Assignment No: 04

Name: RUTUJA TIKHOLE

PRN: 22310034

Roll NO: 381004

Problem Statement:

Track experiments using MLflow or Weights & Biases.

Theory:

1. Introduction

In machine learning projects, data preprocessing and model training often involve multiple steps such as data extraction, cleaning, feature engineering, training, evaluation, and deployment. Manually running these steps is error-prone, time-consuming, and difficult to scale. Workflow orchestration tools help automate and schedule these processes, ensuring reproducibility, reliability, and collaboration.

Two popular orchestration tools for ML pipelines are:

- Apache Airflow (open-source, widely used for complex workflows and scheduling)
- Prefect (modern, Pythonic, cloud-ready orchestration platform)

Both tools enable automation of end-to-end machine learning workflows, from preprocessing to training and deployment.

2. Apache Airflow

Apache Airflow is an open-source workflow orchestration platform designed to programmatically author, schedule, and monitor workflows. It is widely used in data engineering and machine learning pipelines. Key features include:

1. DAGs (Directed Acyclic Graphs) – Represent workflows as tasks with dependencies.
2. Scheduling – Automate preprocessing and training jobs at regular intervals.
3. Integration – Supports connectors for databases, cloud platforms, and ML tools.
4. Monitoring – Provides a web UI for tracking pipeline status and failures.
5. Scalability – Handles large and complex workflows across distributed systems.

3. Prefect

Prefect is a modern workflow orchestration tool designed to simplify pipeline development and monitoring. Unlike Airflow, it emphasizes developer-friendly design and cloud-native capabilities. Its main features are:

1. Pythonic API – Build workflows directly in Python without complex configuration.
2. Flows and Tasks – Define pipelines as flows with smaller task components.
3. Fault Tolerance – Automatic retries, error handling, and logging.
4. Hybrid Execution – Run locally, on-prem, or in the Prefect Cloud.
5. Ease of Use – Lightweight setup compared to Airflow, with fast prototyping.

4. Importance of Workflow Automation in ML

- Reproducibility – Ensures data preprocessing and training are consistently executed.
- Scalability – Automates large-scale data and model training pipelines.
- Error Reduction – Handles retries and failure recovery automatically.
- Time Efficiency – Saves manual effort by scheduling recurring tasks.

- Collaboration – Provides shared, visible pipelines for ML and data engineering teams.

5. Workflow of Automating Preprocessing and Training

1. Define Workflow – Create DAGs (Airflow) or Flows (Prefect) representing preprocessing → training → evaluation.
2. Automate Preprocessing – Schedule tasks for data extraction, cleaning, and feature engineering.
3. Automate Training – Run training scripts with defined hyperparameters.
4. Evaluate Models – Log metrics and compare results automatically.
5. Deployment Trigger – Pass trained models to deployment stages if performance thresholds are met.
6. Monitor and Retry – Track execution, handle failures, and rerun tasks as needed.

6. Key Differences Between Apache Airflow and Prefect

Feature	Apache Airflow	Prefect
Workflow Definition	DAGs (YAML + Python operators)	Python-native flows and tasks
Setup & Complexity	Heavier, requires more setup	Lightweight, faster setup
Scheduling	Built-in, strong for batch workflows	External scheduler or Prefect Cloud
Fault Tolerance	Manual configuration	Built-in retries & error handling
Monitoring	Airflow Web UI	Prefect UI or Prefect Cloud Dashboard

Feature	Apache Airflow	Prefect
Best Use Case	Enterprise-scale, complex pipelines	Agile teams needing fast, flexible setup

Execution:

Code:

- **preprocessing.py**

```
import pandas as pd
from sklearn.model_selection import train_test_split
def preprocess():
    # Example: load CSV
    df = pd.read_csv("training_data.csv")
    # Basic cleaning
    df = df.dropna()
    # Train/test split
    train, test = train_test_split(df, test_size=0.2, random_state=42)
    # Save preprocessed files
    train.to_csv("training_data.csv", index=False)
    test.to_csv("training_data.csv", index=False)
    return "Preprocessing done"
```

- **flow.py**

```
from prefect import flow, task
from preprocessing import preprocess
from train_model import train

@task
```

```
def preprocessing_task():
    return preprocess()
@task
def training_task():
    return train()
@flow
def ml_pipeline():
    prep_result = preprocessing_task()  # ☑ lowercase
    print(prep_result)
    train_result = training_task()
    print(train_result)
if __name__ == "__main__":
    ml_pipeline()
```

- **train_model.py**

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
import joblib

def train():
    train_df = pd.read_csv("traning_data.csv")

    X = train_df.drop("target", axis=1)
    y = train_df["target"]

    model = RandomForestClassifier()
    model.fit(X, y)
    joblib.dump(model, "model.pkl")

    return "Training done"
```

Output:

```
pprise<2.0.0,>=1.1.0->prefect) (3.3.1)

E:\Documents\Mlops>python ml_pipeline_perfect.py
21:02:16.714 | INFO    | prefect - Starting temporary server on http://127.0.0.1:8802
See https://docs.prefect.io/v3/concepts/server#how-to-guides for more information on running a dedicated Prefect server.
21:02:22.520 | INFO    | Flow run 'classic-junglefowl' - Beginning flow run 'classic-junglefowl' for flow 'heart-disease
-pipeline'
21:02:23.649 | INFO    | Task run 'load_data-62d' - Finished in state Completed()
21:02:24.870 | INFO    | Task run 'preprocess_data-248' - Finished in state Completed()
21:02:25.654 | INFO    | Task run 'split_data-1a1' - Finished in state Completed()
21:02:26.444 | INFO    | Task run 'train_model-c08' - Finished in state Completed()
✅ Model Accuracy: 0.80
21:02:27.196 | INFO    | Task run 'evaluate_model-f30' - Finished in state Completed()
💾 Model saved to heart_model.pkl
💾 Scaler saved to scaler.pkl
21:02:27.922 | INFO    | Task run 'save_artifacts-eee' - Finished in state Completed()
21:02:27.961 | INFO    | Flow run 'classic-junglefowl' - Finished in state Completed()
21:02:27.975 | INFO    | prefect - Stopping temporary server on http://127.0.0.1:8802
```

Conclusion:

Automating preprocessing and training is a critical step in operationalizing machine learning pipelines. **Apache Airflow** provides a powerful, enterprise-grade solution for orchestrating large-scale and complex workflows, while **Prefect** offers a developer-friendly and flexible approach with strong fault-tolerance and cloud-native execution.

By leveraging these tools, teams can reduce manual intervention, improve reproducibility, and accelerate model development. Whether choosing Airflow for its mature ecosystem or Prefect for its simplicity and modern design, workflow automation ensures that machine learning pipelines are reliable, scalable, and production-ready.