🧠 Project: Probability of Default (PD) Prediction Model with MLflow Tracking and Registry

This document provides a clear, beginner-friendly explanation of a machine learning (ML) mini-project using MLflow. It walks through the entire process step by step, even for those who have not used MLflow or built ML models before.

🎯 Objective

To build a machine learning model that predicts whether a customer will default on a loan (Probability of Default or PD), and to track the entire model development and deployment process using **MLflow**.

🧰 What is MLflow?

MLflow is a tool that helps you manage the lifecycle of machine learning models. With MLflow, you can:

- Track model parameters, metrics, and artifacts (like plots)
- Save and load models.
- Register models in a model registry.
- Manage different versions of models.
- Transition models to various stages (e.g., Staging, Production)

🧱 Step-by-Step Explanation of the Project

✅ Step 1: Install Libraries

```
!pip install pandas scikit-learn matplotlib seaborn mlflow
```

These libraries help us with data processing (pandas), machine learning (scikit-learn), visualizations (matplotlib, seaborn), and MLflow tracking.

✅ Step 2: Import Libraries

We load all the required libraries for model training and tracking.

✅ Step 3: Generate Sample Data

from sklearn.datasets import make_classification

We generate a fake dataset that simulates customer data for loan applications, with a target variable representing whether they defaulted or not.

✅ Step 4: Train a Classification Model

from sklearn.ensemble import RandomForestClassifier

We train a **Random Forest classifier** to predict loan default based on customer features. We evaluate it using two metrics:

- **Accuracy**: How many predictions were correct
- **ROC AUC**: How well the model separates defaulters from non-defaulters

---

✅ Step 5: Track the Model Using MLflow

```
mlflow.set_experiment("PD_Classification_Experiment")

with mlflow.start_run():

  ...
```

Inside this block, we:

- Log **parameters** (e.g., number of trees, depth)
- Log **metrics** (e.g., accuracy, ROC AUC)
- Log **artifacts** (e.g., confusion matrix plot, ROC curve)
- Log and **register** the trained model to the **Model Registry**

✅ Step 6: Visualize Model Performance

We generate:

- A **confusion matrix**: Shows true/false predictions
- A **ROC curve**: Plots model's performance across thresholds Both are logged as images in MLflow.

✅ Step 7: Register and Version the Model

```
mlflow.sklearn.log_model(model, "model",
registered_model_name="pd_model_v1")
```

This saves the model in MLflow and adds it to a **named model registry** (pd_model_v1). Each time you log a new model with this name, MLflow creates a new **version**.

✅ Step 8: Load the Model Later for Use

```
mlflow.sklearn.load_model("models:/pd_model_v1/latest")
```

You can load the saved model any time in the future and use it to make predictions.

✅ Step 9: Promote the Model to Production

MlflowClient().transition_model_version_stage(...)

This marks a version of the model as **"Production"**, meaning it's now the official model to use. You can also transition models to:

- **Staging** (for testing)
- **Archived** (for old versions)

✅ Step 10: Fine-Tune the Model

from sklearn.model_selection import GridSearchCV

We use grid search to find the best hyperparameters for our model and improve performance. The best model can also be logged and promoted using MLflow.

📊 MLflow Dashboard

After starting the MLflow UI with:

mlflow ui

You can view all your:

- Experiments
- Runs
- Metrics & parameters
- Model versions
- Visual artifacts
- Registered models

Access it via: http://127.0.0.1:5000

✅ Final Result

You've created a complete end-to-end ML pipeline that:

- Trains and evaluates a model.
- Logs everything automatically
- Registers and version-controls the model.
- Loads and reuses the model.
- Promotes the best model to production.

🏁 Why This Is Valuable

This mimics a real-world ML workflow used in companies:

- Track experiments
- Compare results.
- Reuse & deploy reliable models.
- Keep everything reproducible & organized.
