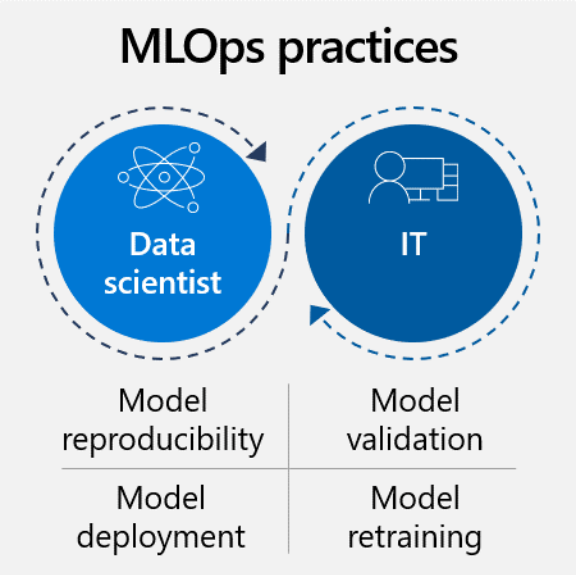
Microsoft

Discover the business value of MLOps

In this document, we discuss the importance of MLOps – the practice of collaboration among data scientists, ML engineers, app developers, and other IT teams to manage the ML lifecycle.

Business value of MLOps

MLOps processes and tools help those teams collaborate and provide visibility through shared, auditable documentation. MLOps technologies provide the ability to save and track changes to data sources, code, libraries, SDKs, and models. These technologies can also create efficiencies and accelerate the lifecycle with automation, repeatable workflows, and reusable assets.



Model reproducibility

During initial iterative training and later model retraining, there are a few things that can make the complex process more manageable. First, it's helpful to centrally **manage assets** like environments, code, datasets, and models so teams can share and reuse them.

- **Model registry:** As teams experiment with different versions of a model, a model registry provides a central place to save each version. With a registry, teams can easily revert to a previous version if something isn't working, even after the solution has gone into production. The model registry also serves as an audit trail for each model's history and makes it possible to automatically trigger workflows after certain actions or events.
- **Code management:** Technical decision-makers will need to determine which technologies and processes their teams will use for code management. This generally includes code repositories like GitHub where code can be saved, versioned, shared, and reused. It also includes tools for using and versioning code libraries, notebooks, and software development kits (SDKs).
- **Dataset management:** We also recommend saving training datasets centrally. This way, teams can reuse them, share them with colleagues, or monitor how they change over time in order to manage drift.
- **Shared environments:** Create model environments that can be shared among individuals. This simplifies the handoff between steps in the model creation process and makes it possible for teams to collaborate on certain steps.

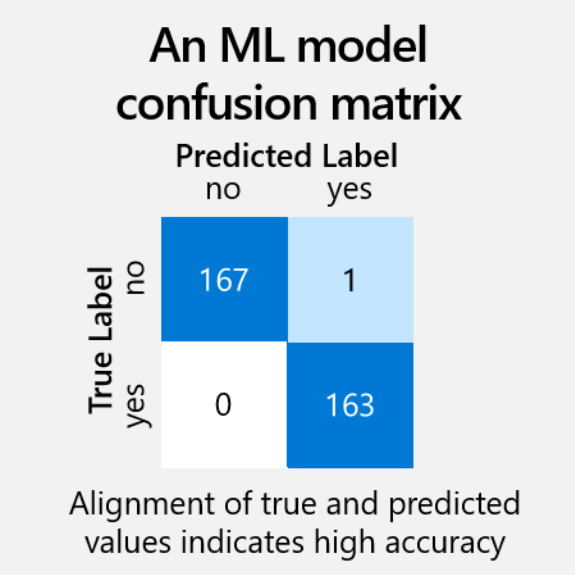
Second, we recommend creating **machine learning "pipelines."** Pipelines are independently executable workflows of complete machine learning tasks (such as data preparation, training configuration, training processes, and model validation). Having independent steps saved to a pipeline allows multiple data scientists to work on the same pipeline concurrently. Additionally, when data scientists need to go back and make changes to their work, they can start from where the change needs to occur instead of going back to the beginning. This helps them avoid running costly and time-intensive steps like data ingestion again if the underlying data hasn't changed.

Model validation

Before a model is deployed, it's critical to validate its performance metrics against the business use case. For example, perhaps you designed a model to predict patient health. As a healthcare provider dealing with life and death situations, you likely prefer to have false positive diagnoses rather than an incredibly high rate of accuracy that misses diagnoses.

You may have several metrics that are used to indicate the "best" model. It's important to work with data scientists to understand what metrics are important and evaluate them before deployment. There are **tools to evaluate model metrics**, such as a loss function or a confusion matrix.

If the model is a newer version of an existing model, you'll need to see if it performs better than the previous one on key metrics.



Model deployment

Model developers should work with the infrastructure or app developers to determine how to best deploy the model into production. One option is deploying models using the cloud (often leveraging an API). Scalable web infrastructure(s) like Kubernetes or Azure Container Instances are often used to automate and simplify this process. Models can also be deployed directly in on-prem servers or on edge devices like cameras, IoT gateways, and machinery.

No matter where you deploy the model, the workflow is similar. First, you'll register the model in the model registry. Then, you'll prepare to deploy the model by specifying assets, usage, and the compute target. Finally, you'll deploy it to your desired location, test it, and continue to monitor model-specific metrics throughout the lifecycle.

Model retraining

Although this is the end of the development process, this is just the beginning of the maintenance cycle. Models need to be monitored and periodically retrained to correct performance issues and take advantage of newer training data. To set yourself up for success, you'll want to create a retraining loop—or a systematic and iterative process to continually refine and ensure the accuracy of the model.

TransLink case study

TransLink, the transit agency for Vancouver, Canada, wanted to provide more accurate time estimates for bus departures. TransLink partnered with Microsoft and T4G to build 18,000 AI models that together automatically predict accurate departure times by considering factors like traffic, bad weather, and other schedule disruptions. TransLink succeeded in creating and managing this high volume of sophisticated models because they adopted MLOps strategies to:

- Automate model training and deployment processes through pipelines
- Create an approval process for automated model training results
- Integrate a data drift system into build-and-release pipelines so that retraining is triggered automatically if data drift is detected

The solution improved the accuracy of predicted departure times by 74% and reduced average customer wait times by 50%.