Pipeline Design Documentation

End-to-End Data Management Pipeline for Customer Churn (Assignment I)

This repository contains a complete, submission-ready implementation of an end-to-end data management pipeline for a churn prediction use case.

Contents

- Step 1 – Problem Formulation (included below)
- Step 2 – Data Ingestion
- Step 3 – Raw Data Storage (partitioned “data lake” layout)
- Step 4 – Data Validation (CSV report)
- Step 5 – Data Preparation (clean dataset + EDA outputs)
- Step 6 – Data Transformation & Storage (SQLite)
- Step 7 – Feature Store (metadata + retrieval)
- Step 8 – Data Versioning (DVC workflow)
- Step 9 – Model Building (train + evaluate + save best model)
- Step 10 – Orchestration (Airflow DAG / Prefect flow)

> **IMPORTANT**: This project is designed to run locally. Internet is only required if you want to demonstrate the mock REST API in Step 2. Otherwise, the sample CSV is sufficient.

—

How to Run the Entire Pipeline

Running the Pipeline

1. **Start Docker Desktop**

   Ensure Docker Desktop is running before proceeding.

2. **Navigate to the project directory**

```bash
cd Airflow/airflow-docker
```

3. **Build the Docker containers (without cache)**

```bash
docker compose build --no-cache
```

4. **Start the containers**

```bash
docker compose up -d
```

5. **Access the Airflow UI**

Open http://localhost:8080 in your browser and log in with:

- **Username:** `airflow`

- **Password:** `airflow`

Quickstart (One-click Runner)

Option A – Makefile

```bash
# Run all steps sequentially
make all
```

```

### Option B – Shell Script

```bash

# Same as `make all`

bash run_all.sh

```

Option C – Prefect (optional)

```bash

pip install prefect==2.*

python orchestrate_prefect.py

```

Step 1 – Problem Formulation (Summary for Submission)

Business Problem: Predict addressable customer churn to reduce revenue loss and acquisition costs, and to improve CLV.

Objectives:

1) Predict churn probability for each customer

2) Identify key churn drivers

3) Automate the data pipeline end-to-end

4) Enable continuous model updates via versioned artifacts

Data Sources: Web logs, transaction data, and customer profiles (this demo uses transactions CSV + mock web API).

Outputs: Clean datasets, engineered features, deployable model (.pkl), version-controlled datasets.
```

Metrics: Accuracy, Precision, Recall, F1-score, ROC-AUC.

How Each Step Works

Step 2 – Data Ingestion

- Ingests `sample_data/transactions.csv`

- Optionally fetches web logs from a mock API

- Logs to `logs/ingestion.log`

Run:

```bash
python data_ingestion.py
```

Step 3 – Raw Data Storage

- Moves files from `data/raw/` into a partitioned lake under `data_lake/raw/`:

 `source=<name>/ingestion_date=YYYY-MM-DD/run_id=<YYYYMMDD_HHMMSS>/part-00001.csv`

Run:

```bash
python storage_raw_upload.py
```

Step 4 – Data Validation

- Automated checks (missing, dtypes, ranges, duplicates, date anomalies)

- Report: `reports/data_quality_report.csv`

Run:

```bash
python data_validation.py
```

Step 5 – Data Preparation

- Cleans missing values
- Parses dates, encodes categoricals, scales numerics (keeps IDs intact)
- EDA: histograms & boxplots saved to `reports/`

Run:

```bash
python data_preparation.py
```

Step 6 – Data Transformation & Storage

- Engineers features: total spend, count, average value, tenure days, recency days
- Stores features in SQLite: `data/feature_store.db` (table `customer_features`)

Run:

```bash
python data_transformation.py
```

Step 7 – Feature Store

- Adds metadata table `feature_metadata`
- Registers feature descriptions & versions

- Retrieval function for training

Run:

```bash
python feature_store.py
```

Step 8 – Data Versioning (DVC)

Example workflow:

```bash
pip install dvc
dvc init
dvc add data_lake/raw
dvc add data/clean
dvc add data/feature_store.db
git add data_lake/raw.dvc data/clean.dvc data/feature_store.db.dvc .gitignore
git commit -m "Track raw and transformed datasets with DVC"

# Optional remote example for S3
# dvc remote add -d myremote s3://your-bucket-name/dvcstore
# dvc push
```

Step 9 – Model Building

- Trains Logistic Regression & Random Forest
- Evaluates and saves best model to `models/`

Run:

```bash
python model_building.py
```

Step 10 – Orchestration

Airflow

- DAG in `dags/churn_pipeline_dag.py`
- Suggested schedule: `0 2 * * *`

Setup (example, local):

```bash
export AIRFLOW_HOME="$(pwd)/.airflow"
pip install "apache-airflow==2.9.3" --constraint "https://raw.githubusercontent.com/apache/airflow/constraints-2.9.3/constraints-3.11.txt"
airflow db init
airflow users create --username admin --password admin --firstname Admin --lastname User --role Admin --email admin@example.com
airflow webserver -p 8080    # terminal 1
airflow scheduler           # terminal 2
```

Prefect (alternative)

```bash
pip install prefect==2.*
python orchestrate_prefect.py
```

Screenshots to Capture (for Submission)

- Airflow UI: Graph view + Grid view with a successful run

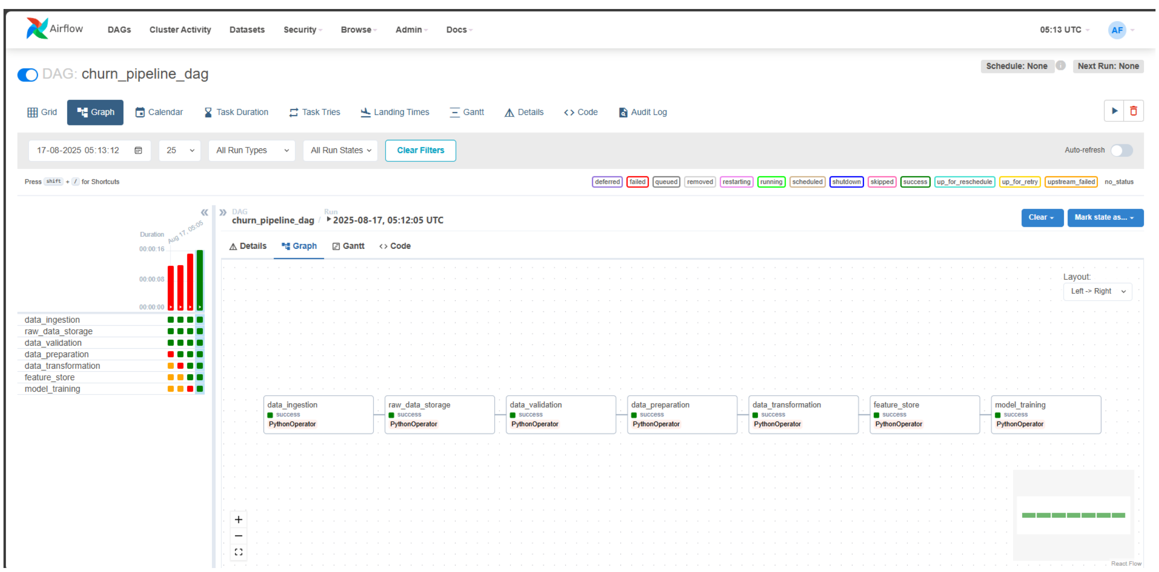
- Task logs for model training metrics

- Folder trees: `data_lake/raw/...`, `reports/*`, `models/*`

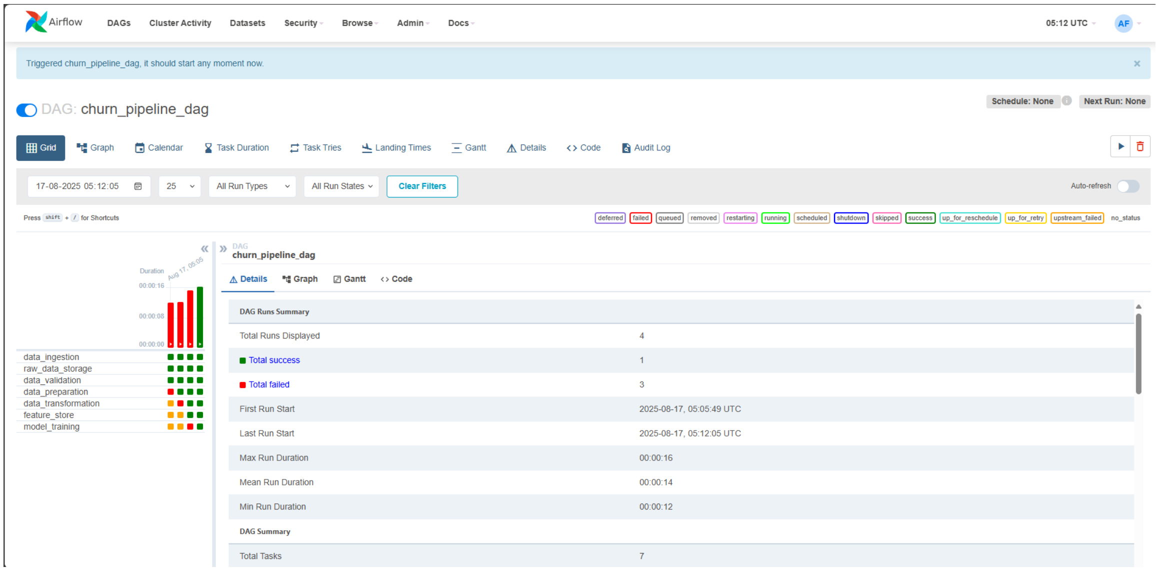
- DVC: terminal output for `dvc status`, `git log`

Screenshots to Include

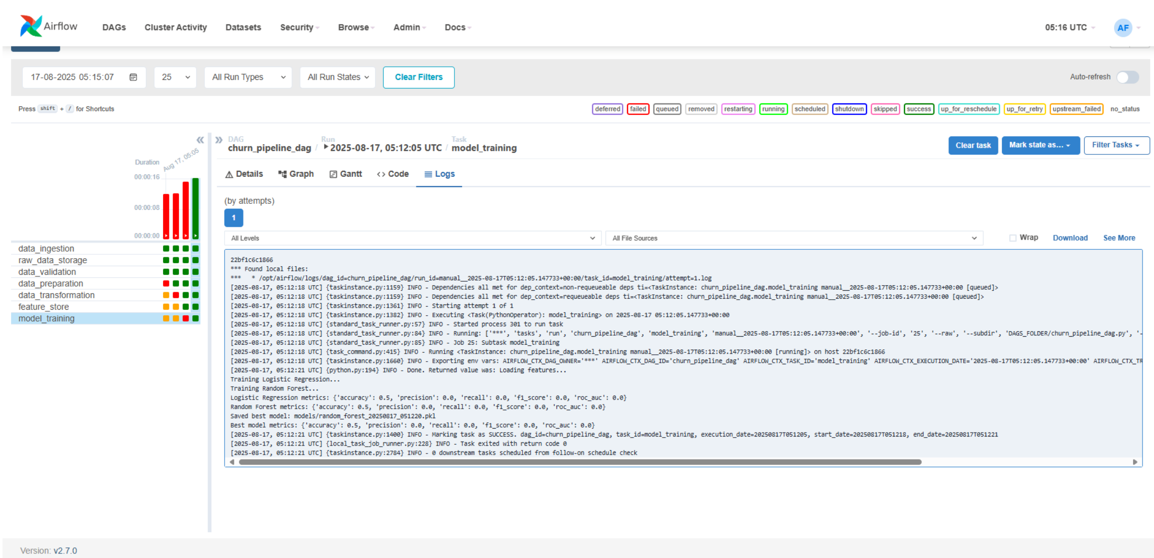
Airflow UI – Graph View of DAG



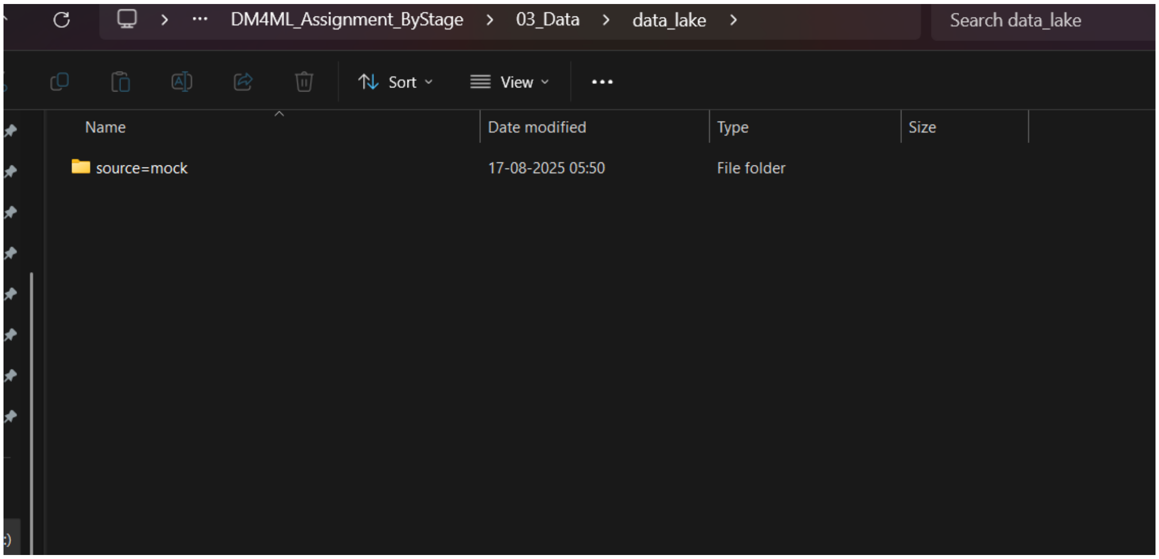
Airflow UI – Grid View after successful run

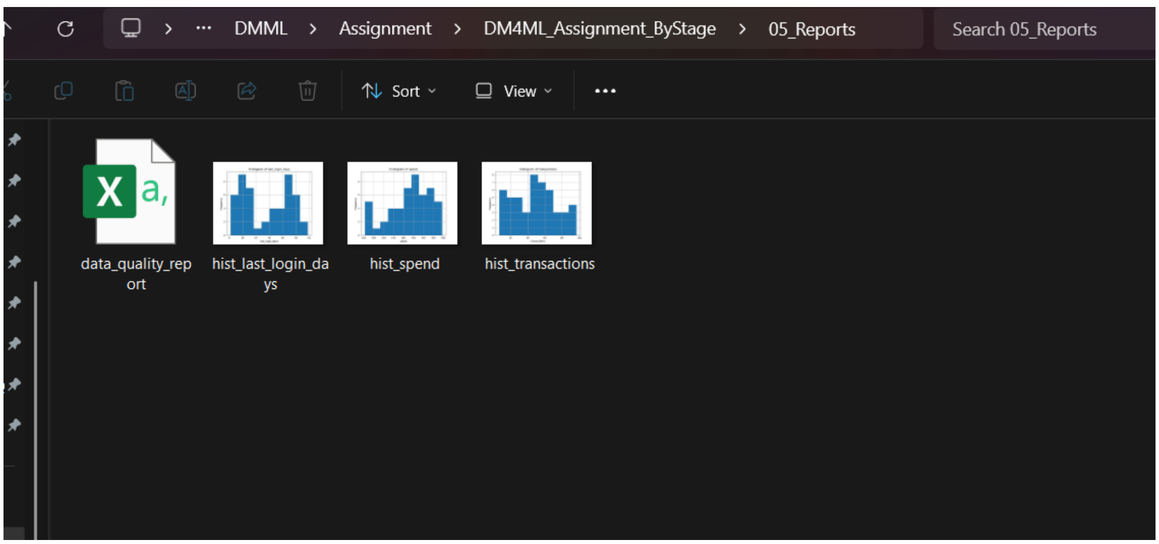


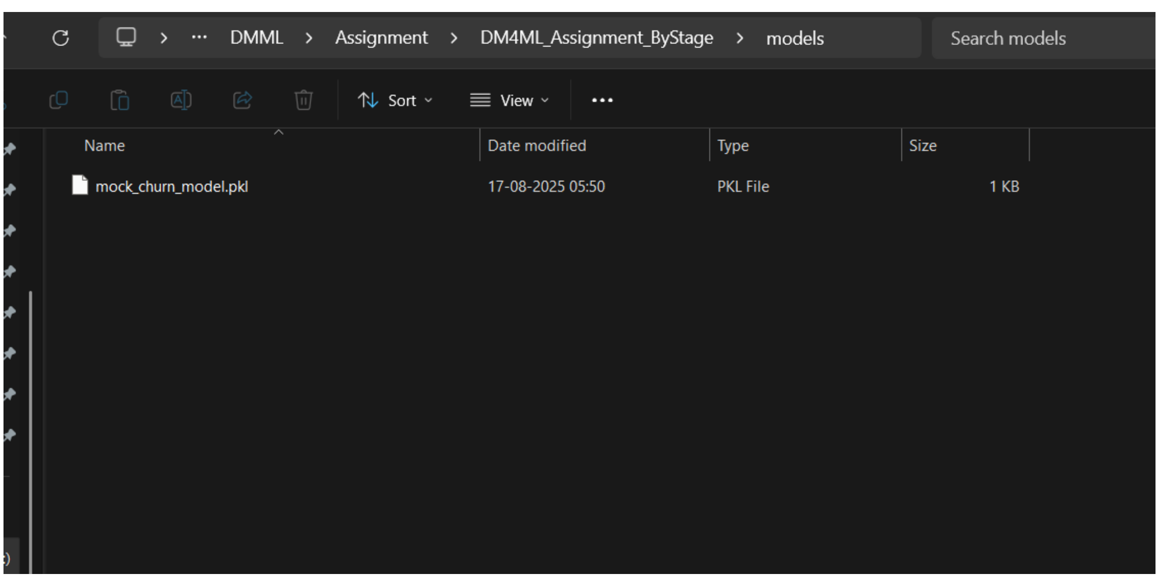
Airflow Task Logs showing model metrics



Local folder structure after run (data_lake, reports, models)


DM4ML_Assignment_ByStage > 03_Data > data_lake >
Search data_lake
Sort
View
Name
Date modified
Type
Size
source=mock
17-08-2025 05:50
File folder


DMML > Assignment > DM4ML_Assignment_ByStage > 05_Reports
Search 05_Reports
Sort
View
data_quality_report
hist_last_login_days
hist_spend
hist_transactions


DMML > Assignment > DM4ML_Assignment_ByStage > models
Search models
Sort
View
Name
Date modified
Type
Size
mock_churn_model.pkl
17-08-2025 05:50
PKL File
1 KB

DVC terminal output (dvc status, git log)

```
airflow@448a1ae96830:/opt/airflow/DM4ML_Assignment_ByStage$
airflow@448a1ae96830:/opt/airflow/DM4ML_Assignment_ByStage$ dvc status
Data and pipelines are up to date.
airflow@448a1ae96830:/opt/airflow/DM4ML_Assignment_ByStage$ git log --oneline --graph --decorate -n 5
* 12b52fa (HEAD -> master) Track transactions.csv with DVC
* 45a4aa2 Stop tracking transactions.csv in Git, move to DVC
* 6ff00bf Initial commit - added project structure
airflow@448a1ae96830:/opt/airflow/DM4ML_Assignment_ByStage$
```