MLOps – CI/CD

DVC, MLFlow and Docker

Course Plan

MLOps Pipeline Overview

Version Control

CI/CD

Experiment Tracking

Model Packaging and Deployment

Model Monitoring and Alerting

Model Serving

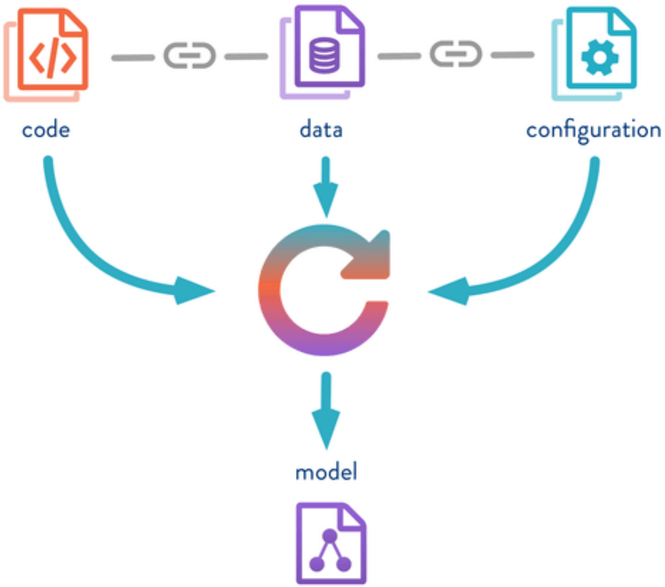
Automated Model Retraining

DVC

A game-changer for managing data and model versions in ML projects

What is DVC?

- DVC (Data Version Control) is an open-source version control system for managing data science and machine learning projects.
- It extends version control systems like Git to handle large data files, models, and pipelines, enabling more efficient and reproducible data science workflows.



Why DVC Matters?

- Traditional approaches to data management are prone to errors and inconsistencies.
- Manually tracking data versions leads to confusion and slows down progress.
- DVC solves these problems by providing a centralized location for managing all your data and model versions.
- Integrates closely with Git, allowing data scientists to leverage existing version control tools and workflows for their data and models, effectively treating data and code as equally important assets in a project.

DVC: Git for Data

- Purpose Comparison:
    - Git: Designed for source code version control, tracking changes in files and directories over time.
    - DVC: Extends Git's versioning capabilities to large data files and machine learning models, enabling data version control.
- Data Handling:
    - Git: Efficient with text files but struggles with large binary files.
    - DVC: Optimized for large data files and binary assets, stores data in remote storage while using Git for metadata.

DVC: Git for Data

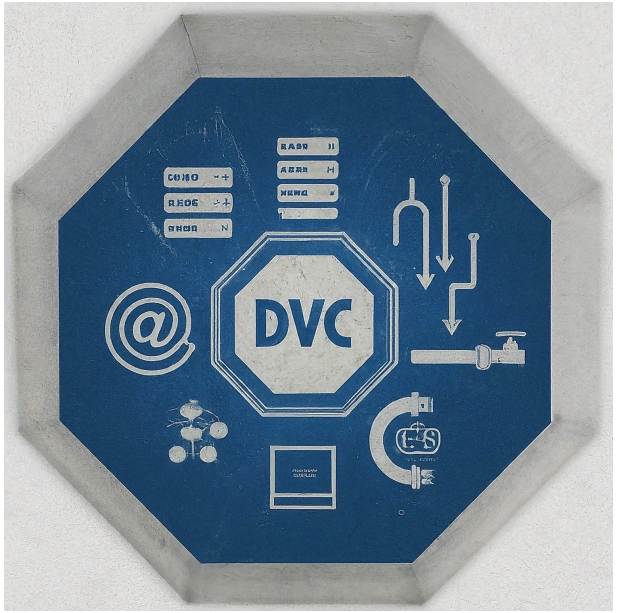
- Versioning Approach:
  - Git: Tracks changes by saving snapshots of the project directory.
  - DVC: Tracks data and model versions using pointers in Git to manage large files stored externally.
- Collaboration Features:
  - Git: Branching and merging facilitate collaborative code development.
  - DVC: Supports collaborative data science workflows by versioning data and models, allowing teams to experiment independently and merge changes.

DVC: Git for Data

- Workflow Integration:
  - Git: Primarily focused on code changes, pull requests, and code review processes.
  - DVC: Integrates with ML pipelines, enabling versioning of the entire ML workflow, including data preprocessing, training, and evaluation stages.
- Use Case Suitability:
  - Git: Ideal for projects where source code is the primary artifact.
  - DVC: Best suited for projects where data and models need to be versioned alongside code.

DVC Fundamentals

- **Data Storage:** DVC stores large datasets in efficient remote storage (e.g., S3, GCS) while keeping metadata (file references) in Git.
- **Versioning:** DVC tracks versions of data and models, enabling you to see how they've changed over time.
- **Pipelines:** DVC pipelines define multi-stage workflows for data processing, training, and model evaluation.
    - This promotes modularity, automation, and easier collaboration by sharing defined workflows.



Data Storage with DVC

- Remote Storage Integration:
  - DVC integrates with a variety of remote storage solutions, including Amazon S3, Google Cloud Storage (GCS), Microsoft Azure Blob Storage, and others.
  - This flexibility allows teams to store large datasets and models externally, reducing the load on source code repositories.
- Data Tracking:
  - DVC tracks data files and directories by creating lightweight metafiles.
  - These metafiles are stored in Git, allowing users to version control their data without storing the actual data in Git.
  - This approach keeps the Git repository clean and manageable.

Data Storage with DVC

- Efficient Data Transfer:
  - DVC optimizes data transfer to and from remote storage.
  - It only uploads or downloads changes, similar to how Git handles code, making data management more efficient.
- Linking Data to Code:
  - By storing metadata in Git, DVC links specific data versions directly to corresponding code versions.
  - This ensures that every project state is reproducible, as both code and data can be checked out to a specific version.

Data Storage with DVC

- Data Caching:
  - DVC uses caching to prevent duplication of data.
  - If multiple projects use the same dataset, DVC stores a single copy in the cache, saving disk space and speeding up data retrieval.
- Access Control and Security:
  - When using cloud storage services (S3, GCS, etc.), data security and access control are managed through the storage provider, leveraging existing permissions and security mechanisms to protect sensitive data.

Versioning Data and Models

- Efficient Version Control:
  - DVC extends Git capabilities to handle large files, datasets, and machine learning models, enabling efficient version control without bloating the Git repository.
- Snapshot Creation:
  - With DVC, every change in data or models can be tracked, and snapshots of the entire project can be created at any point in time.
  - This includes both the data files and the corresponding metadata.
- Reproducibility:
  - DVC ensures that every version of the data and models can be precisely matched with the code that processed or generated them, making experiments fully reproducible.

Versioning Data and Models

- Collaboration and Sharing:
  - By linking data and model versions to specific stages in the pipeline, DVC facilitates seamless collaboration among team members.
  - It ensures that everyone is working with the correct version of data and models.
- Workspace Management:
  - DVC allows for easy switching between different versions of data and models, making it simpler to test new ideas or hypotheses without disrupting the current workflow.
- Experiment Tracking and Comparison:
  - DVC provides tools to compare different experiments side-by-side, making it easier to evaluate the impact of changes and select the best performing models.

DVC Pipelines for Workflow Management

- DVC Pipelines:
  - Automates and manages complex machine learning workflows.
  - Defines stages in a pipeline through simple YAML files.
- Defining Multi-Stage Workflows:
  - Each stage can represent a step in the ML model lifecycle, such as data preprocessing, training, and evaluation.
  - Stages are connected, allowing output from one stage to serve as input to another.

DVC Pipelines for Workflow Management

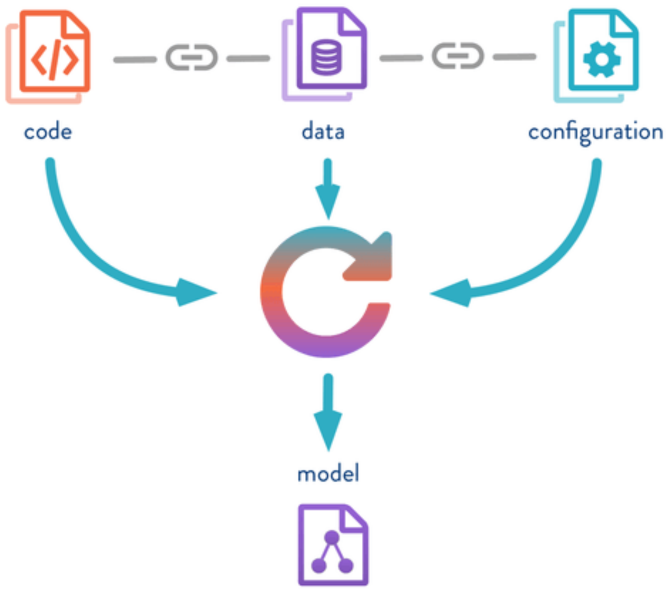
- Benefits of Using DVC Pipelines:
  - **Reproducibility:** Ensures experiments can be precisely replicated by tracking the exact data and parameters used at each stage.
  - **Collaboration:** Simplifies sharing of workflows and results with team members, enhancing collaboration.
  - **Efficiency:** Reduces the time and effort needed to recreate experiments, facilitating a more efficient research and development process.
- Pipeline Execution and Tracking:
  - Execute entire pipelines or individual stages with a single command.
  - Automatically tracks changes to each stage, ensuring that only the necessary parts of the pipeline are rerun when modifications are made.

DVC Pipelines for Workflow Management

- Visualization and Management:
  - Visualize pipeline structure and dependencies with DVC commands, aiding in understanding and optimizing workflows.
  - Manage and switch between different versions of data and models, allowing for easy experimentation with various configurations.
- Integration with Version Control:
  - Integrates seamlessly with Git, storing pipeline definitions and changes in version control, alongside code.
  - Facilitates the versioning of both data/models and their processing stages, providing a comprehensive version control system for machine learning projects.

DVC Pipelines for Workflow Management

- https://dvc.org/doc/start
- https://dvc.org/doc/command-reference



MLFlow

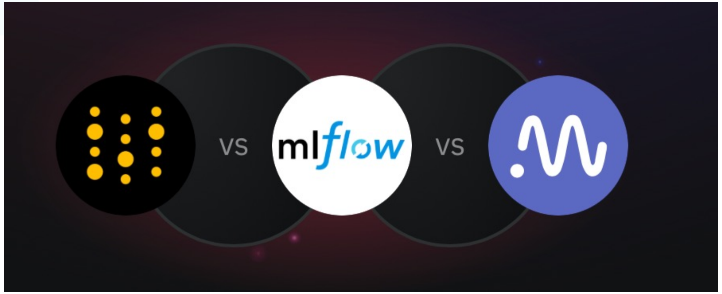
Managing the Machine Learning Lifecycle

Introduction to MLflow

- Purpose of MLflow in the Machine Learning Lifecycle
  - Simplifies the complex process of machine learning model development and deployment.
  - Provides a unified platform to manage the end-to-end machine learning lifecycle.
  - Enhances productivity and fosters collaboration among data scientists and engineers.
- Overview of Four Main Components
  - Tracking: Log and compare parameters, metrics, and artifacts from different runs.
  - Projects: Package ML code in a reusable and reproducible form to share with others.
  - Models: Standardize the format for packaging models to simplify deployment across various platforms.
  - Registry: Centralize model storage to manage versions and lifecycle stages (staging, production).

Introduction to MLflow

- Importance of Experiment Tracking, Model Packaging, and Versioning
  - Experiment Tracking: Essential for understanding model performance and iterating over models efficiently.
  - Model Packaging: Enables consistent model deployment across different environments, reducing operational complexities.
  - Versioning: Critical for managing model iterations, ensuring reproducibility, and facilitating smooth rollouts to production.
  - Other tools: Weights & Biases, Neptune



MLflow Components

- Introduction to MLflow Components
  - MLflow offers a unified platform to manage the end-to-end machine learning lifecycle, simplifying the process of building, training, and deploying models.
- MLflow Tracking
  - Purpose: Logs and tracks experiments, including code, data, config, and results.
- Key Features:
  - Tracks parameters, metrics, and outputs to compare across runs.
  - Facilitates experiment reproducibility and collaboration.

MLflow Components

- MLflow Projects
  - Purpose: Packages ML code in a reusable and reproducible format for any ML library.
  - Key Features:
    - Uses standard formats for environment specification (Conda, Docker).
    - Enables easy sharing and collaboration on ML projects.
- MLflow Models
  - Purpose: Standardizes the format for ML model packaging across diverse ML libraries.
  - Key Features:
    - Supports multiple model flavors for flexibility.
    - Simplifies deployment to various production environments (e.g., cloud services, local servers).

MLflow Components

- MLflow Model Registry
  - Purpose: Manages model lifecycle stages: from development to staging and production.
  - Key Features:
    - Tracks model versions and metadata.
    - Facilitates model review, approval, and rollback processes.
- Integration of Components
  - Seamless Workflow: MLflow's components work together to streamline the ML lifecycle from experimentation to production.
    - Tracking → Projects: Experiment tracking feeds into project packaging by capturing all necessary components to reproduce runs.
    - Projects → Models: Packaged projects facilitate model creation, standardization, and preparation for deployment.
    - Models → Registry: Deployed models are versioned and managed through the Model Registry, enabling safe rollout and monitoring in production environments.

MLflow Components

- Summary
  - MLflow provides an integrated platform that addresses key challenges in machine learning development and deployment, ensuring efficiency, reproducibility, and scalability across projects.

Docker

Docker

- What is Docker?
  - A platform for developing, shipping, and running applications.
  - Utilizes containerization to make applications portable and consistent across different environments.
  - Containers encapsulate an application with all of its dependencies.

Docker

- Relevance in Modern Software Development
  - Facilitates continuous integration and continuous deployment (CI/CD) by ensuring that software runs the same in all environments.
  - Reduces "it works on my machine" problems by providing a consistent environment from development to production.
  - Enables microservices architecture by allowing each service to be containerized and scaled independently.
  - Streamlines development by allowing developers to create predictable and efficient work environments.
  - Enhances collaboration between development and operations teams for faster and more reliable software delivery.

Docker: Core Concepts

- Containers vs. Virtual Machines
  - Containers are an abstraction at the app layer that packages code and dependencies together.
  - Virtual Machines (VMs) are an abstraction of physical hardware, turning one server into many servers.
  - Containers share the host system's kernel with other containers, lightweight, and require less start-up time.
  - VMs include the application, the necessary binaries and libraries, and an entire guest operating system – all of which can be tens of GBs.

Docker: Core Concepts

- Images and Containers
  - Docker images are lightweight, stand-alone, executable software packages that include everything needed to run a piece of software, including the code, runtime, libraries, environment variables, and config files.
  - Containers are a runtime instance of Docker images – an image becomes a container when it runs on Docker Engine.
  - Containers are isolated from each other and the host system, but can communicate through well-defined channels.

Docker: Core Concepts

- Docker Hub
  - Docker Hub is a cloud-based registry service that allows you to link to code repositories, build your images, test them, store manually pushed images, and link to Docker Cloud.
  - Provides a comprehensive repository for Docker container images with both public and private storage options.
  - It is the default registry where Docker looks for images.
  - Offers automated build capabilities for creating Docker images from online source code repositories.

Docker

- Docker's Benefits
  - Promotes consistency across development, staging, and production environments.
  - Ensures that applications run the same, regardless of where they are deployed.
  - Reduces overhead and increases efficiency compared to traditional virtual machines.

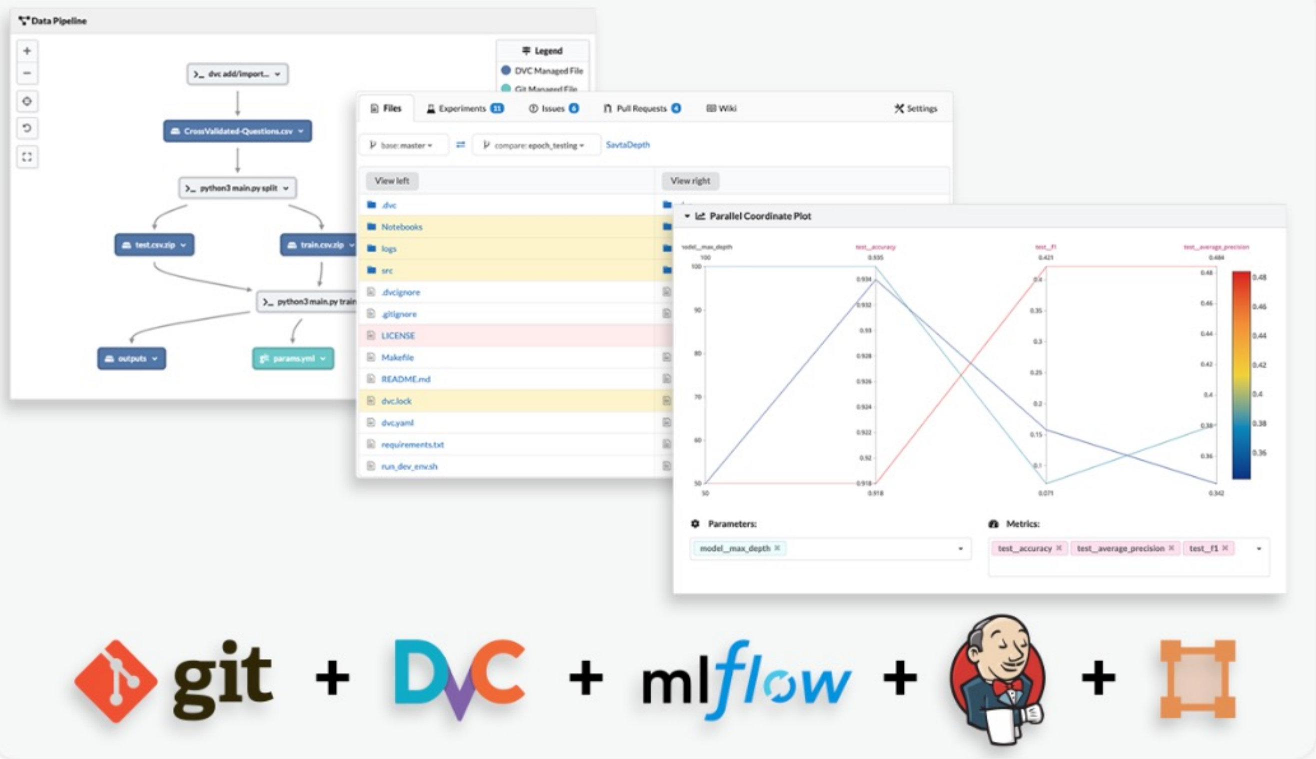
Docker

- Docker in the Development Toolchain
  - Integrates with GitHub for version control, enabling seamless code updates and tracking.
  - Works alongside GitHub Actions for automated testing and deployment pipelines.
  - Complements AWS Compute services to provide scalable and secure hosting for containerized applications.
  - Facilitates data version control with DVC, making it easier to track and manage datasets and machine learning models.
  - Enhances ML operations by working with MLflow to manage the lifecycle of machine learning models, including experimentation, reproducibility, and deployment.

Docker

- Synergy Between Technologies
  - Containerization with Docker offers a standardized unit for software development, enabling a smooth workflow across tools like GitHub, AWS, DVC, and MLflow.
  - Simplifies the complexity of managing dependencies and environments in machine learning projects.
  - Empowers teams to build, test, and release software faster and more reliably, contributing to the DevOps culture of collaboration and efficiency.

Docker



Data Pipeline
Legend
DVC Managed File
dvc add/import...
CrossValidated-Questions.csv
python3 main.py split
test.csv.zip
train.csv.zip
outputs
params.yml
Files
Experiments
Issues
Pull Requests
Wiki
Settings
View left
View right
.dvc
Notebooks
logs
src
.dvcignore
.gitignore
LICENSE
Makefile
README.md
dvc.lock
dvc.yaml
requirements.txt
run_dev_env.sh
Parallel Coordinate Plot
Parameters:
model_max_depth
Metrics:
test_accuracy
test_average_precision
test_f1
git + DVC + mlflow + +

DagsHub

Introduction to DagsHub

- What is DagsHub?
  - DagsHub is a collaboration platform designed specifically for Data Science and Machine Learning projects.
  - It integrates features for version control, data management, and experiment tracking into a single, user-friendly interface.
- Purpose of DagsHub
  - To streamline the collaboration process for data scientists and ML engineers.
  - To address common challenges in data science projects, such as managing data versions, tracking experiments, and sharing results with team members.

Introduction to DagsHub

- Why DagsHub?
  - Traditional version control systems are not optimized for data science workflows.
  - DagsHub fills this gap by providing tools that are tailored to the needs of data science teams, enabling more efficient and effective collaboration.
- The Role of DagsHub in the Data Science Community
  - Acts as a central hub for data science projects, promoting open-source collaboration and sharing of knowledge.
  - Connects data scientists, researchers, and developers, facilitating a community-driven approach to solving complex data problems.

Introduction to DagsHub

- Key Features (Preview)
  - Data and code version control, experiment tracking, model management, and project collaboration features.
  - Compatibility with popular data science tools and platforms for seamless integration into existing workflows.

The Need for Collaboration in Data Science

- Complexity of Data Science Projects
  - Multidisciplinary teams combining expertise in data engineering, data analysis, and machine learning.
  - Projects often involve large, complex datasets and sophisticated algorithms.
- Challenges in Reproducibility
  - Difficulty in replicating results due to differences in data, environment, or code.
  - The need for rigorous version control of both data and code.

The Need for Collaboration in Data Science

- Experiment Tracking and Management
  - Hundreds of experiments with different parameters, data versions, and outcomes.
  - The necessity of a systematic approach to log, compare, and retrieve experiment results.
- Efficient Collaboration and Knowledge Sharing
  - Seamless sharing of data, models, and experiments within and across teams.
  - Reducing duplication of effort and leveraging collective knowledge for faster innovation.

The Need for Collaboration in Data Science

- Security and Access Control
  - Managing permissions and access to sensitive data and proprietary algorithms.
  - Ensuring compliance with data protection regulations while collaborating.
- The Role of DagsHub
  - DagsHub addresses these challenges by providing a platform specifically designed for data science and ML collaboration.
  - Integrates data version control, experiment tracking, and collaborative features in one interface.

Overview of DagsHub Features

- Data Version Control:
  - Enables version control for datasets, similar to Git for code, ensuring data changes are trackable and reversible.
  - Integrates with DVC (Data Version Control) for handling large datasets and binary files efficiently.
- Experiment Tracking:
  - Facilitates the logging and comparison of ML experiments with detailed metrics and parameters.
  - Supports visualizing changes between experiments, helping identify the best performing models.

Overview of DagsHub Features

- ML-Specific Project Management Tools:
  - Offers project boards tailored for ML projects to track progress, manage tasks, and collaborate on data science workflows.
  - Provides an integrated environment for ML projects, combining code, data, and experiments in one platform.
- Collaborative Notebooks:
  - Enables sharing and collaborating on Jupyter Notebooks directly within the platform, promoting interactive development and analysis.

Overview of DagsHub Features

- Model Registry:
  - A centralized hub for managing and versioning ML models, making it easier to store, share, and deploy models.
- Integrated Code and Data Review:
  - Supports pull requests not just for code, but also for datasets and models, facilitating comprehensive review processes.
- Open and Reproducible Science:
  - Promotes open science by making projects easily shareable and reproducible, with public and private project support.