Milestone 4 – MLflow, FastAPI, Dashboard, Monitoring, and Local Deployment Documentation

---

1. MLflow Tracking Documentation

Overview

This module integrates **MLflow** to track machine learning experiments such as:

- Hyperparameters
- Evaluation metrics
- Artifacts (plots, models, etc.)
- Model versioning

The tracking is performed inside `train_with_mlflow_tracking()` during model training.

MLflow Experiment Setup

```
mlflow.set_experiment("Customer_Churn_Prediction")
```

Creates or selects an MLflow experiment named **Customer_Churn_Prediction**.

Function: train_with_mlflow_tracking()

Purpose

Train a RandomForestClassifier while automatically logging:

- Best hyperparameters
- Metrics (accuracy, precision, recall, F1, ROC-AUC)
- Trained model
- Feature importance plot

Workflow

1. **Start MLflow Run**
2. `with mlflow.start_run(run_name="RandomForest_Churn_Prediction"):`
3. **Perform GridSearchCV** for hyperparameter tuning.
4. **Train best model** based on F1 score.
5. **Log parameters**
6. `mlflow.log_params(grid_search.best_params_)`
7. **Log all important metrics**
8. **Log model**
9. `mlflow.sklearn.log_model(best_model, "random_forest_model")`
10. **Log artifacts**
    - Feature importance plot saved locally
    - Logged as artifact to MLflow

Outcome

After running the training function, MLflow UI will show:

```
mlflow ui
```

Open: **http://127.0.0.1:5000**

---

2. FastAPI Service Documentation

The API exposes a production-ready machine learning inference service for churn prediction.

Project Structure

```
main.py            → FastAPI application
run_api.py         → Starts the API server
encoders.pkl       → Saved encoders for categorical preprocessing
customer_churn_model.pkl → Serialized RandomForest model
```

FastAPI Application Overview

Endpoints

Endpoint	Method	Purpose
`/`	GET	Root endpoint, returns basic metadata
`/health`	GET	API and model health check
`/model-info`	GET	Returns feature names and model details
`/predict`	POST	Predict churn for one customer
`/batch_predict`	POST	Predict churn for multiple customers

Data Models

Input Model – CustomerData

Defines all required customer attributes such as:

- Demographics
- Account details
- Billing
- Service subscriptions

Output Model – PredictionResponse

Includes:

- churn_probability
- churn_prediction (0 or 1)
- prediction_class (“Churn” / “No Churn”)
- confidence level

Preprocessing Explanation

The API performs:

- Label encoding using saved encoders
- Handling unseen categories
- Ensuring correct feature order
- Filling missing features with defaults

Prediction Workflow

1. Receive JSON input
2. Preprocess using `preprocess_input()`
3. Predict class & probability
4. Compute confidence level
5. Return formatted response

---

3. Streamlit Dashboard Documentation

Overview

A real-time dashboard that interacts with the FastAPI backend to:

- Predict churn
- Analyze CSV batch data
- Display model info
- Monitor API health

Dashboard Sections

1. Single Prediction

- Interactive form for entering customer data
- Sends request to `/predict` endpoint
- Displays results:
  - Probability bar plot
  - Risk level
  - Recommendations

---

2. Batch Analysis

Allows users to:

- Upload CSV
- Preview data
- (Future extension) send batch to API

---

3. Model Info

Displays:

- Model status
- Loaded model type
- Feature list

---

4. API Health

Shows:

- Whether API is running
- Sample prediction test
- Timestamps

Visualization

Uses matplotlib and seaborn for:

- Probability visualization
- Customer risk breakdown

---

4. Model Monitoring System Documentation

---

1. ModelMonitor Class

Purpose:

Provides:

- Data drift detection
- Concept drift monitoring
- Logging predictions

- Generating monitoring reports

Core Methods

log_prediction()

Stores each inference:

- input data
- prediction
- probability
- optional actual label

Used for continuous model evaluation.

---

detect_data_drift()

Uses **Kolmogorov–Smirnov test** to compare:

- reference training data
- new incoming data

Flags drift when `p-value < 0.05`.

---

detect_concept_drift()

Compares model accuracy over last 100 predictions with baseline.

Drift is detected when accuracy drops more than 5%.

---

generate_monitoring_report()

Returns:

- avg probability
- distribution of risk levels
- drift detection summary
- total predictions

---

2. ModelRetrainingStrategy Class

Purpose:

Defines rules determining when the model should be retrained.

Triggers:

- Time-based schedule (default: every 30 days)
- Data drift detected
- Concept drift detected

`retrain_model()`

Retrains the model on new collected data.

---

5. Local Deployment Documentation

Install Dependencies

```
pip install -r requirements.txt
```

Start FastAPI

Run from root directory:

```
uvicorn deployment.app:app --host 0.0.0.0 --port 8000 --reload
```

FastAPI Docs:
➡ http://localhost:8000/docs

Start Streamlit Dashboard

```
streamlit run dashboard/app.py
```

Dashboard URL:
➡ http://localhost:8501