Default of Credit Card Clients Dataset

Hà Bảo Ngọc • 27/09/2025

Introduction & Objectives

Problem: Predict whether a credit card client will default next month.

Practical importance:

- Support credit risk management.
- Assist lending decision-making.

Objectives

- Build a reliable prediction model.
- Balance accuracy with interpretability.

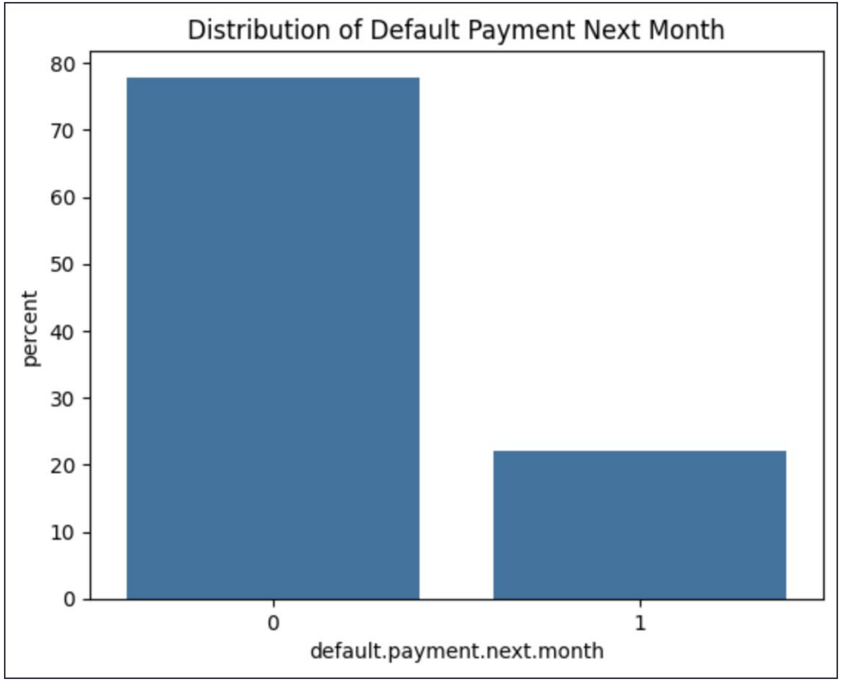
Data Overview

Source: UCI Credit Card Dataset (30,000 clients).

Features:

- Demographics: gender, age, marriage, education.
- Credit info: LIMIT_BAL, payment history (PAY_x).
- Billing & repayment records (BILL_AMT, PAY_AMT, last 6 months).

Data Characteristics



Imbalanced classes (~22% default vs. 78% non-default).

```
df.nunique() # Use this to divide in
✓ 0.0s

LIMIT_BAL                        81
SEX                               2
EDUCATION                         7
MARRIAGE                          4
AGE                              56
PAY_0                            11
PAY_2                            11
PAY_3                            11
PAY_4                            11
PAY_5                            10
PAY_6                            10
BILL_AMT1                     22723
BILL_AMT2                     22346
BILL_AMT3                     22026
BILL_AMT4                     21548
BILL_AMT5                     21010
BILL_AMT6                     20604
PAY_AMT1                       7943
PAY_AMT2                       7899
PAY_AMT3                       7518
PAY_AMT4                       6937
PAY_AMT5                       6897
PAY_AMT6                       6939
default.payment.next.month        2
dtype: int64
```

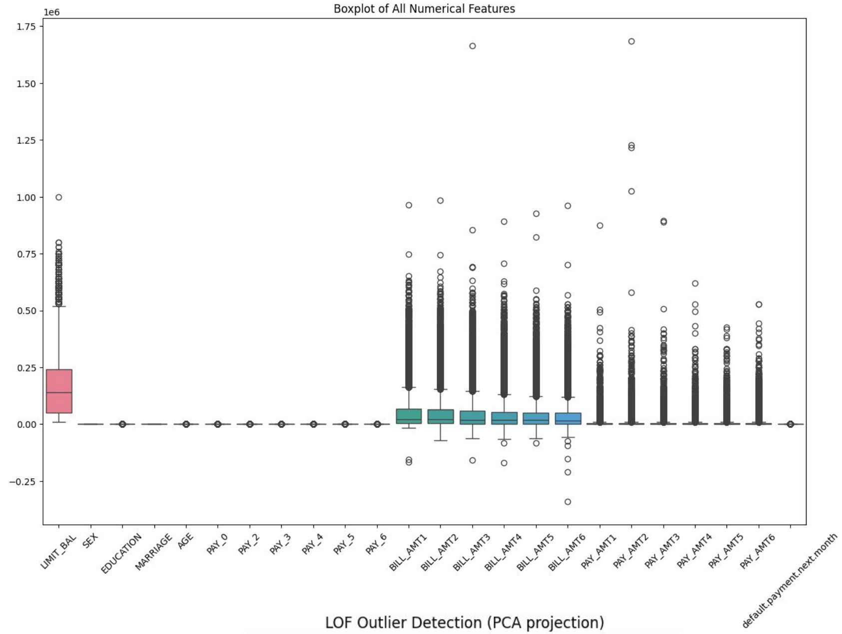
Mix of numerical & categorical variables.

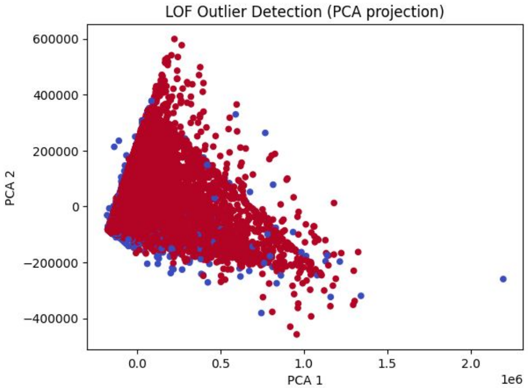
```
df.isnull().sum() # Dataset has
```

✓ 0.0s

```
LIMIT_BAL                     0
SEX                           0
EDUCATION                     0
MARRIAGE                      0
AGE                           0
PAY_0                         0
PAY_2                         0
PAY_3                         0
PAY_4                         0
PAY_5                         0
PAY_6                         0
BILL_AMT1                     0
BILL_AMT2                     0
BILL_AMT3                     0
BILL_AMT4                     0
BILL_AMT5                     0
BILL_AMT6                     0
PAY_AMT1                      0
PAY_AMT2                      0
PAY_AMT3                      0
PAY_AMT4                      0
PAY_AMT5                      0
PAY_AMT6                      0
default.payment.next.month    0
dtype: int64
```

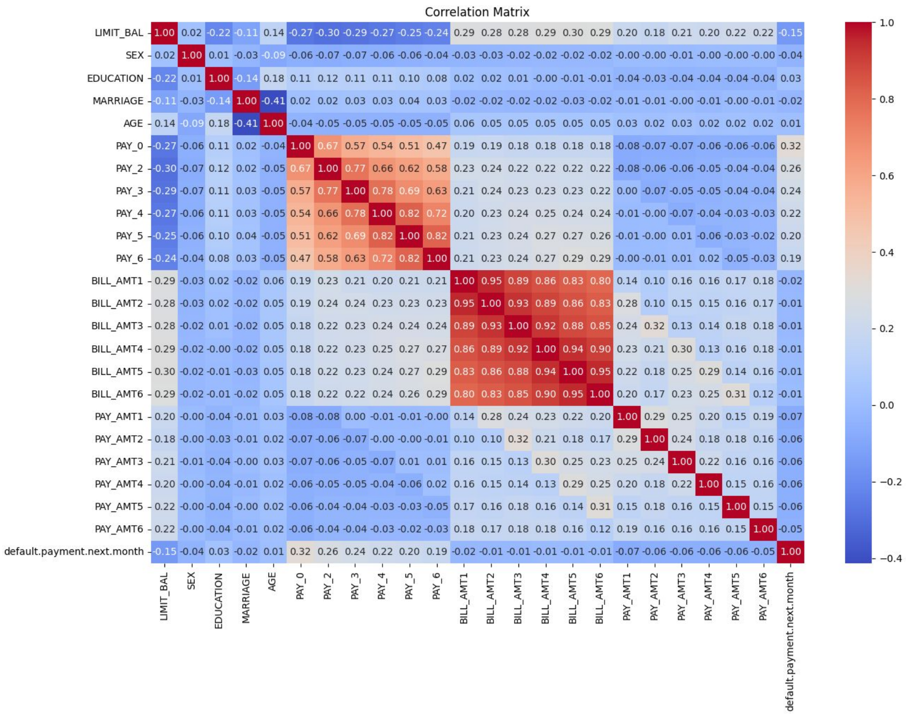
No missing value





Contains a large number of outliers

->These extreme values reflect real-world cases of very high-income or high-debt clients, but they can also skew model training.



Some features are highly correlated (e.g., bill amounts across months, repayment amounts).

Methodology

- **EDA**: Checked data quality, outliers, imbalance, and feature–target relationships.
- **Train/Test Split**: Time-series split (past -> future).
- **Preprocessing**: Robust scaling, one-hot/ordinal encoding, log transform for skew.
- **Modeling**: Cross-validation with F1-score; compared tree-based vs. linear models.
- **Evaluation**: Confusion matrix, ROC/AUC, precision–recall.
- **Advanced Analytics**: Feature importance, overfitting checks, feature stability (PSI), and temporal stability via time-series CV.

Implementation (Brief)

Exploratory Data Analysis (EDA)

We first performed an extensive EDA to understand the dataset's quality and structure.

- **Data quality check:** Assessed missing values, duplicate entries, inconsistencies, and outliers.
- **Target variable analysis:** Examined the distribution of the default vs. non-default classes to confirm class imbalance.
- **Univariate analysis:** Analyzed the distribution of each feature individually (both categorical and numerical).
- **Multivariate analysis:** Studied correlations and interactions between features.
- **Multivariate with target:** Investigated how different features relate to the target class to identify potentially strong predictors.

Implementation (Brief)

Train/Test Split

Since the data has a temporal nature, we used a time-series train-test split instead of random splitting. This approach better simulates real-world prediction scenarios, where past data is used to predict future outcomes.

Implementation (Brief)

Data pipeline

- **Numerical features:** Applied Robust Scaling to handle outliers effectively.
- **Categorical features (nominal):** Used One-Hot Encoding, which preserves interpretability while avoiding artificial ordering.
- **Categorical features (ordinal):**
  - **For tree-based models:** kept them as-is.
  - **For linear models:** applied Ordinal Encoding to respect feature order.
- **Highly skewed features:** Applied log transformation to reduce skewness and improve linear model performance.
- **Evaluation of preprocessing:** Compared the effects of these transformations on feature distributions using visualizations, sparsity checks, and skewness measures.

Implementation (Brief)

Model Selection

We adopted a **cross-validation strategy** to compare multiple candidate models. Given the **class imbalance**, **F1-score** was chosen as the primary selection metric. From this process, the top-performing models were selected from two categories:

- **Tree-based models** (robust to outliers and feature scaling).

- **Linear models** (require scaling and transformations but offer higher interpretability). The best candidate from each category was then further trained and evaluated for final model selection.

Implementation (Brief)

Model Evaluation

The final models were benchmarked using:

- **Confusion Matrix** to assess prediction errors.
- **Classification Report** with precision, recall, F1-score, and support.
- **ROC Curve and AUC** to evaluate separability.
- **Precision-Recall Curve** to measure performance on the minority (default) class.

Implementation (Brief)

Advanced Model Analytics

Feature Importance Analysis

- Extracted feature importances from the best LightGBM model.
- Visualized the top 15 features using a bar plot.
- Identified the 10 most influential predictors contributing to default risk.

Implementation (Brief)

Validation Curve and Overfitting Check

- Performed a validation curve analysis with varying numbers of estimators.
- Compared training vs validation F1-scores to assess bias–variance tradeoff.
- Determined the optimal number of estimators by maximizing validation F1.
- Explicitly checked for overfitting by comparing gaps between training and validation scores.

Implementation (Brief)

Feature Stability Analysis (PSI)

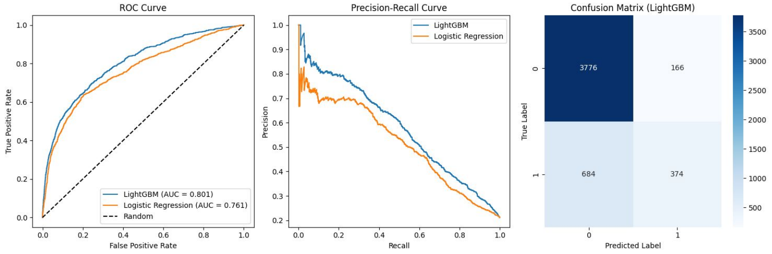
- Calculated the Population Stability Index (PSI) to measure how feature distributions change over time.
- Flagged features with PSI > 0.1 as potentially unstable.
- Highlighted the most stable features, ensuring consistency for long-term deployment.

Implementation (Brief)

Model Stability Analysis (Time-series Cross-validation)

- Used **TimeSeriesSplit** to evaluate performance stability across different temporal folds.
- Measured Accuracy, Precision, Recall, F1, and AUC for each fold.
- Calculated the **Coefficient of Variation (CV)** for each metric:
  - $CV < 0.1$ → Stable
  - $0.1 \leq CV < 0.2$ → Moderately stable
  - $CV \geq 0.2$ → Unstable
- Visualized metric variations across folds to verify temporal stability.

Results & Conclusion




```
=== LightGBM Classification Report ===
              precision    recall  f1-score   support

           0       0.85      0.96      0.90      3942
           1       0.69      0.35      0.47      1058

    accuracy                           0.83      5000
   macro avg       0.77      0.66      0.68      5000
weighted avg       0.81      0.83      0.81      5000


=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

           0       0.82      0.97      0.89      3942
           1       0.69      0.23      0.34      1058

    accuracy                           0.82      5000
   macro avg       0.76      0.60      0.62      5000
weighted avg       0.80      0.82      0.78      5000


=== Final Model Comparison ===
                 Model  Accuracy  Precision  Recall  F1 Score  ROC AUC
0             LightGBM    0.8300     0.6926  0.3535    0.4681   0.8009
1  Logistic Regression    0.8154     0.6923  0.2297    0.3449   0.7615
```

Model Comparison:

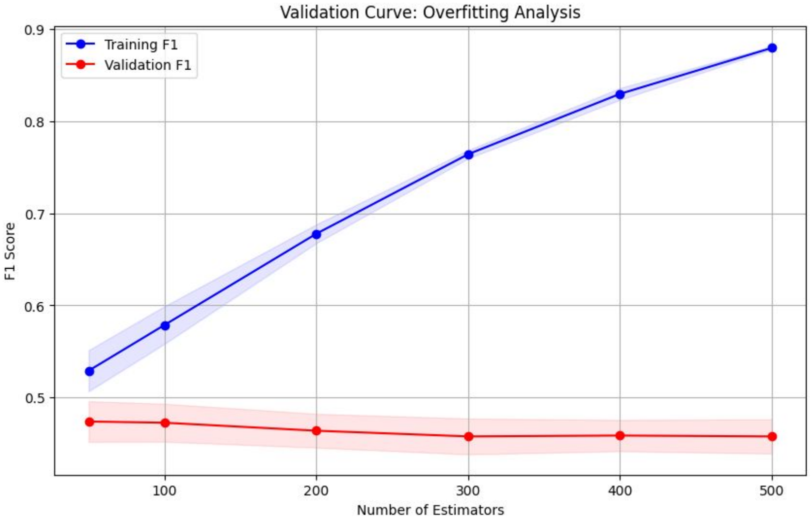
- LightGBM achieved higher **F1-score (0.47 vs. 0.34)** and **AUC (0.80 vs. 0.76)** than Logistic Regression, making it the stronger model for this imbalanced classification task.
- Confusion matrix shows LightGBM maintains high specificity (96% TN) but recall for the minority class remains modest (35%).

	fold	accuracy	precision	recall	f1	auc
0	1	0.8010	0.6333	0.3238	0.4285	0.7525
1	2	0.8058	0.6227	0.3336	0.4345	0.7489
2	3	0.8266	0.7295	0.4225	0.5351	0.7983
3	4	0.8376	0.6992	0.3578	0.4734	0.7812
4	5	0.8300	0.6926	0.3535	0.4681	0.8009

```
=== Coefficient of Variation (lower = more stable) ===
Accuracy: 0.0194 (Stable)
Precision: 0.0676 (Stable)
Recall: 0.1076 (Moderately Stable)
F1: 0.0908 (Stable)
Auc: 0.0317 (Stable)
```

Cross-Validation:

- Performance across folds was consistent: Accuracy (~0.83), AUC (~0.80), F1 (~0.47).
- Coefficient of variation was low, confirming stable generalization.



Overfitting Check:

- Validation curve shows training F1 keeps rising, but validation F1 levels off and does not improve, meaning larger ensembles risk overfitting without real gains.

Conclusion

LightGBM outperformed Logistic Regression with higher F1 and AUC, making it the best choice for this imbalanced task. The model is stable across folds and interpretable via feature importance. While recall for defaulters is modest, the pipeline provides a robust and practical solution for credit risk prediction, with room for improvement using resampling or cost-sensitive methods.

Future directions

Extend benchmarks to more ensemble models (e.g., CatBoost, XGBoost) and deep learning approaches.

Apply more advanced techniques across preprocessing, modeling, optimization, and monitoring to improve performance

Q&A