta Perspective

In our opinion, this is one of the steps in the model lifecycle that many organizations don't do very well or thoroughly enough. Oftentimes, teams hurry through validation after a handful of high level tests that often don't stress models enough to get a sense for how they will perform (or not perform) in the wild — which can lead to significant problems downstream.

Step 4

Deploy

Once a ML team has arrived at a trained, validated model, then the real fun begins.

That model needs to be deployed into production — or moved out of a development sandbox and into the real world. If it helps, you can think of Deploy using a ship metaphor — it's when the ship takes off from port.

In simple terms, this means packaging the model and running it. But any ML engineer worth her weight in salt will quickly point out that deployment involves a number of factors that need to be considered. For example:

1. Packaging models and their myriad dependencies into a consistent runnable package (Due to the nature of ever-evolving Python dependencies, containers have emerged as the best practice in deploying models.)
2. Adding scaffolding code to actually serve the model and export metrics and logging
3. Setting up the rest of the infrastructure to accommodate the new model service (e.g., API gateways, services, deployments, IAM)
4. Integrating the deployment of models into existing CI/CD processes and tools
5. Finally, the model may need to be deployed not in a container but into a specific data processing system, e.g., Spark or Kafka; each system brings with it specific demands on packaging and execution

The deploy phase in the model lifecycle marks an important handoff between data scientists and ML engineering teams, as the skills that make a great data scientist are distinct from those that make a good developer/engineer.

The Verta Perspective

In our experience, deploying a model can be a significant pain point in the model lifecycle for many organizations. Oftentimes, processes for deploying models to production are unclear — and some teams end up "reinventing the wheel" each time they serve a model — which is highly inefficient and opens room for error. Investing in standardization at this step can significantly reduce time to time while increasing safety.

Step 5

Operate

Ideally, the Operate phase is where a model would spend the majority of its time.

When a model is operating effectively, it serves accurate, useful predictions quickly and responds to changes in both demand and data. This phase of the model lifecycle is where businesses derive value from their models.

Operating a model requires constant attention to a number of considerations:

Autoscaling

Depending on workloads, a model needs to be able to elastically scale up and down to meet dynamic demands. Models need resources and should be able to expand resources to address needs.

High availability

Scaling models up and down requires that models be able to replicate dynamically to serve multiple requests. When models scale down, they need to be able to do so without coming to a screeching halt.

Performance

Production models have strict requirements around how fast they need to serve predictions (e.g., a pricing model must return predictions within milliseconds) and how many predictions they should be able to serve in parallel. Commonly called latency and throughput, these metrics are the core SLAs (service level agreements) that models are expected to meet. A large part of operating models is ensuring that model performance meets SLAs.

The Verta Perspective

Returning to the nautical metaphor, while operating the "ship is underway," running and providing expected insights and predictions.

But, like a captain of a ship, just because a ship is moving through the open ocean, it does not mean that she can take her hands off the wheel or fall asleep on the bridge. Effective model operations, requiring constant attention to scale and performance, is crucial for businesses to get value from their models and meet SLA metrics.

Step 6

Monitor

The benefits of machine learning are well-documented. But operationalizing ML at scale also comes with a risk.

For example, last year a financial company began to lose $20,000 every ten minutes because one of its machine learning models began to misbehave. When a model begins to degrade, it can do so quickly, without warning and with disastrous effects.

Whenever a model is running in production, ML teams need to always have a view of real-time model performance — and should be notified immediately when performance begins to degrade. KPIs for model performance vary from model to model, but some of the more important metrics include:

System-Level Metrics

These relate to system-level SLAs discussed above such as latency and throughput.

Model Quality Metrics

These metrics combine model predictions with ground-truth to produce metrics such as accuracy, precision, recall, F1 score, both by the full dataset and by cohort.

Model Data Metrics

Models are meant to be used with data that looks like the training dataset. As a result, changes in input data and changes in prediction data are essential to track. Drift metrics can include distributional shifts, outliers, changes in a number of distinct values, null values, etc.

Ultimately, the results of the monitoring phase are used to further optimize and improve the model. For example, monitoring may indicate that the input data has shifted from the training data, indicating that a retraining is in order. Monitoring may also indicate that the model mis-predicts on certain types of data. In that case, model debugging may indicate that the model or data needs to be updated to account for the errors.

The Verta Perspective

Metrics and techniques used for model monitoring can vary widely from model to model. Monitoring systems should be customizable, extensible, and allow you to monitor a variety of data types. And they should provide you with flexibility to compute and visualize data in different ways for different audiences.

Conclusion

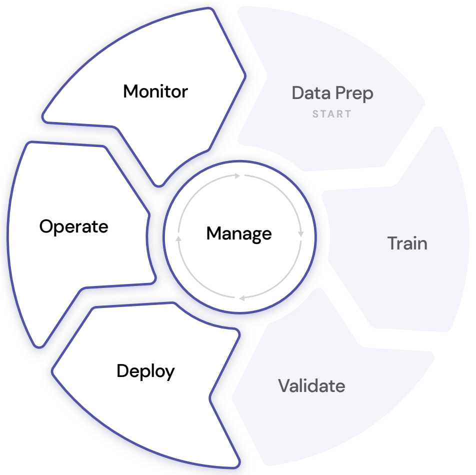
Verta enables rapid and safe operationalization of models across all phases of the AI/ML lifecycle.

The model lifecycle is complex and highly collaborative, drawing on unprecedented amounts of data, multiple data pipelines and specialized expertise across disciplines. But complexity can stand in the way of collaboration, especially in the rapidly evolving AI/ML discipline.

When individual phases of the model lifecycle are siloed, tools aren't interoperable, or teams lack visibility, the lifecycle often breaks down. Many companies have the technical ability to create powerful predictive models — but fail to realize the full value because they can't properly operationalize what they've built.

Unlocking value from predictive technology depends on optimizing the entire model lifecycle. Teams must invest either in platforms that cover multiple phases of the lifecycle or develop processes and tools that connect the lifecycle stages. The ultimate success of AI initiatives will depend on an organization's ability to efficiently perform the "Manage" phase of the Model Lifecycle.

The Verta AI/ML Model Lifecycle



Verta allows high-velocity data science and machine learning teams to deliver and manage AI and ML at scale more quickly and safely than ever. We offer the **only platform that lets teams manage, deploy, and monitor models in one place,** enabling greater speed, safety, visibility and control across the AI/ML lifecycle.

LEARN MORE
www.verta.ai

Schedule a demo

Verta