International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Supercharge AI model deployment using Databricks MLOps

Vamshi Krishna Malthummeda

mvamsikhyd@gmail.com

Abstract:
Rapid adoption of AI models necessitates robust source code management, better collaboration, faster and consistent model deployments, efficient model and data drift management, minimal oversight and accelerated time-to-market[1]. To address these challenges this paper proposes implementation of Databricks Machine Learning Operations (MLOps). It is a set of tools and methodologies designed to streamline the entire machine learning lifecycle within unified Lakehouse platform, from experimentation (facilitated by Databricks MLFlow Experiments component) and development (Databricks Notebooks are leveraged for this purpose) to deployment (GitHub Actions provides robust platform for implementing CI/CD pipelines), monitoring and maintenance (Lakehouse monitoring is designed to monitor the quality and performance) of ML models. The implementation of Databricks MLOps resulted in significant reduction of the time required to bring models to production, lead to faster iteration cycles and higher quality models, ensured optimal performance and mitigated the risk of model performance degradation. Databricks MLOps helps the organizations to get high ROI with faster time-to-business value AI powered solutions, effective resource utilization and reduced operational costs.

Keywords: Databricks, MLFlow, Github, Lakehouse, Model Serving, Machine Learning, Artificial Intelligence, CI/CD, pipeline.

INTRODUCTION

In retail industry, as customer behavior, market conditions and product trends evolve the product recommendation, pricing and inventory management models should adapt quickly and should continuously detect the changes to remain accurate for making informed merchandising decisions. Databricks MLOps will address this challenge of **data drift/ model drift** along with other challenges like **reproducibility** and **governance.**

Data drift:
Data drift refers to changes in the statistical properties of the input data over time. This means the distribution of the features used by the model shifts from what it was during training.

Model drift:
It is also known as model decay, signifies a decline in the performance of a machine learning model over time. This degradation occurs because the model's learned patterns and relationships no longer accurately reflect the current reality[2].

Reproducibility:
It is a practice of being able to re-create an ML model's exact results by tracking and versioning all components of its development and deployment, including the code, data, parameters, and environment.

Governance:
It is a set of systematic processes and controls (like versioning, auditing, and access management) to ensure models are developed, deployed, and managed reliably, transparently, and compliantly throughout their lifecycle.

DATABRICKS PROVIDES BELOW FEATURES TO ADDRESS THE ABOVE-MENTIONED CHALLENGES:

GitHub:

GitHub plays a crucial role in Databricks MLOps by providing robust version control, facilitating collaborative development, and enabling automated CI/CD pipelines for machine learning projects.

- **Version Control and Collaboration:** GitHub serves as the central hub for managing all ML project code, including notebooks, scripts, and configuration files. It tracks every change, allowing teams to roll back to previous versions, if necessary, which ensures data integrity and reproducibility. Features like pull requests and code reviews enable data scientists and engineers to collaborate effectively, ensuring high code quality and adherence to best practices.
- **CI/CD and Automation:** Databricks integrates seamlessly with GitHub Actions to automate various MLOps tasks. Automated workflows can be triggered by changes in the repository to handle model training, testing, deployment, and monitoring. Databricks MLOps Stacks leverage GitHub for CI/CD by defining resources and pipelines as code within Databricks Asset Bundles, which ensures consistency and version control for the entire ML environment.

MLflow:

Databricks Managed MLflow is a fully integrated and managed version of the open-source MLflow platform, designed for managing the machine learning lifecycle on the Databricks Lakehouse Platform. It provides a centralized system for tracking experiments, logging parameters, metrics, artifacts, model registry, managing models, and deploying machine learning AI applications.

Delta Lake:

ACID Transaction feature of Delta Lake ensures data integrity and consistency, preventing data corruption and making pipelines more reliable. Provides storage with versioning of the tables, which is helpful for maintaining baselines and historical data. Schema evolution feature allows for controlled updates to schemas without disrupting existing models. Time travel feature provides the ability to access and reproduce the historical versions of the data.

Feature Store:

The Databricks Feature Store provides a central registry for features used in your AI and ML models. Feature tables and models are registered in Unity Catalog, providing built-in governance, lineage, and cross-workspace feature sharing and discovery.

Lakehouse Monitoring:

It is a data-centric monitoring solution to ensure that both data and AI assets are of high quality, accurate and reliable. Built on top of Unity Catalog, it provides the unique ability to implement both data and model monitoring while maintaining lineage between the data and AI assets of an MLOps solution. Monitors data tables and inference tables (inputs, predictions, optionally ground truth) for data quality, data drift and model performance drift[3].

Model Serving:

Databricks Model Serving provides a production-ready, serverless solution to simplify real-time ML model deployment. With Model Serving, it is possible to efficiently deploy models as an API so that you can integrate model predictions with applications or websites. Given the complexity that is often involved with deploying a real-time ML model, Model Serving reduces operational costs, streamlines the ML lifecycle, and makes it easier for data science teams to focus on the core task of integrating production-grade real-time ML into their solutions[4].

ROLE OF UNITY CATALOG

Unity Catalog plays a central and critical role in Databricks MLOps by providing a unified governance solution for all data and AI assets. This integration enhances the management, security, and reproducibility of machine learning workflows.

Unified Governance:
Unity Catalog enables consistent governance and security policies across both data and models. By bringing these assets together in a single unified platform, it streamlines management, enhances security, and simplifies the overall MLOps process.

Read only access to assets in production environment:
With Unity Catalog, data and models are governed at the account level, promoting easy sharing of assets across Databricks workspaces. Data scientists can train models directly on production data, identify and troubleshoot model quality issues through production monitoring tables, analyze predictions using inference tables, and seamlessly compare models in development with those of production. This approach not only speeds up model development but also enhances the robustness and overall quality of the models produced.

Model Registry in Unity Catalog:
Models will be registered under a three-level name in the form <catalog>. <schema>. <model>. This 3 level name will allow you to apply governance at each level of the hierarchy. A mutable named reference or alias can be associated with a particular version of a registered model. The inference traffic can be targeted to that particular named reference. Ex: A "Champion" alias is assigned to a model version which is used for inference traffic.

Lineage:
With Unity Catalog, a robust link between data and AI assets can natively be recorded. Lineage can be traced from a model version in Unity Catalog back to the data used for training.

Discoverability:
Through centralizing data and AI assets in a single solution, Unity Catalog enhances the discoverability of these assets, making it simpler and quicker to locate and utilize the appropriate resources for a particular component of the MLOps solution. For example, teammates with access to view each other's models can see which data sources are being used, and use that information to address their own use cases.

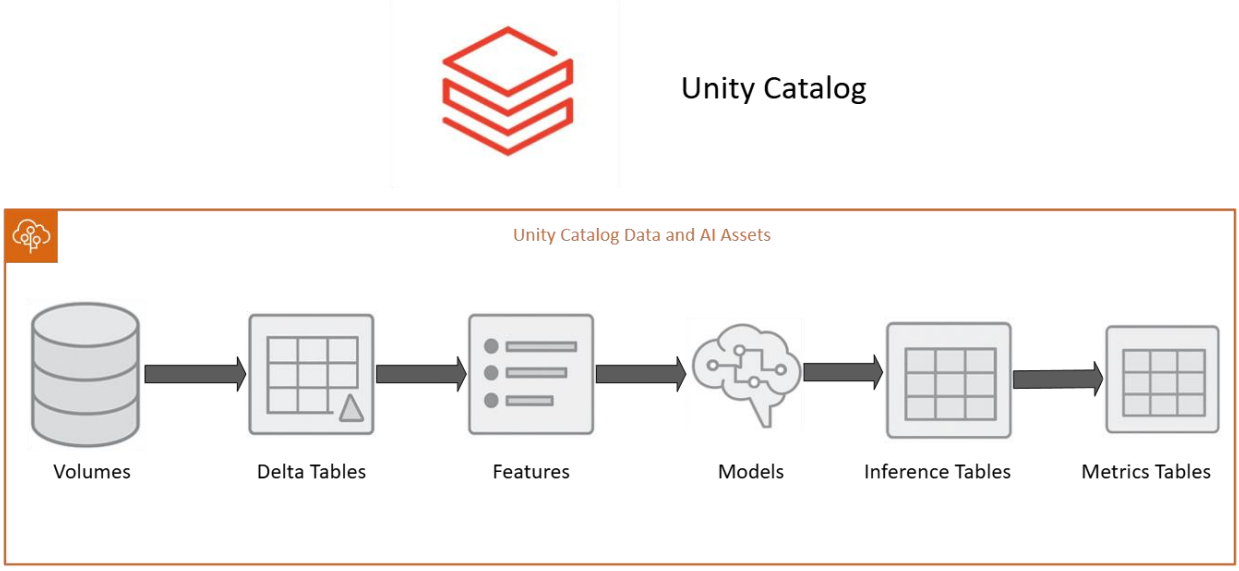

Figure 1: Unity Catalog Data & AI Assets

DATABRICKS MLOPS CI/CD PIPELINE

It is an automated workflow that helps you build, test, deploy, and monitor machine-learning models on Databricks. This pipeline integrates continuous integration (CI) and continuous delivery/deployment (CD) principles to streamline the ML lifecycle[5].

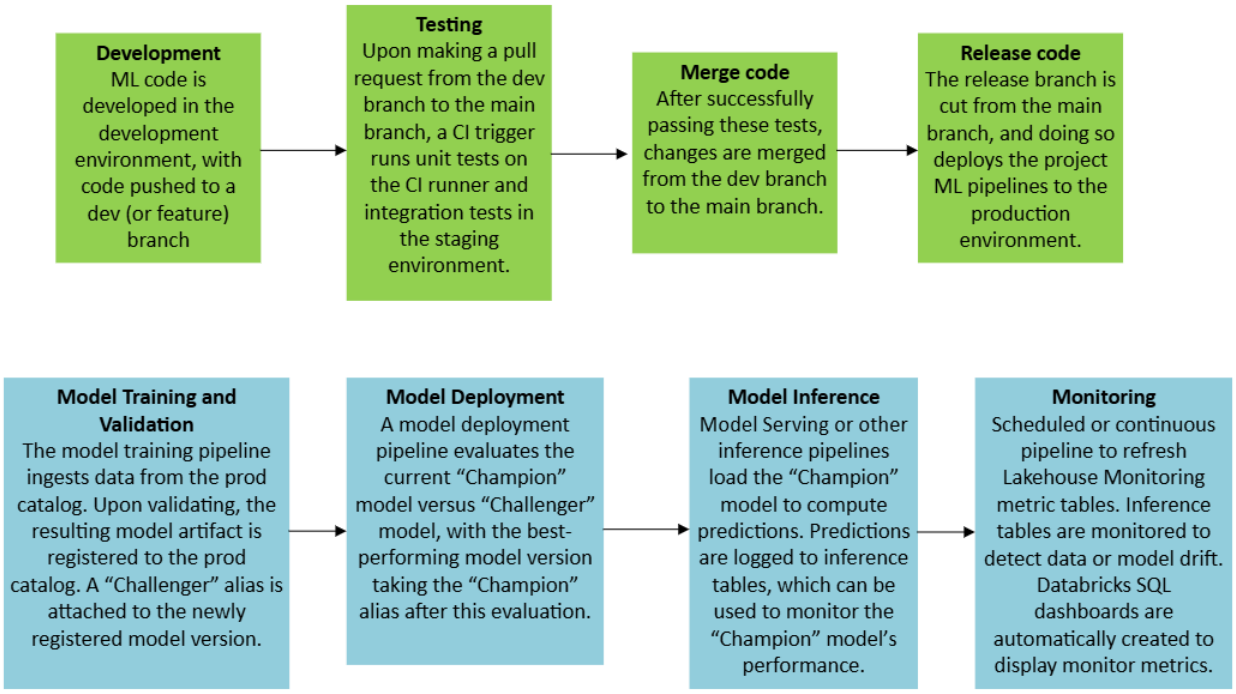

Figure 2: Code and Model life cycle stages

MACHINE LEARNING MODELS DEPLOYMENT STRATEGIES

There will be a situation where the model gets updated in production without any changes being made to the code. Hence, the machine learning models and their associated code progress asynchronously through development, staging and production stages. Just like models and code, data also will be labeled as development, staging and production grade data.

Below is the life cycle of ML code:

1. After being developed, ML project code moves from the development environment to the staging environment for testing. Once it passes all tests, the code is deployed to the production environment to run.
2. The model training code is first tested using a data subset in the **staging environment**. Following this, the complete model training pipeline is executed in the **production environment**.
3. The validation of a new model and its comparison to the current production model are both performed within the production environment during deployment.

Below is the lifecycle of ML Model:

1. A model is **trained** in the development environment. The resulting model artifact is then moved to the staging environment for **validation checks** before being deployed to the production environment.
2. This method necessitates two distinct deployment paths: one for the model itself and another for supporting code like inference and monitoring scripts. The code for these operational pipelines must be tested in a staging environment before being deployed to production.

CONCLUSION

As organizations increasingly rely on machine learning to drive business outcomes, MLOps has become essential for ensuring models remain accurate, scalable, and reliable in production. Data drift and model drift are inevitable parts of production ML. Using the Databricks capabilities like Delta Lake, Feature Store, MLflow and Lakehouse Monitoring the organizations can build pipelines that track, detect and respond to drift in a systematic way. Future work will refine explainability, automated drift handling and methods for sparse or delayed label settings. By adopting Databricks MLOps, companies can accelerate innovation, reduce operational friction, and maximize the value of their machine learning investments.

REFERENCES:

1. Siltala, V. (2023). MACHINE LEARNING OPERATIONS ARCHITECTURE IN HEALTHCARE BIG DATA ENVIRONMENT.
2. Bayram, F., Ahmed, B. S., & Kassler, A. (2022). From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems*, *245*, 108632.
3. Databricks Lakehouse Monitoring: Introduction to Databricks Lakehouse Monitoring | Databricks on AWS
4. Databricks Model Serving: Deploy models using Mosaic AI Model Serving | Databricks on AWS
5. Parvathinathan, K. Building CI/CD Pipelines for Machine Learning with GitOps Principles.