**Streamlining MLOps Workflows with DVC, Airflow, and MLFlow**

**Introduction**

In the rapidly evolving field of machine learning, managing workflows and ensuring reproducibility can be challenging. This project explores how tools like **DVC (Data Version Control)**, **Apache Airflow**, and **MLFlow** simplify and automate MLOps workflows. By integrating these tools, we aimed to build an end-to-end pipeline that handles data collection, preprocessing, training, and deployment, ensuring traceability and efficiency.

**Project Overview**

**Workflow Goals:**

* Automate data collection and preprocessing.
* Version data and models using DVC.
* Implement a robust pipeline with Apache Airflow.
* Use MLFlow for model tracking and versioning.

**Tools Used:**

* **DVC**: For tracking datasets and ensuring version control.
* **Airflow**: For orchestrating tasks in an automated pipeline.
* **MLFlow**: For model tracking and performance monitoring.
* **Docker and Minikube**: For containerization and deploying Kubernetes clusters.

The project aimed to create a repeatable and scalable MLOps pipeline that can easily handle new data and retraining.

**Data Collection and Preprocessing**

**Data Collection:**

We collected weather data using the OpenWeatherMap API, gathering key fields such as:

* Temperature
* Humidity
* Wind Speed
* Weather Condition

Here’s a code snippet for the data collection script:

python

Copy code

import requests

import pandas as pd

from datetime import datetime

API\_KEY = "your\_api\_key"

URL = f"https://api.openweathermap.org/data/2.5/forecast?q=city&appid={API\_KEY}"

response = requests.get(URL)

data = response.json()

# Parse and save data

df = pd.DataFrame([{

"Date": datetime.utcfromtimestamp(entry["dt"]).strftime('%Y-%m-%d'),

"Temperature": entry["main"]["temp"],

"Humidity": entry["main"]["humidity"],

"Wind Speed": entry["wind"]["speed"]

} for entry in data["list"]])

df.to\_csv("raw\_data.csv", index=False)

**Data Versioning:**

Using DVC, we tracked the raw\_data.csv and ensured reproducibility:

bash

Copy code

dvc init

dvc add raw\_data.csv

git commit -m "Add raw data"

**Workflow Automation with Airflow**

Airflow orchestrated the data collection and preprocessing tasks. The pipeline included:

1. **Data Collection Task**: Periodically fetched weather data.
2. **Data Preprocessing Task**: Cleaned, normalized, and saved data for model training.

Here’s a snippet of the Airflow DAG:

python

Copy code

from airflow import DAG

from airflow.operators.python import PythonOperator

from datetime import datetime

def collect\_data():

# Data collection logic

pass

def preprocess\_data():

# Data preprocessing logic

pass

with DAG("weather\_pipeline", start\_date=datetime(2023, 1, 1), schedule\_interval="@daily") as dag:

task1 = PythonOperator(task\_id="collect\_data", python\_callable=collect\_data)

task2 = PythonOperator(task\_id="preprocess\_data", python\_callable=preprocess\_data)

task1 >> task2

**Diagram**:

**Model Training and Monitoring**

**Model Training:**

A simple **linear regression model** was trained to predict temperature based on humidity and wind speed. The training pipeline was implemented as part of the Airflow DAG.

**DVC for Model Versioning:**

Each model version was tracked using DVC:

bash

Copy code

dvc add model.pkl

git commit -m "Add trained model"

**MLFlow Integration:**

MLFlow tracked model performance metrics and allowed version comparison:

python

Copy code

import mlflow

mlflow.log\_param("algorithm", "linear regression")

mlflow.log\_metric("mse", mean\_squared\_error(y\_test, y\_pred))

mlflow.sklearn.log\_model(model, "linear\_regression\_model")

**Key Learnings**

1. **DVC for Reproducibility**: Ensures datasets and models are version-controlled, making experiments reproducible.
2. **Airflow for Workflow Automation**: Simplifies complex workflows and provides a clear task dependency visualization.
3. **MLFlow for Tracking**: Streamlines model versioning and performance monitoring.

These tools together create a robust MLOps workflow that reduces manual overhead and ensures scalability.

**Conclusion**

Integrating DVC, Airflow, and MLFlow demonstrated the power of adopting MLOps practices. This project highlighted the importance of:

* **Reproducibility** in machine learning workflows.
* **Automation** for efficiency and error reduction.
* **Versioning** to track and compare datasets and models.

By adopting these practices, we can build scalable, production-ready pipelines that are resilient to new data and evolving requirements.