Predicting Banking Loan Defaults

Team Members: Dheekshith, Ninad, Mouneesh, Mahith

Github Link:
https://github.com/f20212948/Gramener_Training/tree/main/Project1_Banking_Default_Group1

Kaggle Data Link:
https://www.kaggle.com/datasets/yasserh/loan-default-dataset/data?select=Loan_Default.csv

This project focuses on building and comparing machine learning models to predict loan defaults (the Status column) using the provided banking loan dataset. The process involves comprehensive data analysis, preprocessing (handling missing values, skewness, and categorical features), feature selection, managing class imbalance, and training various classification models, with a focus on **XGBoost**.

Project Structure and Flow

The analysis follows a standard machine learning workflow:

1. **Data Loading and Initial Exploration:** Load the dataset and perform initial checks for shape, data types, and missing values.
2. **Exploratory Data Analysis (EDA):** Visualize distributions, check for outliers, analyze correlations, and examine the relationship between features and the target variable (Status).
3. **Data Preprocessing and Transformation:**
   - **Handling Missing Values:** Missing values in numerical columns are imputed using the **median strategy** (SimpleImputer).
   - **Skewness Transformation:** Highly skewed numerical features are transformed using **$\log(1+x)$** to make their distributions more Gaussian-like.
   - **Categorical Encoding:** Object type columns are imputed with the **most frequent value** and then transformed using **One-Hot Encoding**.
4. **Feature Selection and Scaling:**
   - A smaller sample of the data is used for **Hyperparameter Tuning** to save time.
   - Numerical features are **Standard Scaled**.
   - The **XGBoost Classifier** is used on the sampled data to identify the **Top 30 Most Important Features**.
5. **Handling Class Imbalance:** The full training data is subjected to **Random Under-Sampling (RUS)** to balance the majority (non-default) and minority (default) classes.
6. **Model Training and Evaluation:** Three main models are trained, tuned, and evaluated: **Logistic Regression**, **Random Forest Classifier**, and **XGBoost Classifier**.
   - **Hyperparameter Tuning** is performed using RandomizedSearchCV on the sampled data to find optimal parameters (e.g., C for Logistic Regression, max_depth for Random Forest and XGBoost).

    - Final models are trained on the **under-sampled training data** and evaluated on a separate **test set** and **validation set**.
7. **Conclusion:** Compare model performance using metrics like **ROC AUC**, **Precision-Recall AUC (PR AUC)**, and **Accuracy**.

Key Code Logic Details

Section	Technique Used	Purpose
Missing Values	SimpleImputer(strategy='median')	To fill missing values in numerical columns, as the median is less sensitive to outliers.
Data Cleaning	Custom type_transform and purpose_transform functions	To convert cryptic categorical codes (e.g., 'type1', 'p1') into descriptive strings (e.g., 'Conventional Loans', 'Home Purchase') for better interpretability.
Skewness	numpy.log1p(x)	Applies a $\log(1+x)$ transformation to reduce positive skewness in numerical features.
Categorical Processing	ColumnTransformer with SimpleImputer('most_frequent') and OneHotEncoder	Handles both imputation for categorical features and converting them into a format suitable for machine learning models.

Hyperparameter Tuning	RandomizedSearchCV	Efficiently searches a wide range of hyperparameters for **Logistic Regression**, **Random Forest**, and **XGBoost** to find the best configuration based on **ROC AUC** score.
Feature Selection	XGBClassifier.feature_importances_	Uses the tuned XGBoost model's feature importance to select the **Top 30 most impactful features**, reducing dimensionality for final models.
Class Imbalance	RandomUnderSampler from imblearn	Reduces the number of samples in the majority class to balance the dataset, which is crucial for improving the prediction of the minority class (loan defaults).
Final Evaluation	ROC AUC, PR AUC, Classification Report	These metrics are critical for imbalanced classification tasks. **PR AUC** is especially important as it focuses on the performance on the positive class (default).

How to Run the Code

1. **Save the Code:** Ensure the Python script containing the code is saved (e.g., loan_prediction.py).
2. **Acquire Data:** Ensure the dataset Loan_Default.csv is in the same directory as the script.

Execute: Run the script from your terminal:
Bash
```
python loan_prediction.py
```

3. The script will output the following:

- Multiple EDA plots (histograms, boxplots, heatmaps).
- Best hyperparameters and best ROC AUC scores from RandomizedSearchCV.
- Feature importance list.
- Classification Reports and performance scores (Accuracy, ROC AUC, PR AUC) for each model on the test and validation sets.
- The final ROC Curve and Precision-Recall Curve plots comparing all three models.

Required Installations

This project requires several common Python libraries for data manipulation, machine learning, and visualization.

You can install all necessary packages using pip:

```bash
pip install pandas numpy matplotlib seaborn scikit-learn xgboost imblearn
```

Library	**Purpose**
pandas	Data manipulation and analysis.
numpy	Numerical operations, especially for mathematical transformations.
matplotlib	Creating static, interactive, and animated visualizations.
seaborn	High-level interface for drawing attractive statistical graphics.
scikit-learn	Machine learning framework (models, preprocessing, evaluation).
xgboost	Highly efficient Gradient Boosting implementation.
imblearn	Handling imbalanced datasets (specifically, RandomUnderSampler).

Summary of Model Performance

Based on the test set evaluation after addressing class imbalance and feature selection, the **XGBoost Classifier** is the recommended model:

Model	**Accuracy**	**ROC AUC**	**PR AUC**
Logistic Regression	0.867	0.831	0.757
Random Forest	0.855	0.806	0.735
XGBoost Classifier (Best)	**0.869**	**0.842**	**0.768**

The XGBoost model demonstrates the best overall ability to distinguish between loan defaulters and non-defaulters.