Automated Machine Learning System for Loan Risk Prediction

1. Overview

This document outlines the design and automation plan for a machine learning (ML) system to predict loan application risk. The system follows a structured pipeline to preprocess data, build models, and evaluate performance. The automation ensures efficient, scalable, and reproducible model training and deployment.

2. Pipeline Design

The ML pipeline consists of multiple sequential steps, each performing a specific task in the modeling workflow:

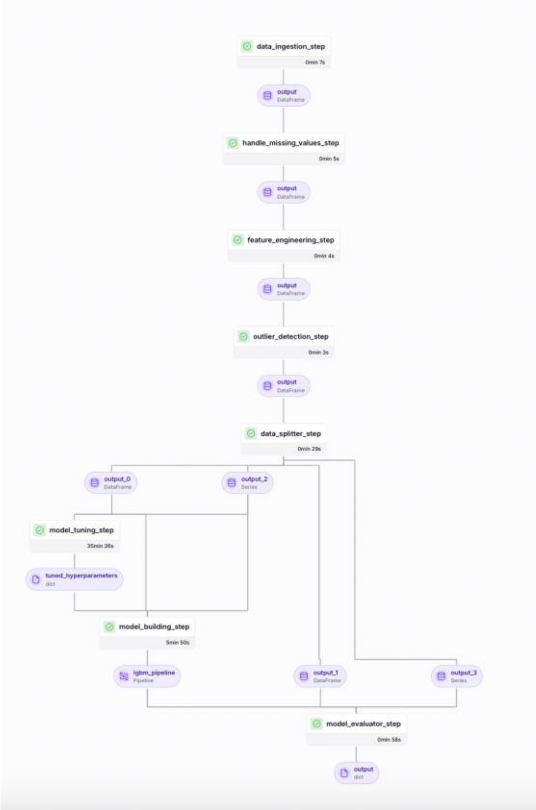

Figure 1: Pipeline Diagram

Step 1: Data Ingestion

- Ingests data from a ZIP file using the appropriate DataIngestor.
- Extracts the ZIP file and loads the relevant CSV file into a Pandas DataFrame.

Consideration:

- Ensures data integrity through schema validation.

Step 2: Handling Missing Values

- Drops irrelevant columns to remove unnecessary data.
- Fills missing numeric values using the **mean** to ensure data consistency.
- Fills missing categorical values using the **mode** to maintain category distributions.
- Uses a modular approach with strategy-based missing value handling (`DropMissingValuesStrategy` and `FillMissingValuesStrategy`).

Consideration:

- Use appropriate imputation strategies for numerical and categorical data.

Step 3: Feature Engineering

- **Feature Creation**: Generates domain-specific features like **is_monthly_payment** and **loan_to_payment_ratio**.
- **Feature Transformation**: Standardizes/normalizes numerical features and encodes categorical variables.

Consideration:

- Create meaningful features to improve model performance.
- Standardize and encode features correctly.

Step 4: Outlier Detection

- Implements an **OutlierDetector** that supports multiple detection strategies, including **Z-Score** and **IQR-based methods**.
- Ensures only numeric columns are processed for outlier detection.
- Applies the **IQR method** to detect values outside the interquartile range (1.5 * IQR).
- Provides flexible outlier handling strategies:
  - **Remove**: Excludes rows containing outliers from the dataset.
  - **Cap**: Adjusts extreme values to the 1st and 99th percentiles.
- Supports visualization of outliers using box plots for feature analysis.

Consideration:

- Ensure consistency in training and inference.

Step 5: Data Splitting

- Uses `DataSplitter` with a configurable strategy.
- Default: **80/20 train-test split**, **random seed = 42**, **stratified sampling**.

- **Handles class imbalance** using **SMOTETomek**, with fallback to **RandomOverSampler** if any class has <2 samples.
- Outputs `X_train`, `X_test`, `y_train`, and `y_test`.
- Supports alternative splitting strategies if needed.

Consideration:

- Maintain reproducibility with a fixed random seed.

Step 6: Model Tuning

- Hyperparameter tuning using **Optuna** to optimize LightGBM parameters.
- Uses **Stratified K-Fold cross-validation** to ensure robust evaluation.
- Incorporates preprocessing pipelines for **numerical** and **categorical** features.
- Evaluates model performance using **Accuracy**.
- Returns the best hyperparameters for model training.
- Saves tuned parameters with a timestamp.

Consideration:

- Validate on multiple metrics to ensure robustness. Avoid overfitting during hyperparameter search.

Step 7: Model Building

- Trains **LightGBM** with optimized parameters.
- Implements **preprocessing pipelines**.
- Tracks experiments with **MLflow** and **ZenML**.
- Saves **trained model** and **expected feature columns**.
- Serializes model using **pickle** for reproducibility.

Consideration:

- Regularization and early stopping should be applied to avoid overfitting.
- Ensure proper **versioning** of saved models.

Step 8: Model Evaluation

- Evaluates the trained model using the **ModelEvaluator** class.
- Uses the **ClassificationModelEvaluationStrategy** to compute key classification metrics.
- Computes the following metrics:
  - **Accuracy**
  - **Precision** (weighted)
  - **Recall** (weighted)
  - **F1-Score** (weighted)
  - **AUC-ROC** (one-vs-rest for multi-class problems)
  - **Log Loss**

  - **Classification Report** (detailed breakdown of performance per class)
  - **Confusion Matrix** (to analyze misclassifications)
- Logs evaluation metrics for tracking and comparison.

Consideration:

- Use relevant metrics (Accuracy, F1-Score, AUC-ROC).
- Track and compare performance for improvements.

3. Automation Strategy

- **Orchestration**: The pipeline is automated using workflow orchestration tools (**ZenML** and **MLflow**).
- **Continuous Integration/Continuous Deployment (CI/CD)**: Implementing CI/CD ensures seamless updates to the model.
- **Monitoring & Logging**: Logs data processing and model performance metrics for troubleshooting and reproducibility.
- **Model Deployment**: The trained model is deployed as an API using FastAPI or integrated into an existing system.

4. Conclusion

This automated ML system streamlines the loan risk prediction process by structuring data processing, model training, and evaluation in a reproducible and scalable manner. By incorporating automation and best practices, it ensures high-performance and fair decision-making in loan risk assessment.