2020 EDITION | UPDATED

Standardizing the Machine Learning Lifecycle

From experimentation to production with MLflow

mlflow™

databricks

Contents

Preface

Technology changes quickly. Data science and machine learning (ML) are moving even faster. In the short time since we first published this eBook, businesses across industries have rapidly matured their machine learning operations (MLOps) — implementing ML applications and moving their first models into production. This has turned ML models into corporate assets that need to be managed across the lifecycle.

That's why MLflow, an open-source platform developed by Databricks, has emerged as a leader in automating the end-to-end ML lifecycle. With 1.8 million[1] downloads a month — and growing support in the developer community — this open-source platform is simplifying the complex process of standardizing and productionizing MLOps. This updated eBook explores the advantages of MLflow and introduces you to the newest component: MLflow Model Registry. You'll also discover how MLflow fits into the Databricks Unified Data Analytics Platform for data engineering, science and analytics.

1 As of March 2020

CHAPTER 1: Machine Learning Lifecycle Challenges

Building machine learning models is hard. Putting them into production is harder. Enabling others — data scientists, engineers or even yourself — to reproduce your pipeline and results is equally challenging. How many times have you or your peers had to discard previous work because it was either not documented properly or too difficult to replicate?

Getting models up to speed in the first place is significant enough that it can be easy to overlook long-term management. What does this involve in practice? In essence, we have to compare the results of different versions of ML models along with corresponding artifacts — code, dependencies, visualizations, intermediate data and more — to track what's running where, and to redeploy and roll back updated models as needed. Each of these requires its own specific tools, and it's these changes that make the ML lifecycle so challenging compared with traditional software development lifecycle (SDLC) management.

This represents a serious shift and creates challenges compared with a more traditional software development lifecycle for the following reasons:

The diversity and number of ML tools involved, coupled with a lack of standardization across ML libraries and frameworks

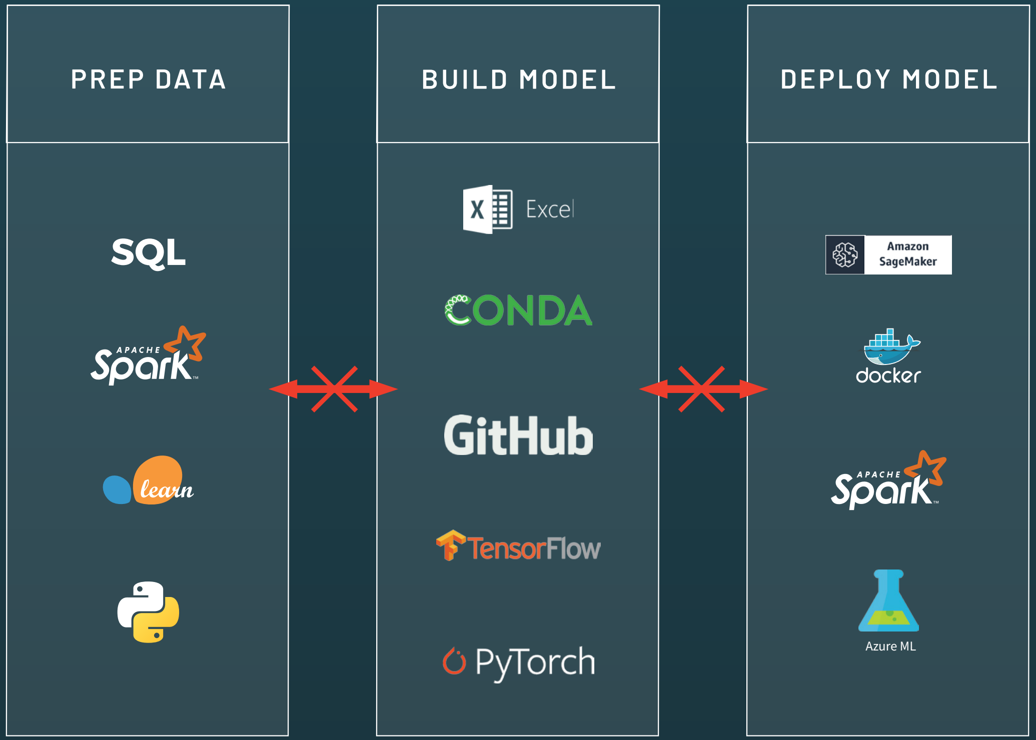
The continuous nature of ML development, accompanied by a lack of tracking and management tools for machine learning models and experiments

The complexity of productionizing ML models due to the lack of integration among data pipelines, ML environments and production services

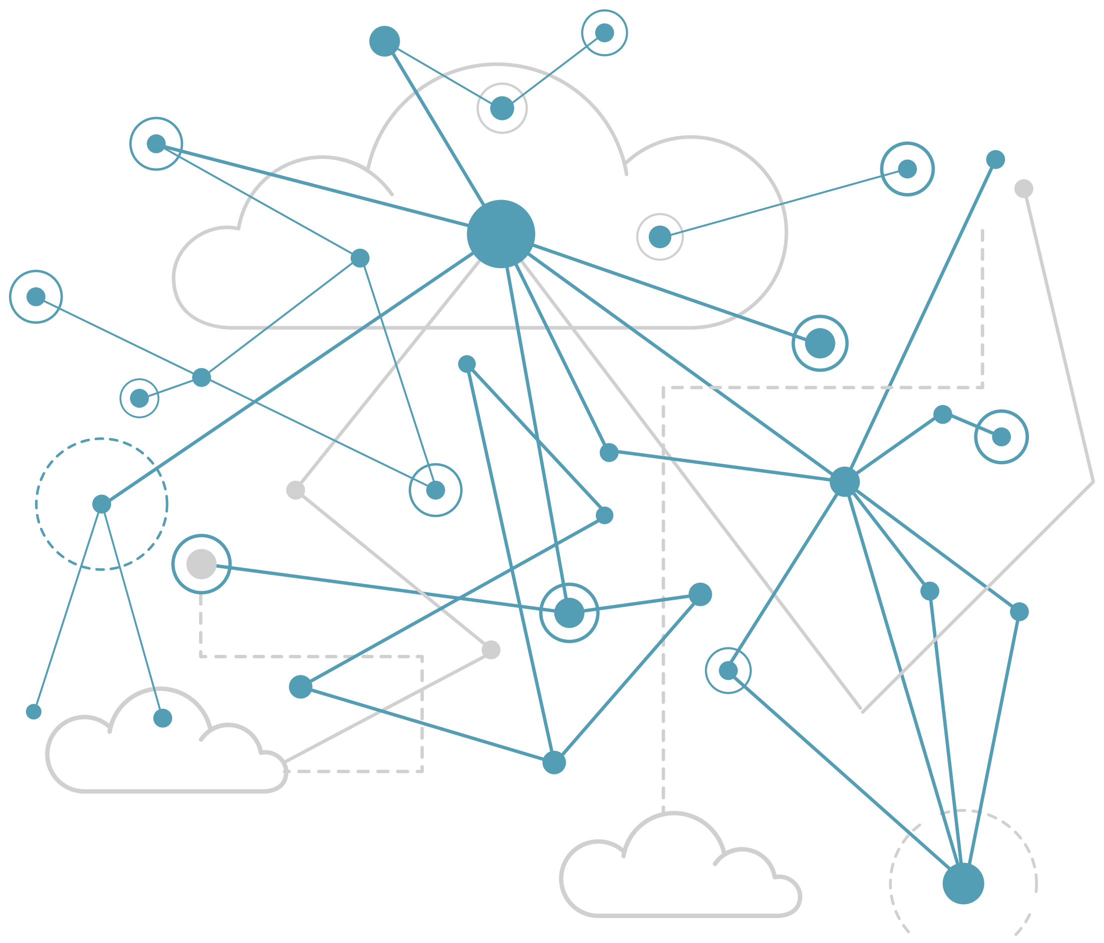
Let's look at each of these areas in turn.

The diversity and number of ML tools involved

While the traditional software development process leads to the rationalization and governance of tools and platforms used for developing and managing applications, the ML lifecycle relies on data scientists' ability to use multiple tools, whether for preparing data and training models, or deploying them for production use. Data scientists will seek the latest algorithms from the most up-to-date ML libraries and frameworks available to compare results and improve performance.



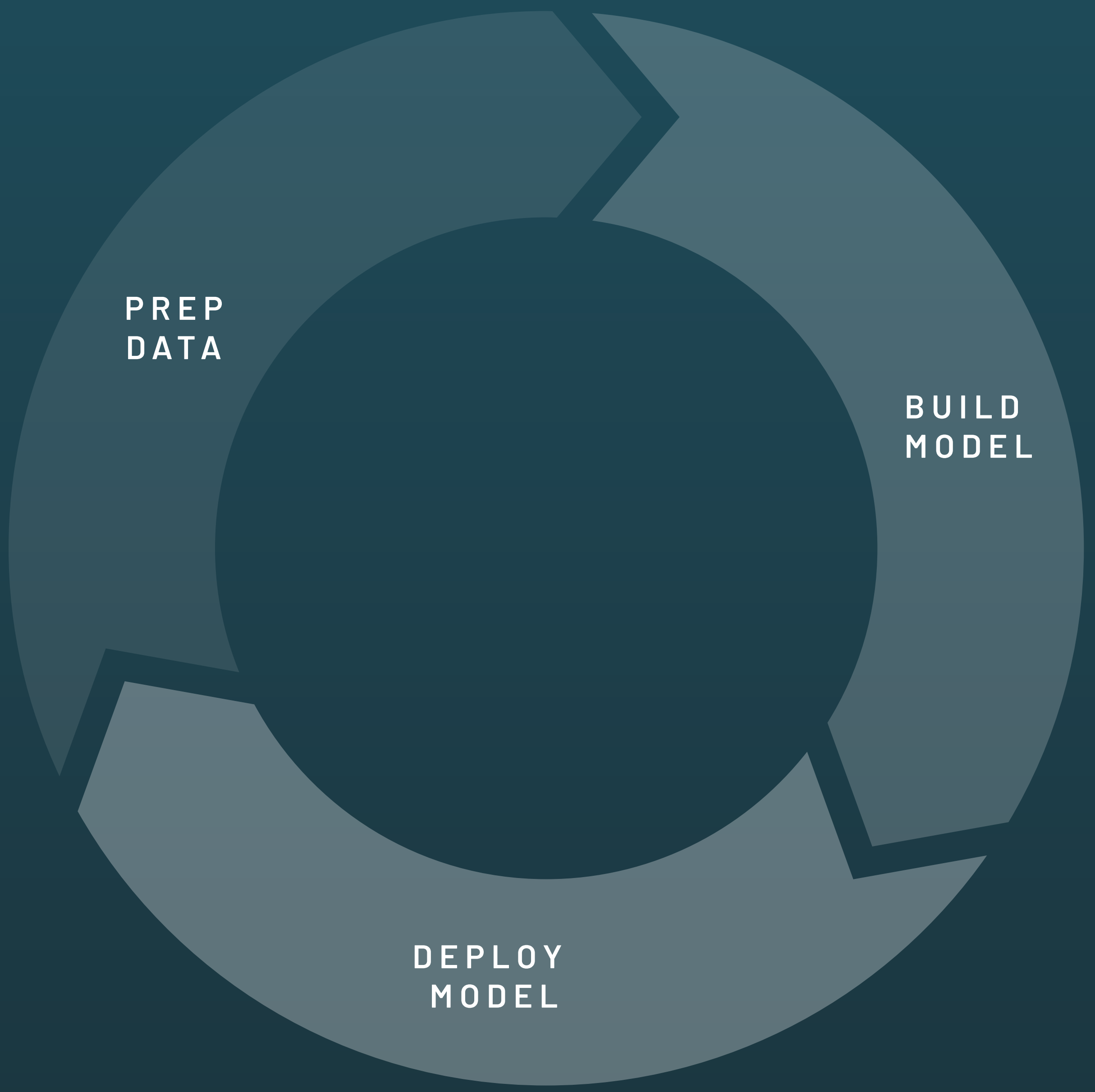
However, due to the variety of available tools and the lack of detailed tracking, teams often have trouble getting the same code to work again in the same way. Reproducing the ML workflow is a critical challenge, whether a data scientist needs to pass training code to an engineer for use in production or go back to past work to debug a problem.



The continuous nature of ML development

Technology never stands still. New data, algorithms, libraries and frameworks impact model performance continuously and, thus, need to be tested. Therefore, machine learning development requires a continuous approach, along with tracking capabilities to compare and reproduce results. The performance of ML models depends not only on the algorithms used, but also on the quality of the data sets and the parameter values for the models.

Whether practitioners work alone or on teams, it's still very difficult to track which parameters, code and data went into each experiment to produce a model, due to the intricate nature of the ML lifecycle itself.



The complexity of productionizing ML models

In software development, the architecture is set early on, based on the target application. Once the infrastructure and architecture have been chosen, they won't be updated or changed due to the sheer amount of work involved in rebuilding applications from scratch. Modern developments, such as the move to microservices, are making this easier, but for the most part, SDLC focuses on maintaining and improving what already exists.

With machine learning the first goal is to build a model. And keep in mind: a model's performance in terms of accuracy and sensitivity is agnostic from the deployment mode. However, models can be heavily dependent on latency, and the chosen architecture requires significant scalability based on the business application. End-to-end ML pipeline designs can be great for batch analytics and looking at streaming data, but they can involve different approaches for real-time scoring when an application is based on a microservice architecture working via REST APIs, etc.

One of today's key challenges is to effectively transition models from experimentation to staging and production — without needing to rewrite the code for production use. This is time-consuming and risky as it can introduce new bugs. There are many solutions available to productionize a model quickly, but practitioners need the ability to choose and deploy models across any platform, and scale resources as needed to manage model inference effectively on big data, in batch or real time.