Step 2: ML Pipeline Experiment Tracking with MLflow

University: Tek-Up
Group: ING-4-SDIA

Tutor: Haythem Ghazouani

1 Introduction

In professional Data Science, training a model in a single notebook is not enough. You need to build **reproducible pipelines** and **track every experiment** to find the best configuration.

2 The Modular Pipeline

We use Scikit-Learn's `Pipeline` and `ColumnTransformer` to ensure that preprocessing (Scaling, Encoding) is always consistent between training and inference.

- **Numerical Features:** Scaled using `StandardScaler`.
- **Categorical Features:** Encoded using `OneHotEncoder`.
- **Classifier:** The algorithm (Random Forest, XGBoost, etc.).

3 Advanced Algorithms: Boosting vs. Ensembling

This module focuses on:

- **Ensembling (Bagging):** Reducing variance using **Random Forest**.
- **Boosting:** Reducing bias sequentially using **XGBoost** or **LightGBM**. These are the industry standard for tabular data.

4 Experiment Tracking with MLflow

The MLOps Standard: MLflow

MLflow is an open-source platform to manage the ML lifecycle. Every time you train a model, you should log:

1. **Parameters:** Hyperparameters like `learning_rate`, `n_estimators`, etc.
2. **Metrics:** Performance scores like `Accuracy`, `F1-Score`, `ROC-AUC`.
3. **Artifacts:** The serialized model files (.pkl, .onnx) and plots.

5 Implementation Guide

The script `code/modeling.py` demonstrates how to wrap a model in an MLflow run:

```python
import mlflow

mlflow.set_experiment("Bank_Churn_Prediction")

with mlflow.start_run(run_name="XGBoost_Exp1"):
    # Log hyperparameters
    mlflow.log_param("learning_rate", 0.05)

    # Train pipeline
    pipeline.fit(X_train, y_train)

    # Log results
    mlflow.log_metric("accuracy", 0.865)

    # Save the model
    mlflow.sklearn.log_model(pipeline, "churn_model")
```

6 Visualization: MLflow UI

To view your experiments, run the following command in your terminal:

```
mlflow ui
```

Then open `http://localhost:5000` in your browser. You can now compare all your runs side-by-side.

7 Exercise

1. Implement a **VotingClassifier** that combines Random Forest and XGBoost.
2. Log a **Confusion Matrix** plot as an artifact in your MLflow run.
3. Try varying the `max_depth` of XGBoost and identify the point where the model starts over-fitting.