CAD Boost With Optuna

🧠 Goal of the Model

The model predicts whether a borrower will default on a loan (binary classification of loan_status). It uses **CatBoost**, a gradient boosting algorithm optimized for handling categorical variables and tabular data, combined with **Optuna** for hyperparameter tuning and **SHAP** for interpretability.

🧱 Step-by-Step Explanation

🔧 STEP 1-2: Library Installation & Imports

- Loads all necessary libraries for machine learning (CatBoost), tuning (Optuna), visualization (Seaborn, Matplotlib), and interpretability (SHAP).

📦 STEP 3: Load & Preprocess Data

- Reads the dataset from Google Drive.
- Handles missing values:
  - person_emp_length and loan_int_rate are filled with their **median**.
- Sets up:
  - X = features
  - y = target (loan_status)
- Lists categorical features so CatBoost can natively process them (no one-hot encoding needed).

🎯 STEP 4: Hyperparameter Optimization with Optuna + Cross-Validation

- Uses **Optuna** to find the best model parameters via **Bayesian optimization**.

- Evaluates model performance using **3-fold Stratified Cross-Validation** to preserve class balance.
- Optimization metric: **AUC (Area Under the ROC Curve)**.
- Sample of hyperparameters tuned:
    - depth, learning_rate, l2_leaf_reg, rsm, etc.

📌 **Why CV & AUC?**
Cross-validation ensures the model generalizes well to unseen data, and AUC is great for imbalanced classification problems like loan defaults.

✅ **STEP 5: Final Model Training**

- Trains the **best CatBoost model** (as found by Optuna) on an 80% training split.
- Evaluates it on a 20% hold-out test set.

📊 **STEP 6: Evaluation Metrics**

- **Accuracy**: % of correct predictions.
- **F1 Score**: Balance between precision and recall—especially useful for imbalanced classes.
- **ROC AUC**: Ability to discriminate between default and non-default.
- **Classification Report**: Shows precision, recall, and F1 per class.

📄 **Sample Output Mentioned**:

- **Accuracy**: ~91%
- **ROC AUC**: ~0.86–0.90
- **F1 Score**: ~0.55–0.65
  (Note: Lower F1 is expected in imbalanced classes; most loans are likely repaid, making defaults the minority class.)

🔍 **STEP 7: Model Explainability via SHAP**

- Uses SHAP to explain **feature importance**:
    - **Beeswarm Plot**: Shows how much each feature contributes to the prediction.
    - **Summary Plot (Bar)**: Aggregates feature importance over all samples.

👑 **Top Features Identified**:

- person_income
- loan_int_rate
- loan_percent_income
- loan_grade

These features heavily influence whether a loan will default.

⚠️ **STEP 8: Drift Detection with PSI**

- **Simulates data drift**:
    - Alters distribution of loan_int_rate and loan_intent.
- Calculates **PSI (Population Stability Index)** between original and drifted model predictions.
- PSI thresholds:
    - < 0.1: no drift
    - 0.1–0.2: minor drift
    - 0.2: significant drift → **retraining suggested**

🎯 Helps monitor **model degradation over time**, especially in volatile economic environments.

💾 **STEP 9: Save, Visualize, & Export Results**

- Saves final predictions to CSV and downloads.
- Visualizes:
    - **Confusion Matrix**: Breakdown of TP, FP, TN, FN.

- **ROC Curve**: Tradeoff between sensitivity and specificity.
- **Precision-Recall Curve**: Useful when dealing with imbalanced data.
- **XGBoost Feature Importance** (this seems like leftover code; xgb_model isn't defined—it likely doesn't run).

✅ **Results Summary**

Metric	Value (Approximate)
Accuracy	✅ ~89.5% – 91%
ROC AUC	📈 ~0.86 – 0.90
F1 Score	🔁 ~0.55 – 0.65
PSI (after drift)	📊 Printed value (e.g., 0.22)
SHAP Key Predictors	income, interest rate, etc.
Final Classification Accuracy	📌 93% (your reported outcome)

⚙️ **Advanced Design Highlights**

1. **Minimal Preprocessing**:
   - CatBoost handles categorical encoding internally.
2. **Tuned & Regularized**:
   - Uses L2 regularization and bagging_temperature to reduce overfitting.
3. **Drift Awareness**:
   - Incorporates PSI and retraining triggers → **model ops (MLOps) ready**.
4. **Explainable**:
   - SHAP gives granular insight into feature contributions.

⚠️ **Potential Issues or Notes**

- ⚠️ xgb_model is used in the feature importance section, but nowhere in the script is XGBoost actually trained—this part likely throws an error unless fixed or removed.
- ✅ Suggestion: Replace it with final_model.get_feature_importance() if you want feature importance from CatBoost.