Home Credit

Credit Risk Prediction Model

Lingyi Li 206543047

Background

Home Credit is an international consumer finance provider founded in 1997 in the Czech Republic. It offers accessible financial services, primarily targeting individuals with limited or no credit history, operating across multiple European and Asian countries.

Default Prediction

Predicting borrower defaults is crucial for financial institutions to mitigate credit risk and ensure timely loan recovery. This project aims to utilize historical loan application data to develop machine learning models that forecast clients' default risks.

Dataset Overview

Data Source:

- This dataset is designed for loan default prediction and is sourced from Kaggle. It encompasses client information derived from both internal Home Credit data and external sources such as tax registries and credit bureaus.

Dataset Size:

- The dataset contains approximately 26GB of data spread across 17 tables.
- The dataset contains 466 features, featuring information on clients' demographics, credit history, loan applications, and payment behavior.

Variable	Description
actualdpd_943P	Days Past Due (DPD) of previous contract (actual).
actualdpdtolerance_344P	DPD of client with tolerance.
addres_district_368M	District of the person's address.
addres_role_871L	Role of person's address.
addres_zip_823M	Zip code of the address.
amount_1115A	Credit amount of the active contract provided by the credit bureau.

. . . . . .

Dataset Structure

All tables have a "depth" value to indicate the category of information they contain.

The base table store the basic information about the observation and case_id. This is a unique identification of every observation and I use it to join the other tables to the base table.

Depth 0	These are static features directly tied to a specific case_id
Depth 1	Each case_id has associated historical records, indexed by num_group1.
Depth 2	Each case_id has associated historical records, indexed by num_group1 and num_group2

PS. All case_id in base table is unique.

Data Preprocessing and EDA

Data Preprocessing

Step 1: Join the tables using the case_id with the base table.

Step 2: For tables with depth = 1 and depth = 2, aggregate the features by case_id, calculating the min, max, average, and count. Alternatively, perform aggregations based on the feature data type, such as date or string aggregation.

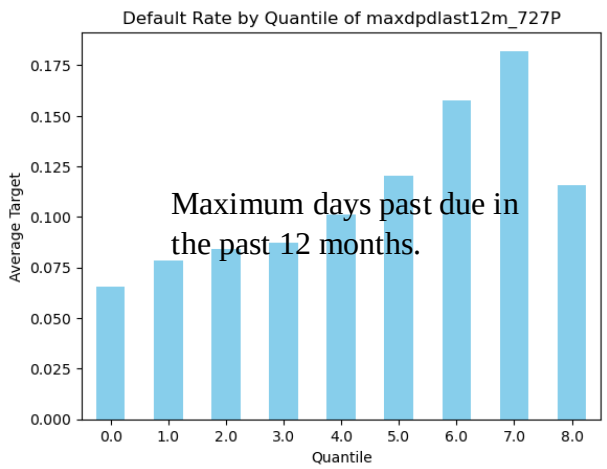
Step 3: Drops columns with more than 100 categories or just 1 and columns with more than 80% of null values.

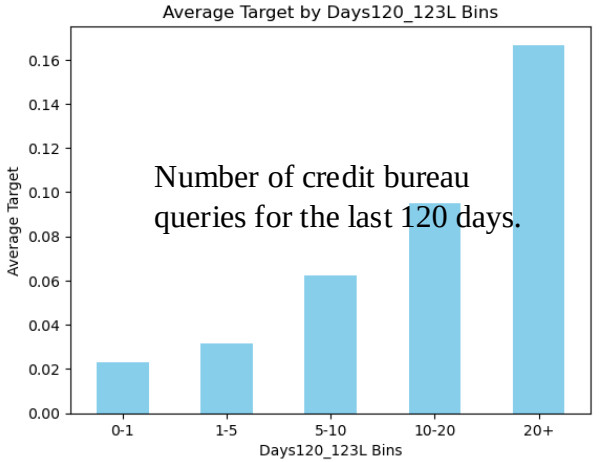
Due to the large volume of data, I used Polars instead of Pandas here.

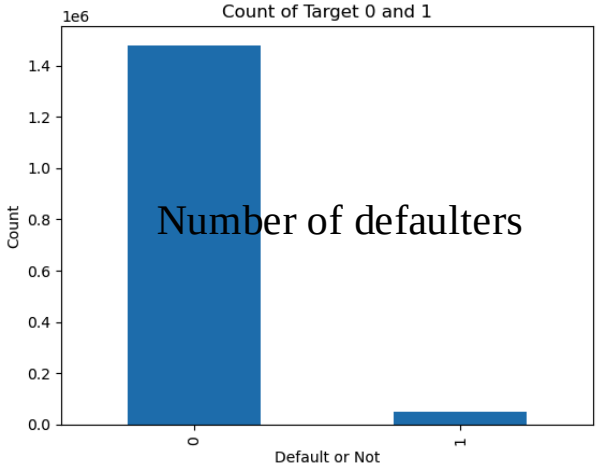
1526659 rows
369 columns

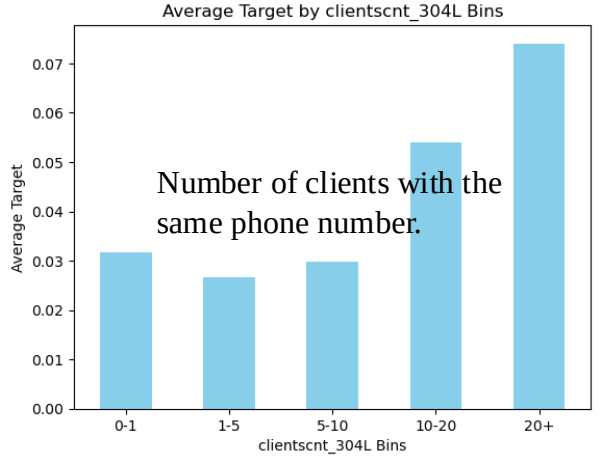
80% for Training

20% for Testing









Due to the large number of features, it‘s not feasible to check the distribution and select them one by one, so I directly perform aggregation to generate the features.

Feature Engineering

1. Baseline Model

I first conducted experiments using all features and ran models such as Random Forest, CatBoost, XGBoost, and Logistic Regression. I found that **XGBoost** performed the best, with an **AUC of 0.8**.

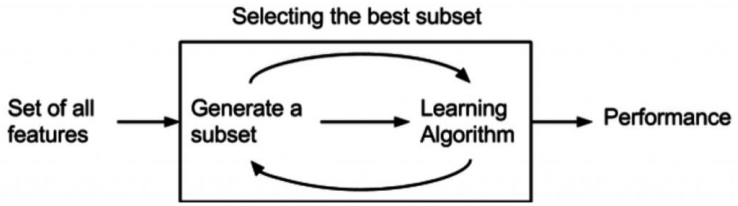
2. Feature Deviation

To further improve the model's performance, I decided to generate more useful features. My approach consists of two steps and got 965 features finally.

1. I examined the feature importance from XGBoost and found that the top 10 most important features were mostly related to historical loan defaults, followed by features of other types.
2. I then created new features by performing multiplication and division between the top 10 features and those ranked 10 to 50 in importance.

3. Feature Selection

However, after putting so many features into the model, I found that the performance was not good, possibly due to the large number of irrelevant features, making the model too complex. Therefore, I chose to use Lasso regression for feature selection, and ultimately selected 468 features.



Lasso regression is used for feature selection because it applies L1 regularization, which shrinks the coefficients of less important features to zero. This helps reduce model complexity, prevent overfitting, and improve generalization by keeping only the most relevant features. Additionally, Lasso handles multicollinearity by selecting one feature from a group of correlated ones.

Model Building

Model Choosing

Tree-based models are ideal for default prediction because they can handle complex, non-linear relationships between features and the target variable. They also provide valuable insights into feature importance, are robust to outliers, and do not require feature scaling or imputation for missing data. Additionally, these models are interpretable, making them useful for understanding and explaining default risk.

Specifically, I chose **XGBoost, CatBoost, and LightGBM** here for the experimentations.

Model Blending

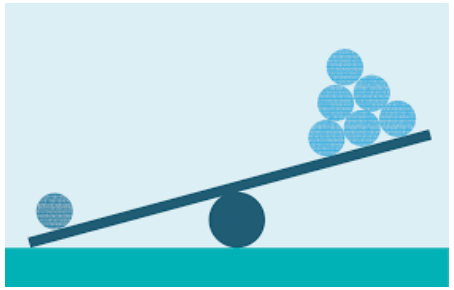
Model blending, specifically using **soft voting**, combines multiple models to make predictions by averaging their probabilities. This approach improves accuracy by leveraging the strengths of different models and reducing the impact of individual model biases.

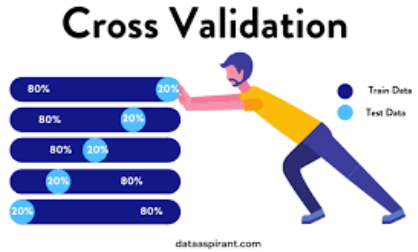
Cross-Validation

To reduce over-fitting issue, I used cross-validation. I chose StratifiedKFold cross-validation to ensure that each fold of the dataset maintains the same proportion of class labels as the original dataset. This is particularly beneficial for imbalanced datasets, as it ensures that both the training and validation sets have a similar distribution of classes, leading to more reliable and consistent model performance evaluation.

Data Imbalance

I focused on the class imbalance issue by adjusting parameters such as scale_pos_weight and class_weight within the model.





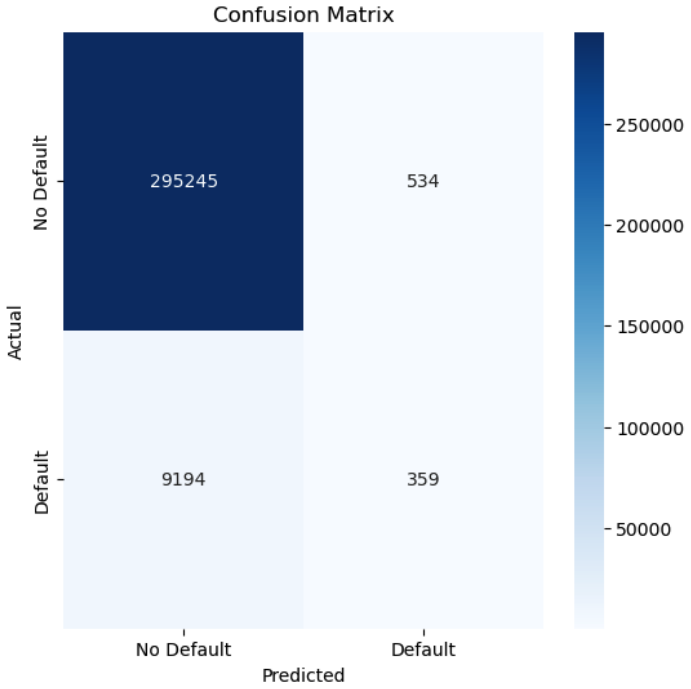
Model Evaluation

Result for the final model:

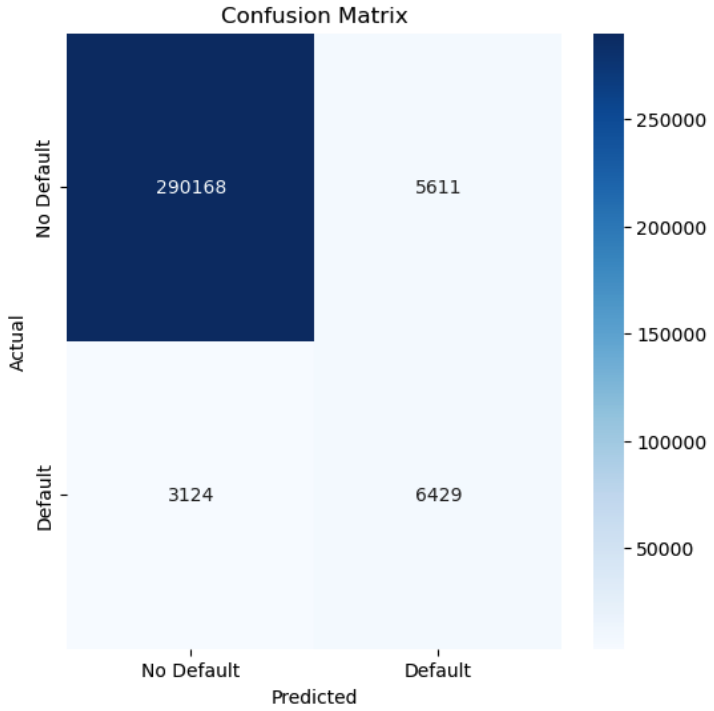
AUC = 0.8439

Recall = 0.67, Precision = 0.53

Not a very good result, but they are a big improvement over the baseline.



Baseline



New Model

Thank you !