Preface

Successfully building and deploying a machine-learning model can be difficult to do once. Enabling other data scientists (or yourself) to reproduce your pipeline, compare the results of different versions, track what’s running where, and redeploy and rollback updated models is much harder.

In this eBook, we’ll explore what makes the ML lifecycle so challenging compared to the traditional software development lifecycle, and share how to address these challenges with Azure Databricks.
Contents

The Machine Learning Lifecycle Challenges ................................................................. 4
   The Diversity and number of ML Tools involved ...................................................... 5
   The continuous nature of ML development ............................................................. 5
   The complexity of productionizing ML models ...................................................... 6
The need for standardization .................................................................................... 8
Introducing MLflow .................................................................................................... 9
   Key benefits ........................................................................................................... 9
Elements of the solution .......................................................................................... 11
   Experiment tracking ............................................................................................. 11
   Reproducible projects ......................................................................................... 11
   Model packaging ................................................................................................. 11
Open and extensible by design .................................................................................. 13
   Support for multiple programming languages ...................................................... 13
   Integration with popular ML libraries and frameworks ........................................ 13
   Integration with Azure Machine Learning .......................................................... 13
Image classification .................................................................................................. 14
Rapid community adoption ....................................................................................... 15
Making organizations successful with ML ............................................................... 16
Introducing Azure Databricks .................................................................................. 17
   Data Engineering .................................................................................................. 17
   Data Science ....................................................................................................... 17
   Providing managed MLflow on Azure Databricks ............................................... 18
   Getting data ready for ML with Delta Lake ......................................................... 19
   Getting to results faster with the Azure Databricks runtime for Machine Learning 20
Standardizing the Machine Learning lifecycle on Azure Databricks ....................... 21
Getting started ........................................................................................................ 21
Comparison matrix .................................................................................................. 22
Resources ................................................................................................................ 23

Copyright ©2020 Microsoft Corporation
The Machine Learning Lifecycle Challenges

Building and deploying a machine-learning model can be difficult to accomplish. Enabling other data scientists—or even yourself—to reproduce your pipeline is equally challenging. Moreover, doing so can impact the productivity of your data science team, leading to a significant waste of time and resources. 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 such a significant endeavor that it can be easy to overlook long-term management. What does this involve in practice? In essence, you have to compare the results of different versions of ML models to track what’s running where and to redeploy and rollback updated models as needed. Each of these requires its own specific tools. These changes are what make the ML lifecycle so challenging compared to traditional software development lifecycle (SDLC) management, which in itself isn’t all that easy, either.

This represents a serious shift, and challenges compare to those of a more traditional software-development lifecycle, for the following reasons:

- The diversity and number of ML tools involved, along with a lack of ML library and framework standardization
- The continuous nature of ML development, along with 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 between 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 without detailed tracking, teams often have trouble getting the same code to work again in the same ways. 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

Nothing ever stands still. New data, algorithms, libraries, and frameworks impact model performance continuously and thus need to be tested as mentioned before. 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 value of the parameters for the models.

Even when practitioners work as a team, tracking which parameters, code, and data went into each experiment to produce a model is still difficult. The intricate nature of the ML lifecycle itself and the lack of standardization of ML tools and processes are the reasons for this issue.

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 in software development, such as the move to microservices, are making this easier, but for the most part, SDLC focuses on maintaining and improving what already exists.
Conversely, in machine learning, the first goal is to build a model. For example, a model’s performance in terms of accuracy and sensitivity is agnostic from the deployment mode. However, applications 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 microservices architecture working via REST APIs, etc.

Therefore, one of the key challenges today is to effectively transition models from experimentation to production, without necessarily rewriting the code for production use, which is time-consuming and risky because it can introduce new bugs. Many solutions are available to put a model in production quickly. However, practitioners need to be able 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.
The need for standardization

Some of the world’s largest tech companies have already begun solving these problems internally with their own machine-learning platforms and lifecycle management tools. These internal platforms have been successful and are designed to accelerate the ML lifecycle by standardizing the process of data preparation, model training, and deployment via APIs built for data scientists. The platforms help standardize the ML lifecycle, and they also play a major role in retaining knowledge and best-practices and maximizing data-science-team productivity and collaboration, thereby leading to a greater ROI.

There are still limitations to internally driven strategies. First, they are limited to a few algorithms or frameworks. Adoption of new tools or libraries can lead to significant bottlenecks. Of course, data scientists always want to try the latest and the best algorithms, libraries, and frameworks, e.g., the latest versions of PyTorch, TensorFlow, and so on. Unfortunately, production teams cannot easily incorporate these into the custom ML platform without significant rework. The second limitation is that each platform is tied to a specific company’s infrastructure, which makes it difficult for data scientists to share efforts with one another. As each framework is so specific, options for deployment can be limited.

The question, then, is, can we address these limitations by providing the benefits of proprietary tools using an open platform? With such an approach, data scientists would be able evolve their ML models and keep pace with industry developments. Moreover, by making it available as open source, the wider industry will be able to join in and contribute to the wider adoption of ML. This also makes it easier to move between various tools and libraries over time.

---

1 Facebook has implemented its FBLearner platform, Uber has a service called Michelangelo, and Google has TFX.
Introducing MLflow

MLflow is a modular, API-first, open source framework designed to manage the complete ML lifecycle. Open and extensible by design, it works well with all popular ML frameworks and libraries.

With MLflow, data scientists can package code as reproducible runs, execute and compare hundreds of parallel experiments, and leverage any hardware or software platform for training, hyperparameter tuning, and more. They can track and share experiments locally or in the cloud and deploy models virtually anywhere. Also, organizations can deploy and manage models in production on a variety of clouds and serving platforms.

Azure Databricks provides a fully managed and hosted version of MLflow integrated with enterprise security features, high availability, and other Azure Databricks workspace features such as experiment and run management and notebook revision capture. MLflow on Azure Databricks offers an integrated experience for tracking and securing machine learning model training runs and running machine learning projects.

Key benefits

Experiments tracking
As mentioned previously, getting ML models to perform is a painstaking process of trial and error, continuous configuration, building, tuning, testing, and more. Therefore, it is imperative to allow data-scientist teams to track all that goes into a specific run, along with the results. With MLflow, data scientists can quickly record runs and keep track of model parameters, results, code, and data from each experiment, all in one place.
Reproducible projects

The ability to reproduce a project—entirely or just parts of it—is key to data-science productivity, knowledge sharing, and accelerating innovation. With MLflow, data scientists can build and package composable projects, capture dependencies and code history for reproducible results, and quickly share projects with their peers.

Model deployment

There are different ways to architect ML applications for production. Various tools can be used for deploying models, which often leads to code rewrites prior to deploying ML models into production. With MLflow, your data scientists can quickly download or deploy any saved models to various platforms locally or in the cloud, from experimentation to production.
Elements of the solution

Experiment tracking

A European energy company is using MLflow to track and update hundreds of energy-grid models. This company’s goal is to build a time-series model for every major energy producer (e.g., power plant) and consumer (e.g., factory), monitor these models using standard metrics, and combine the predictions to drive business processes, such as pricing. Because a single team is responsible for hundreds of models, possibly using different ML libraries, a standard development and tracking process is crucial. For development, the team can standardize on Jupyter notebooks. It can track metrics with MLflow and Azure Machine Learning for metrics and use Azure Databricks jobs for inference.

Reproducible projects

An online marketplace is using MLflow to package deep-learning jobs using Keras and run them in the cloud. Each data scientist develops models locally on a laptop using a small dataset, checks them into a Git repository with an MLproject file, and submits remote runs of the project to GPU instances in the cloud for large-scale training or hyperparameter search. Running MLflow projects remotely on Azure Databricks clusters using the MLflow CLI, makes it easy to create the same software environment in the cloud and share project code among data scientists.

Model packaging

An e-commerce site’s data-science team is using MLflow Models to package recommendation models for use by application engineers. The technical challenge here was that the recommendation application includes both a standard, off-the-shelf recommendation model and custom business logic for pre- and post-processing. For example, the application might include custom code to make sure that the recommended items are diverse. This business logic needs to change in sync with the model, and the data science team wants to control both the business logic and the model without having to submit a patch to the Web application each time the logic has to change.
Moreover, the team wants to A/B test distinct models with distinct versions of the processing logic.

The solution was to package both the recommendation model and the custom logic using the `python_function` flavor in an MLflow Model, which can then be deployed and tested as a single unit.
Open and extensible by design

Support for multiple programming languages

To give developers a choice, MLflow supports R, Python, Java, and Scala, along with a REST server interface that can be used from any language.

Integration with popular ML libraries and frameworks

MLflow has built-in integrations with the most popular machine-learning libraries, such as scikit-learn, TensorFlow, Keras, PyTorch, H2O, and Apache Spark MLlib, to help teams build, test, and deploy machine-learning applications.

Integration with Azure Machine Learning

You can use the MLflow tracking URI and logging API, which are collectively known as MLflow Tracking, to connect your MLflow experiments and Azure Machine Learning. With MLflow Tracking, you track an experiment’s run metrics and store model artifacts in your Azure Machine Learning workspace. These include local runs, remote runs, and Azure Databricks runs as shown in this graphic.

To link MLflow tracking to your Azure Machine Learning workspace, set the MLflow tracking URI. After linking, all your experiments will land in the managed Azure Machine Learning tracking service.
Image classification

If a company that delivers the technology that powers thousands of travel websites needs to classify tourism or hotel photographs, their data science can create an image classification model using MLflow integrated with Azure Machine Learning. In this scenario, the team can use the PyTorch deep learning library to train a classification model against MNIST data, while tracking the metrics using MLflow and monitoring them in Azure Machine Learning Workspace. The team can save the model MLflow’s framework-aware API for PyTorch and deploy it to Azure Container Instance using Azure Machine Learning Model Management APIs.
Rapid community adoption

1.7M monthly downloads

180+ code contributors

400+ contributing organizations
Making organizations successful with ML

Standardizing the ML lifecycle ensures that data scientists can share and track experiments, compare results, reproduce runs, and productionize faster.

To accelerate innovation with ML, you also need to:

- Ingest, ETL, and store big data
- Use state-of-the-art ML frameworks
- Scale compute from single to multi-nodes
Introducing Azure Databricks

Azure Databricks can unify data science, engineering, and the business. Through a fully managed, cloud-based service, the Azure Databricks Unified Analytics Platform makes it easier for enterprises to infuse AI in innovation initiatives.

### Data Engineering

Speed up the preparation of high-quality data, essential for best-in-class ML applications, at scale.

### Data Science

Collaboratively explore large datasets, build models iteratively and deploy across multiple platforms and deployment capabilities with enterprise reliability, security and scale.

```
<table>
<thead>
<tr>
<th>Azure Databricks</th>
<th>Data engineer</th>
<th>Data scientist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure Data Services integrations</td>
<td>Azure Data Factory</td>
<td>Azure Synapse Analytics</td>
</tr>
<tr>
<td>Power BI</td>
<td>Azure Data Lake Storage</td>
<td></td>
</tr>
<tr>
<td>ML Lifecycle Management integrations</td>
<td>MLflow</td>
<td>Azure Machine Learning</td>
</tr>
<tr>
<td>ML Runtime</td>
<td>PyTorch</td>
<td>TensorFlow</td>
</tr>
<tr>
<td>Optimized Databricks runtime engine</td>
<td>Databricks I/O</td>
<td>Delta Lake</td>
</tr>
</tbody>
</table>
```

“Databricks lets us focus on business problems and make data science very simple.”

—Dan Morris, Senior Director of Product Analytics at broadcast giant Viacom, which used Databricks to help it identify video quality issues, increase customer loyalty, and improve advertising performance.
Providing managed MLflow on Azure Databricks

By using managed MLflow on Azure Databricks, practitioners can benefit from out-of-the-box and seamless models tracking, packaging, and deployment capabilities with enterprise reliability, security and scale.

**Workspaces**
Collaboratively track and organize experiments from the Azure Databricks Workspace.

**Big Data snapshots**
Track large-scale data sets that fed models with Delta Lake snapshots.

**Jobs**
Execute runs as Azure Databricks jobs remotely or directly from Azure Databricks notebooks.

**Security**
Take advantage of one common security model for the entire ML lifecycle.
Getting data ready for ML with Delta Lake

Delta Lake is a storage layer that brings reliability to data lakes. Delta Lake provides ACID transactions, scalable metadata handling, and unifies streaming and batch data processing. Delta Lake runs on top of your existing data lake and is fully compatible with Apache Spark APIs. Delta Lake on Azure Databricks allows you to configure Delta Lake based on your workload patterns and provides optimized layouts and indexes for fast interactive queries.

By using Delta Lake, data engineers and data scientists can keep track of data used for model training.
Getting to results faster with the Azure Databricks runtime for Machine Learning

Azure Databricks Runtime for Machine Learning provides data scientists and ML practitioners with on-demand access to ready-to-use ML clusters that are pre-configured with the latest and most popular ML frameworks, including TensorFlow, Keras, Pytorch, scikit-learn, XGBoost, and Horovod.

Azure Databricks Runtime for Machine Learning lets you start an Azure Databricks cluster with all of the libraries required for distributed training. It ensures the compatibility of the libraries included on the cluster (between TensorFlow and CUDA / cuDNN, for example) and substantially speeds up cluster start-up.
Standardizing the Machine Learning lifecycle on Azure Databricks

Getting started

Sign up for a free Azure account to get started: https://azure.microsoft.com/free/databricks/

Learn more: databricks.com/mlflow

Join the community: mlflow.org
## Comparison matrix

<table>
<thead>
<tr>
<th></th>
<th>Open source MLflow</th>
<th>Managed MLflow on Databricks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Experiments tracking</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLflow tracking API</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>MLflow tracking server</td>
<td>✔ Self-hosted</td>
<td>✔ Fully managed</td>
</tr>
<tr>
<td>Notebooks integration</td>
<td>✗</td>
<td>✔ with Databricks workspace</td>
</tr>
<tr>
<td>Folder/file system</td>
<td>✗</td>
<td>✔ with Databricks workspace</td>
</tr>
<tr>
<td>Big data snapshot</td>
<td>✗</td>
<td>✔ with Delta Lake</td>
</tr>
<tr>
<td><strong>Reproducible projects</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLflow project API</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>GitHub &amp; Conda integration</td>
<td>✔</td>
<td>✔ with Databricks workspace</td>
</tr>
<tr>
<td>Scalable cloud/clusters for project runs</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Models deployment</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLflow model API</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Built-in batch interference</td>
<td>✗</td>
<td>✔ with Databricks runtime</td>
</tr>
<tr>
<td>Built-in streaming analytics</td>
<td>✗</td>
<td>✔ with Delta Lake</td>
</tr>
<tr>
<td><strong>Security and management</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automated updates</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Security model</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
Resources

Documentation
• Azure Databricks best practices
• Introduction to Delta Lake
• Introduction to MLFlow

Webinars
• Get started with Apache Spark
• Azure Databricks best practices
• Machine Learning life cycle management with Azure Databricks and Azure Machine Learning

More info
• Azure Databricks pricing page
• Azure Databricks unit pre-purchase plan

Contact us
• Talk to a sales expert
Invent with purpose