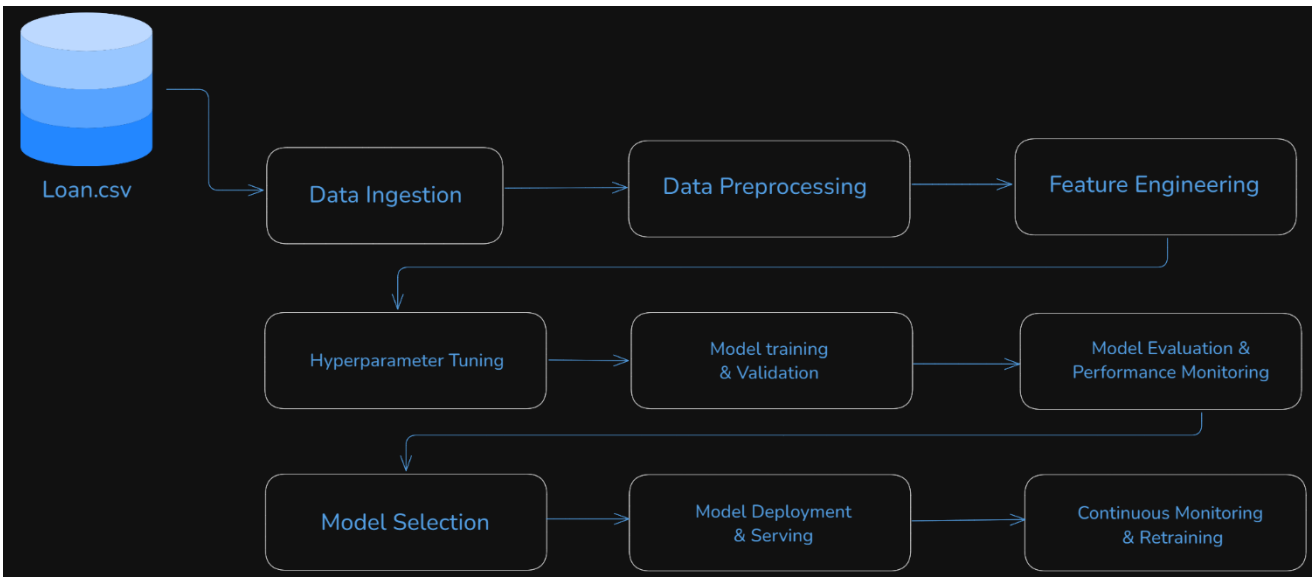
Machine Learning Pipeline Design and Implementation Plan

1. Pipeline Overview

The goal of this pipeline is to predict the loan status based on various features of loan applications. The pipeline is designed to be modular, scalable, and adaptable for continuous improvement. It consists of eight key steps: Data Ingestion, Data Preprocessing, Feature Engineering, Hyperparameter Tuning, Model Training & Validation, Model Evaluation, Model Selection & Deployment, and Continuous Monitoring & Retraining.

2. Pipeline Diagram



3. Detailed Pipeline Plan

Step 1: Data Ingestion

Description:

- Load data from loan.csv.
- Handle missing values and inconsistencies early on.

Implementation:

- Use Python's pandas to load data and validate it for any missing values or data type issues.

Considerations:

- Data integrity and consistency must be ensured at this step. Automate data ingestion to handle updates.

Step 2: Data Preprocessing

Description:

- Clean, transform, and encode the raw data into a format that can be used by the model.
- Handle missing values, encode categorical variables, and scale numerical features.

Implementation:

- Implement using scikit-learn's Pipeline feature for modular and reproducible preprocessing.

Considerations:

- Ensure transformations are applied consistently across training and test data to prevent data leakage.

Step 3: Feature Engineering

Description:

- Create new features from existing data to improve model performance.

Implementation:

- Generate custom features relevant to the problem, like loan_to_payment_ratio.

Considerations:

- Features should be relevant and add predictive value. Automate to maintain consistency.

Step 4: Hyperparameter Tuning

Description:

- Optimize model hyperparameters using techniques like Bayesian optimization with Optuna.

Implementation:

- Perform stratified K-Fold cross-validation with SMOTE for class imbalance handling.

Considerations:

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

Step 5: Model Training & Validation

Description:

- Train the model using the best hyperparameters and validate using cross-validation.

Implementation:

- Train the model using LightGBM and validate on a separate validation set.

Considerations:

- Regularization and early stopping should be applied to avoid overfitting.

Step 6: Model Evaluation

Description:

- Evaluate the model using metrics like accuracy, AUC, precision, recall, F1-score, and log loss.

Implementation:

- Use scikit-learn's metrics to assess performance and generate a classification report and confusion matrix.

Considerations:

- Consider class imbalance in the evaluation by focusing on metrics like recall and F1-score.

Step 7: Model Selection & Deployment

Description:

- Select the best model based on evaluation metrics and deploy it. Save preprocessing objects (e.g., encoders, scalers) for future use.

Implementation:

- Save the model and preprocessing pipeline with joblib, deploy as an API using FastAPI.

Considerations:

- Ensure consistency between training and inference by saving the preprocessing pipeline. Secure and monitor the API.

Step 8: Continuous Monitoring & Retraining

Description:

- Monitor model performance in production and retrain as necessary using new data.

Implementation:

- Set up logging and monitoring to track performance, and schedule retraining with new data.

Considerations:

- Automate monitoring and retraining to adapt to changes in data distribution. Ensure the model remains accurate and relevant.