Building a MLOps System for Minimal Maintenance

When it comes to machine learning (ML), many organizations fail to realize maintainability is a critical part of the equation. About 90% of ML models never make it to production, and once they do, maintaining ML solutions is just as complex and burdensome as getting said models to production.

Many ML teams spend more than 50% of their time maintaining models, and even developers at Google call machine learning “the high-interest credit card of technical debt.” Luckily, companies have been working on these problems since at least 2014, giving rise to the discipline of machine learning operations (MLOps).

What is MLOps?

MLOps is a set of practices to combat issues with productionizing and maintaining models in machine learning. It borrows the ideologies of version control, automation, and CI/CD from DevOps, but has an added layer of complexity due to the use of data and artificial intelligence.

MLOps is non-negotiable for organizations attempting to implement ML sustainably at scale. Those who prioritize automation best practices early can quickly derive value with a model in production, while setting themselves up for long-term success by spending less time on pipeline management.

4 Phases for a Sustainable MLOps Approach

MLOps draws from DevOps best practices that have been refined over the last decade, and many solutions now exist to get models into production and monitor and maintain them once there. However, even with today’s technology solutions, it can take an average of six months for an artificial intelligence (AI) project to get off the ground, which may cause organizations to shy away from ML/AI adoption. With this in mind, we offer an approach for organizations to operationalize their ML models methodically in four phases, accounting for the state of the organization’s specific ML resources and needs.

ML teams should have a defined MLOps roadmap before sending a model to production to reap the benefits of MLOps, which include quick feedback loops to replace stale models, drastically reduced maintenance time, and experiment tracking and data versioning for discoverability and transparency. While every organization has a unique set

of circumstances, we are confident that any organization can harness the power of ML responsibly and sustainably with Credera's phased MLOps approach:

- Phase 1 - Construct a Pipeline Structure That Can Build the Model Itself
- Phase 2 - Orchestrate the Feedback Loop
- Phase 3 - Institutionalize Experiment Tracking and Data Versioning
- Phase 4 - Build and Deploy with a Management Tool

Phase 1 – Construct a Pipeline Structure That Can Build the Model Itself

Initial model development will likely follow a typical data science workflow, starting with exploratory data analysis, problem specification, and ad hoc model. At this stage, you could use a notebook (Jupyter being the most popular choice) to get a sense of what data sources are available, their different characteristics, and what types of models to leverage.

Once you've identified your data, try training a few simple models and see how your model performs on a test set. If a model performs well enough to provide value and there is not any further feature engineering or normalization to implement, take the code that exists in separate notebooks and combine their functionalities to build a first pass at a codebase.

The goal is to build a modularized, automated model training pipeline that can be reproduced in a production environment. Having a simple model with a solid MLOps infrastructure will derive value faster than throwing a complex model over the fence. Instead of handing off a model artifact to be productionized, reframe your deliverable expectations. The deliverable is a pipeline structure that can build the model itself. This code is easy to package into a Docker image for automated deployments and requires documentation around all dependencies and environment requirements for it to function.

Phase 2 – Orchestrate the Feedback Loop

As previously mentioned, getting models to production is only half the battle. Not only are ML teams spending most of their time and resources maintaining deployed models, a large portion of those models that make it to production become stale because organizations don't prioritize their maintenance until it is too late.

An ML team should never deploy a model if there is not already a defined method for that model to be maintained efficiently. A successful feedback loop can train a fresh model at will on new data in an automated way and compare performance metrics with the existing model, replacing the old model if desired.

To orchestrate the MLOps feedback loop:

1. **Creating Alignment**: Align with business partners and collaborate across data science, ML engineering, and product development teams. Leaders at all levels of your company need to reinforce the importance of cross-functional collaboration, creating a culture of cooperation from the top down. Many decisions will have to be made regarding data collection, data transformation, model deployment, and business metrics for model validation and performance monitoring and it is important to have perspectives from across the organization when making these decisions
2. **Automating Validation:** Build automated data and model validation steps into your ML pipeline. Fix data schema issues and deploy the changes to the automated pipeline. Let data value skews (changes in their underlying statistical properties like mean or variance) inform decisions to retrain your model.
3. **Training and Tracking**: Decide, with your business counterparts, what changes in performance or data quality should trigger a model retrain, and make sure you are tracking those metrics. Model drift is eventually going to happen because of fundamental changes in the underlying data your model was trained on. Your customer base could change, or they could behave differently over time. Variables your model heavily relies on could skew as your product sales grow over time.

4. **Removing Steps:** While optional, we recommend moving as many data transformation and extraction steps as possible to an ML data platform (similar to a feature store). This optimal data platform is responsible for providing consistent, high-quality data to the automated model training pipelines (in your experimental and production environments) and to the model being served in production. Data consistency is fundamental to reproducibility and will reduce the multitude of maintenance headaches that come with data quality issues.

Phase 3 – Institutionalize Experiment Tracking and Data Versioning

At this stage, you should already be tracking business metrics for model validation and monitoring, but now it's time to track everything. This starts with all the metadata from model training runs (in all environments) and model serving: code versions, dataset metadata, and training hyperparameters, to name a few. Some of this metadata is useful for your data engineers and DevOps team as they prepare for phase 4, while some of it is useful for your data scientists (now is the time for your data scientists to perform the more complicated ML approaches they wanted to try back in phase 1).

Tracking and logging outputs from each module in the ML pipeline unlocks the ability to resume a failed pipeline run from the most recent successful step, instead of having to re-execute from the beginning. Reuse experiment tracking for faster iterations during phase 1 of your next big ML project!

Versioning your data will empower your ML data platform to provide the consistent, high-quality data outlined in phase 2, drastically decreasing the turnaround time of maintenance issues and strengthening the reliability of your model validation steps.

While this may sound like a lot of work, there are robust open-source platforms such as MLflow, Kubeflow, and Pachyderm to simplify your experiment tracking and data versioning journey, as well as a wide array of enterprise solutions like Azure ML and AWS SageMaker. Many of these platforms will also support the final phase in your MLOps journey.

Phase 4 – Build and Deploy with a Management Tool

You are now ready to bring full continuous integration/continuous deployment (CI/CD) capabilities to your ML pipeline, automating your modularized model training and serving. A fundamental part of continuous integration is to flesh out unit and integration tests throughout your MLOps architecture. Some of these tests are unique to MLOps, such as those concerning model governance, triggers to train or deploy a new model, training on small datasets to test convergence, or performing model validation on a set of specific slices of data to evaluate model fairness or consistency across customer bases.

As is standard in DevOps, rollback strategies are a crucial part of continuous deployment, providing a plan to revert to the most recent 'safe' model in production should the new one start to misbehave. For MLOps, this could include detecting extreme model outputs, model failures, or degradation in downstream key performance indicators (KPIs).

If the model is performing poorly enough, or deteriorating in performance at an alarming rate, the trigger you set will automatically rollback to a prior model stored in the model registry. Multiple models can also be deployed simultaneously, acting as a shadow model or a live A/B test, validating the new model's performance in the wild before it replaces the previous model.

Let Your MLOps System Work for You

At this point, you will have an automated feedback loop for your productionized model, data and model versioning in experimentation, and an automated ML pipeline for re-training and productionizing your model. This sounds great on paper, but it may leave you questioning what this means for the reality of your machine learning efforts.

A mature MLOps strategy optimizes your ML processes by eliminating maintenance tasks for data scientists so they can pursue new business solutions and increase the validity and accuracy of your models. Collaboration between scientists and developers is increased throughout the process, and it is much easier to maintain data consistency with

this “human in the loop” type of process. It also frees up data science resources by automating jobs in the ML pipeline and simplifying troubleshooting when necessary due to the detailed logs created during model and data versioning.

Additionally, your “time to first prediction” decreases with model version deployment since you don’t have to manually carry out the ML pipeline. Re-training and deploying models continuously, you are constantly improving the performance of model predictions.

Most organizations are experimenting with ML models, but few are productionizing them in a scalable manner. Those who invest upfront in crafting an MLOps platform to complement their data science prowess will save time, resources, and money, enabling them to quickly meet their customers’ changing needs.