Pre-reading: Experiment Tracking and Data Versioning with MLflow and DVC

Introduction

In machine learning projects, keeping track of experiments and managing data versions are crucial for reproducibility and collaboration. Without proper tracking, it becomes difficult to compare model performances, revisit previous work, or reproduce results accurately. Two powerful tools that help achieve this are MLflow (for experiment tracking) and DVC (for data versioning).

Why Use MLflow and DVC?

Challenges in ML Workflow:

- **Tracking Experiments**: Running multiple experiments with different parameters can become overwhelming. Without an automated tracking system, managing results manually is inefficient and error-prone.
- **Comparing Model Performance**: Without a systematic way to track model metrics, evaluating different models can become challenging.
- **Managing Large Datasets**: Storing, updating, and sharing large datasets efficiently is a major hurdle.
- **Ensuring Reproducibility**: Without version control for code, data, and models, recreating previous results is difficult.

How MLflow and DVC Solve These Issues:

- **MLflow** automates the logging of experiments, tracking parameters, metrics, artifacts, and models. This reduces manual effort and ensures organized experiment tracking, making it easy to compare different runs and maintain records.
- **DVC** enables version control for datasets and model outputs, integrating seamlessly with Git, ensuring data consistency across different versions.

MLflow: Experiment Tracking

MLflow provides a structured way to track machine learning experiments. It consists of the following components:

1. **MLflow Tracking**: Records parameters, metrics, and artifacts for each experiment.
2. **MLflow Projects**: Defines reusable, reproducible ML projects.
3. **MLflow Models**: Standardizes model packaging for deployment.
4. **MLflow Registry**: Manages and deploys different versions of models.

Problem Statement:

Imagine you are training a linear regression model to predict housing prices based on different features. You want to track multiple experiments with different hyperparameters and compare their performance. MLflow allows you to log each experiment run, store the model, and visualize results efficiently.

Code Example:

```
import mlflow

import mlflow.sklearn

from sklearn.linear_model import LinearRegression

from sklearn.datasets import make_regression

from sklearn.model_selection import train_test_split



# Generate synthetic data for regression

X, y = make_regression(n_samples=100, n_features=1, noise=0.1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)



# Start an MLflow run

with mlflow.start_run():

    model = LinearRegression()

    model.fit(X_train, y_train)  # Train the model
    score = model.score(X_test, y_test)  # Evaluate model performance


    # Log hyperparameters and performance metrics

    mlflow.log_param("alpha", 0.1)

    mlflow.log_metric("r2_score", score)


    # Log the trained model

    mlflow.sklearn.log_model(model, "model")


    print(f"Model logged with R2 score: {score}")
```

Explanation:

- **`mlflow.start_run()`** initializes an experiment run.
- The model is trained and its R² score is calculated.
- **`mlflow.log_param()`** logs hyperparameters.
- **`mlflow.log_metric()`** logs performance metrics.
- **`mlflow.sklearn.log_model()`** stores the trained model for future retrieval.

DVC: Data Versioning

DVC (Data Version Control) is a Git-like system designed specifically for managing large datasets and model artifacts. Unlike Git, which is inefficient for storing large files, DVC enables tracking of large files while storing only metadata in Git.

Problem Statement:

You are working on a machine learning project with a large dataset (`dataset.csv`) that needs version control. Using DVC, you can track changes in data, collaborate with teammates, and ensure consistency between experiments.

Code Example:

```
# Initialize DVC in a project
$ dvc init

# Add data to DVC for version control
$ dvc add dataset.csv

# Track data using Git
$ git add dataset.csv.dvc .gitignore
$ git commit -m "Versioned dataset with DVC"

# Push data to remote storage for sharing
$ dvc remote add myremote s3://mybucket/path
$ dvc push
```

Explanation:

- **`dvc`** `init` sets up DVC in a repository.
- **`dvc add dataset.csv`** versions the dataset.
- **`git add`** and **`git commit`** track changes using Git.
- **`dvc push`** uploads the dataset to cloud storage, allowing team collaboration

Conclusion

Using **MLflow for experiment tracking** and **DVC for data versioning** streamlines ML workflows, ensuring reproducibility, efficiency, and better collaboration. These tools integrate well with Git, making them essential for ML practitioners.

Key Takeaways:

- **MLflow** helps track experiments, log model performance, and compare results.
- **DVC** provides an efficient way to manage and version large datasets.
- Both tools improve **reproducibility**, **collaboration**, and **organization** in ML projects.

In the upcoming demo session, we will explore these tools in action, setting up MLflow and DVC step by step.