SLTC Research University

Machine Learning Module – Final Assignment

End-to-End Customer Churn Prediction with Full MLOps Pipeline

Assignment Title

Production-Grade ML System with DAGsHub (MLflow + DVC + Airflow + Git + API)

Deadline

March 8th – 11:59 PM
(No extensions unless officially approved)

Submission Requirement (Very Important)

Each group must submit:

1️⃣ 5-Minute Live Demo Video (Mandatory)

The video must clearly include:

Part A – Code Explanation (2–3 minutes)

- Project structure
- DVC pipeline
- MLflow experiment tracking
- Airflow DAG explanation
- DAGsHub integration
- API code explanation

Part B – Project Demonstration (2–3 minutes)

- Run Airflow DAG
- Show MLflow experiments
- Show DVC versioning
- Push to DAGsHub
- Run REST API prediction (Live)
- (Optional Bonus) Show LLM retention message generation

Screen recording must show:

- Your face (small corner view)
- Code
- System running

Dataset

Use:

```
telco_customer_churn_data.csv
```

Target variable:

```
Churn (Yes / No)
```

Dataset details are described in the provided ML Engineer Assignment brief

Mandatory Architecture

You MUST integrate ALL of the following:

```
Git
DVC
MLflow
Airflow
DAGsHub
FastAPI/Flask
Docker
```

Your system must be:

✔ Reproducible
✔ Modular
✔ Automated
✔ Production-ready

Detailed Assignment Breakdown

PART 1 – Data Engineering (15 Marks)

You must:

- Handle missing values
- Convert TotalCharges properly
- Encode categorical variables
- Scale numeric features
- Train-test split

Processed data must be tracked using **DVC**.

PART 2 – Model Development (20 Marks)

Train at least 3 models:

- Logistic Regression
- Random Forest
- XGBoost (or Gradient Boosting)

Evaluate using:

- Accuracy
- Precision
- Recall
- F1 Score
- ROC-AUC

All experiments must be logged using:

✅ MLflow

Track:

- Parameters
- Metrics
- Model artifacts
- Confusion matrix
- ROC curve

PART 3 – DVC Pipeline (10 Marks)

Create proper pipeline stages:

```
data_ingestion
preprocessing
training
evaluation
```

Use:

```
dvc.yaml
```

Push DVC remote to DAGsHub.

PART 4 – Airflow Orchestration (15 Marks)

Create an Airflow DAG with tasks:

1. Data ingestion
2. Data validation
3. Feature engineering
4. Model training
5. Model evaluation
6. Model registration

Use:

- PythonOperator
- Proper task dependencies

The DAG must run end-to-end.

PART 5 – REST API Deployment (15 Marks)

Deploy best model using:

- FastAPI (recommended) OR Flask

Endpoint:

```
POST /predict
```

Output format:

```
{
  "churn_probability": 0.82,
  "prediction": "Yes"
}
```

Dockerize the application.

PART 6 – DAGsHub Integration (10 Marks)

Your DAGsHub repository must show:

- Git commits
- DVC tracked data
- MLflow experiments
- Artifacts

Everything must be connected.

BONUS (Optional – 5 Marks)

LLM-Based Retention Incentive Generator

If model predicts churn:

Generate personalized retention incentive using:

- GPT / Open-source LLM

Example:

> "Dear customer, as a Fiber user with 36 months tenure, we are offering a loyalty discount and free streaming upgrade."

Demonstrate:

- Prompt engineering
- Output formatting control

Required Repository Structure

```
churn-mlops-project/
│
├── data/
│   ├── raw/
│   └── processed/
│
├── src/
│   ├── data_ingestion.py
│   ├── preprocessing.py
│   ├── train.py
│   ├── evaluate.py
│
├── airflow_dags/
│
├── api/
│   ├── main.py
│
├── models/
│
├── Dockerfile
├── requirements.txt
├── dvc.yaml
└── README.md
```

Written Report (Minimum 25 Pages)

Must include:

1. Introduction
2. Problem Definition
3. Dataset Description
4. EDA
5. Feature Engineering
6. Model Comparison
7. Hyperparameter Tuning
8. MLOps Architecture Diagram
9. Airflow DAG explanation
10. MLflow experiment analysis
11. Deployment explanation
12. Challenges and lessons learned
13. Future improvements

Marking Scheme

Component	Marks
Data Engineering	15
Modeling	20
DVC	10
MLflow	10
Airflow	15
REST API	15
DAGsHub Integration	10
Documentation	5
Bonus (LLM)	5
Total	100

Important Rules

❌ No Jupyter-only submissions
❌ No hardcoded pipelines
❌ No manual execution steps
❌ No copied GitHub repositories

✔ Everything must be reproducible
✔ Must follow engineering standards
✔ Clean modular code