Home Credit Default Risk

Group members:
MA Rongyue 20826086
NI Xiaohan 20825846
Peng Junkai 20756772
YE Mengxiang 20799762

September 19th, 2021

Abstract

For our project, we choose Light Gradient Boosting Machine (Light GBM) as the primary model, a distributed gradient boosting framework for machine learning. It is based on decision tree algorithms and used for ranking, classification and other machine learning tasks. The reason why we choose this model is Light GBM's capacity of handling large-scale data, faster training speed and higher efficiency. After we checked the input data, we found out there are numerical data as well as categorical values. Moreover, lots of numerical features have abnormal values which need to be adjusted.

First of all, we drop those features whose missing ratio is greater than 60%. It is inappropriate to refill those data if half of it is lost.

Second, we have to deal with the abnormal data in each column, before we replace them as NaN, we create a new column which states the abnormal in bool type. Furthermore, we want to change the days counting time into years counting time, to make the data value seem better.

In addition, we also use manual feature engineering based on our financial knowledge.

Next, we choose the Label_Encoder function in the 'sklearn' package to transform those categorical features. The 'light_bgm' package can automatically train the categorical features in our model. In order to improve our prediction power, we use k-fold CV to avoid overfitting.

As an additional model assessment, we use the 'light_bgm' package to plot out our model's feature importance to exhibit which features provide the most considerable prediction power to our model. On the basis of that graph, we can make further interpretations about this model.

Going forward, we can use other data files to expand our initial training data set and improve the final score.

1 Introduction

1.1 Background

Home Credit, a consumer finance provider, aims at providing customers with financial services. The goal of this competition is to predict the repayment abilities of clients and estimate the loan default probability of customers by using machine learning algorithms through the characteristics of loan users, internal and external historical loan records and other information.

1.2 A Glimpse of Data

This data, provided by *Home Credit* which tries to provide broader ways of loans to the unbanked population, can be used by *Home Credit* to predict the repayment abilities of their clients.

The data is derived from seven different sources, such as *application_train/application_test, bureau, bureau_balance, credit_card balance, installments_payment, POS_CASH_balance and previous_application*. In our analysis, we strive to use two main data to do the project and get the final results, one main file for training called the application training data(with target), and the other main file for testing called the application testing data(without the target).

The application training and testing data include the target variable (whether the customer defaults or not - 0 / 1 variable), the customer's loan application information (loan type, total loan, annuity), the customer's basic information (gender, age, family, education, occupation, industry, residence), the customer's financial information (annual income, room / car situation), the resources provided during the application, etc.

Besides, we could test the data shape and find that the training data has 122 variables including the TARGET which we need to predict, and 307511 observations (each one a separate loan). What's more, we could also find that the testing data has 121 variables without the TARGET column, and 48744 observations.

Then, in the following parts, we will get into exploring the data. As mentioned previously, we will stick to the main data sources and simple models which we can build upon in future work to predict the repayment abilities of clients.

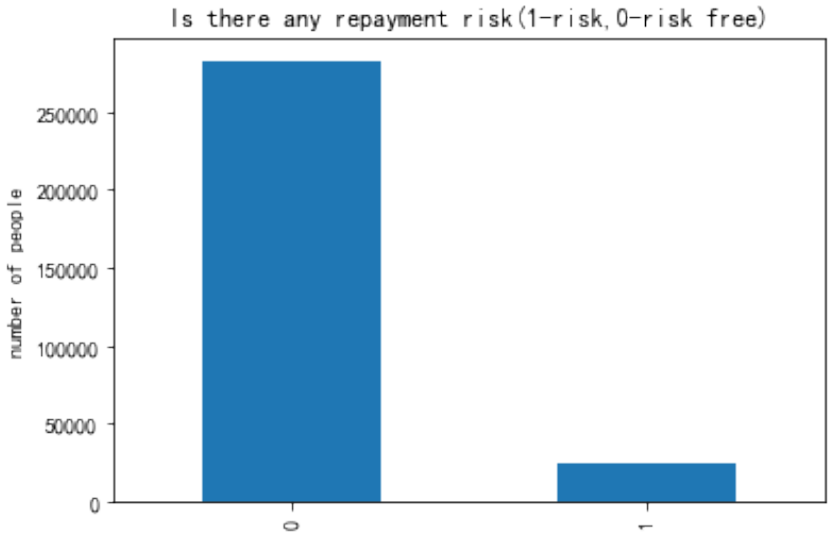
2 Exploratory Data Analysis

The objective of this part: Exploratory Data Analysis (EDA) is an open-ended process where we calculate statistics and make figures to find trends, anomalies, patterns, or relationships within the data. The goal of EDA is to learn what our data can tell us. It generally starts with a high-level overview, then narrows into specific areas as we find intriguing areas of the data. The findings may be interesting in their own right, or they can be used to inform our modeling choices, such as by helping us decide which features to use.

2.1 Distribution of target variable

The target variable shows whether the loan was repaid or not. Then we draw the overall risk histogram according to the number of TARGET under different categories to get the distribution of loan repayment in the training set.

When the target equals to 0, it means that the clients have no difficulty to repay the loan or installments in a given period of time. When the target equals to 1, it means that the clients have problems in repayment. According to the graph, we could see that there are far more loans that were repaid on time than loans that were not repaid. There are approximately 24,000 applicants(about 8% of the total loans) in the training data involved clients who don't have abilities to repay the loan.



2.2 Missing values and outliers examination

Next we can examine the percentage of missing values in each column, and drop all features/columns which have more than 60% missing data. We could see that in the application training dataset, there are 67 columns having missing values, but at first we just drop 17 columns whose missing ratios are more than 60%.

2.2.1 Text columns analysis & label encoding

First, deal with unknown data on feature gender. In the column of gender, there exists an abnormal value 'XNA', and we replace it with 'nan'.

Then, Before we go any further, we need to deal with pesky categorical variables, aiming at transforming categorical variables into numerical ones. Label encoding is an effective way to carry out the process. Label encoding is the process of assigning each unique category in a categorical variable with an integer, and no new columns are created.

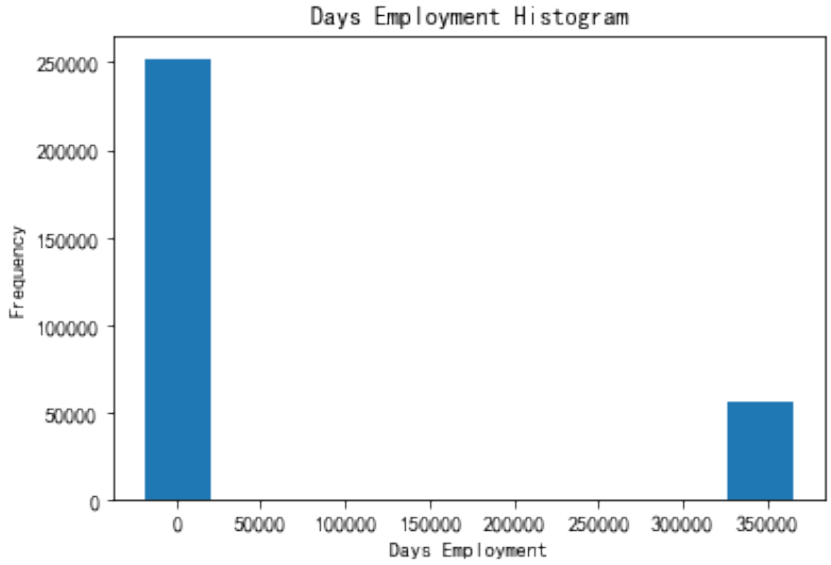
Normally, we use 'label.fit' to fit the data of columns and then transform the categorical variables into numerical ones, and then we could record how many variables are transformed into numerical ones. But if there exists an error, then we will drop the data. The result shows that 16 columns were label encoded.

2.2.2 Numeric columns analysis & abnormal

One problem we always want to be on the lookout for when doing EDA is anomalies within the data. These may be due to mis-typed numbers, errors in measuring equipment, or they could be valid but extreme measurements. The numbers in the DAYS_EMPLOYED and DAYS_BIRTH columns are negative because they are recorded relative to the current loan application. To see these stats in years, we can multiply by -1 and divide by 365 in both application training and testing cases.

From the graph below, we could see that the maximum value of days employed is nearly 1000 years. It is absolutely wrong! So, after transforming daily dates to years, we try to replace abnormal days employed with NaNand label a new feature state abnormal issue.

Finally, we could get the missing count and ratio as follows, and impute missing part by *fillna* method with strategy median.



	missing_count	missing_ratio
LANDAREA_MODE	182590	0.593767
LANDAREA_MEDI	182590	0.593767
LANDAREA_AVG	182590	0.593767
BASEMENTAREA_MODE	179943	0.585160
BASEMENTAREA_MEDI	179943	0.585160
BASEMENTAREA_AVG	179943	0.585160
EXT_SOURCE_1	173378	0.563811
NONLIVINGAREA_MODE	169682	0.551792
NONLIVINGAREA_MEDI	169682	0.551792
NONLIVINGAREA_AVG	169682	0.551792
ELEVATORS_AVG	163891	0.532960
ELEVATORS_MEDI	163891	0.532960

3 Feature engineering

In the section of Feature Engineering, we create new features to the existing data set.

3.1 Domain knowledge feature

Firstly, we try to incorporate our financial knowledge by incorporating some domain knowledge features. we create ratios to derive more valuable information from simple numerical data. Through selecting and comparing, we finally choose the following four ratios which are intuitive. We believe this is a good first step to feature engineering followed by Light GBM.

- ***Term***

  The indicator shows how many months are left until the full repayment. By dividing the credit amount of the loan by the loan annuity, we can get the duration term (TERM = AMT_CREDIT / AMT_ANNUITY)

  The indicator is relatively straightforward as there is a high dependence between the loan term and default probability. The logic behind is that there will be more unexpected factors when the time duration is longer.

- $\%$ ***of the total loan amount relative to the applicant's income***

  The indicator shows what percentage of income is used to repay the loan (CREDIT_INCOME_PERCENT = AMT_CREDIT / AMT_INCOME_TOTAL).

  We expect that the higher the indicator, the higher the probability of default.

- $\%$ ***of the monthly payment relative to the applicant's income***

  The indicator (ANNUITY_INCOME_PERCENT = AMT_ANNUITY / AMT_INCOME_TOTAL) is similar to the 2^{nd} indicator but introducing the loan annuity to make our analysis more comprehensive.

  We expect that the higher the indicator, the higher the probability of default.

- $\%$ ***of employment duration relative to age***

  YEARS_EMPLOYED_PERCENT = YEARS_EMPLOYED / YEARS_BIRTH

  As the pure year of employment can not indicate the capability of repaying the debt, we divide it by age, and the ratio can intuitively suggest the applicant's working competence.

3.2 Missing data pattern

In the above, we have dropped the columns with more than 60% missing data. However, there are still many columns having plenty of missing data that is hard to predict by simply median or mean. In our perspective, we can derive some insights from the missing data pattern as people are more likely to skip the optional field when their answers are negative.

Therefore, we regard the missing data pattern as a new indicator. We create a new binary indicator called "Incomplete" which will be flagged when there are more and equal to 40 blanks in their applications.

```
#create a new feature states the completeness of application file for each cilent
df_train['incomplete'] = 1
df_train.loc[df_train.isnull().sum(axis=1) < 40, 'incomplete'] = 0
df_test['incomplete'] = 1
df_test.loc[df_test.isnull().sum(axis=1) < 40, 'incomplete'] = 0
```

For numerical data columns, we impute the missing values by *fillna* method with strategy median.

```
#impute missing part by fillna method with strategy median
def refill(data):
    for col in data.columns.values.tolist():
        data[col] = data[col].fillna(data[col].median())
    return data
df_train = refill(df_train)
df_test = refill(df_test)
```

4 Model training

At first, we did not use Cross-Validation and found a big gap between our AUC score and the Leader board score. After careful inspection, we are aware of the issue of Overfitting. Then, we introduce the Cross-Validation and adjust the early stopping parameters.

4.1 Cross-Validation

By using 5-fold Cross-Validation, we predict with each fold, and taking the average can make LB (Leader board) score more similar to valid score.

```
#using 5-folds cv to improve the modeling
#using package to randomly split data into 5 piece
folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=6)
preds = np.zeros(X.shape[0])
predictions = np.zeros(predition_X.shape[0])

#perform the cv
valid_score = 0
for n_fold, (train_idx, eval_idx) in enumerate(folds.split(X, y)):
    train_x, train_y = X.iloc[train_idx], y[train_idx]
    eval_x, eval_y = X.iloc[eval_idx], y[eval_idx]

    train_data = lgb.Dataset(data=train_x, label=train_y,categorical_feature=label_list)
    eval_data = lgb.Dataset(data=eval_x, label=eval_y)

    params = {'application':'binary','num_iterations':4000, 'learning_rate':0.05, 'num_leaves':24,
              'feature_fraction':0.8, 'bagging_fraction':0.9,
              'lambda_l1':0.1, 'lambda_l2':0.1, 'min_split_gain':0.01, 'min_data_in_leaf':20,
              'early_stopping_round':100, 'max_depth':7,
              'min_child_weight':40, 'metric':'auc','verbose' : -1}
```

4.2 Light Gradient Boosting Machine

The objective of feature selection is to remove redundant variables to optimize our data set.

There are quantities of methods of feature selection, and we are running Light GBM. Before implementing Light GBM, we have 109 features.

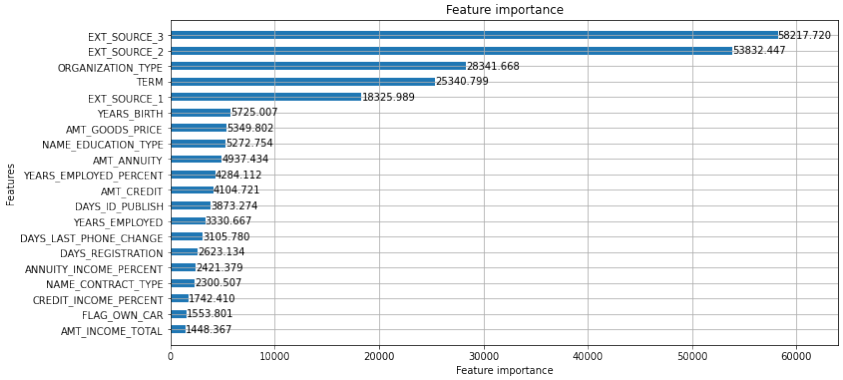
Meanwhile, we also set early stopping parameters to avoid overfitting. Early stopping means the training will stop when the performance of the validation is not improving after the last early stopping round. For our model, we set num_iterations as 4000 and early_stopping_round as 100.

5 Assessment

Please find the output of AUC score and Valid score on the next page.

We then check the feature importance returned from Light GBM by plotting the feature importance graph.

Going forward, we can use other data files to expand our initial training data set and improve the final score.



```
Training until validation scores don't improve for 100 rounds
[100]   training's auc: 0.773206        valid_1's auc: 0.755741
[200]   training's auc: 0.790393        valid_1's auc: 0.762617
[300]   training's auc: 0.802024        valid_1's auc: 0.764401
[400]   training's auc: 0.811519        valid_1's auc: 0.765024
[500]   training's auc: 0.820104        valid_1's auc: 0.765168
Early stopping, best iteration is:
[463]   training's auc: 0.817038        valid_1's auc: 0.765461
AUC of Fold  1  : 0.765
Training until validation scores don't improve for 100 rounds
[100]   training's auc: 0.772858        valid_1's auc: 0.754742
[200]   training's auc: 0.79063 valid_1's auc: 0.761016
[300]   training's auc: 0.80243 valid_1's auc: 0.762504
[400]   training's auc: 0.812079        valid_1's auc: 0.763002
[500]   training's auc: 0.820593        valid_1's auc: 0.762732
Early stopping, best iteration is:
[434]   training's auc: 0.815142        valid_1's auc: 0.763086
AUC of Fold  2  : 0.763
Training until validation scores don't improve for 100 rounds
[100]   training's auc: 0.773175        valid_1's auc: 0.752751
[200]   training's auc: 0.790481        valid_1's auc: 0.759247
[300]   training's auc: 0.802212        valid_1's auc: 0.760811
[400]   training's auc: 0.812478        valid_1's auc: 0.761707
[500]   training's auc: 0.821717        valid_1's auc: 0.761866
[600]   training's auc: 0.829628        valid_1's auc: 0.762083
[700]   training's auc: 0.836512        valid_1's auc: 0.761985
Early stopping, best iteration is:
[622]   training's auc: 0.831356        valid_1's auc: 0.76222
AUC of Fold  3  : 0.762


Training until validation scores don't improve for 100 rounds
[100]   training's auc: 0.772649        valid_1's auc: 0.755845
[200]   training's auc: 0.78968 valid_1's auc: 0.76218
[300]   training's auc: 0.801874        valid_1's auc: 0.764186
[400]   training's auc: 0.812408        valid_1's auc: 0.765011
[500]   training's auc: 0.821347        valid_1's auc: 0.765258
[600]   training's auc: 0.829356        valid_1's auc: 0.765169
Early stopping, best iteration is:
[586]   training's auc: 0.828143        valid_1's auc: 0.765307
AUC of Fold  4  : 0.765
Training until validation scores don't improve for 100 rounds
[100]   training's auc: 0.772192        valid_1's auc: 0.757425
[200]   training's auc: 0.790299        valid_1's auc: 0.763205
[300]   training's auc: 0.802804        valid_1's auc: 0.765017
[400]   training's auc: 0.81283 valid_1's auc: 0.765878
[500]   training's auc: 0.821932        valid_1's auc: 0.765748
Early stopping, best iteration is:
[412]   training's auc: 0.813852        valid_1's auc: 0.765925
AUC of Fold  5  : 0.766
valid score: 0.7644
```

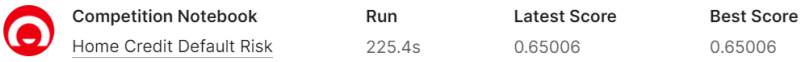
References

[1] oskird. (2018, June 21). EDA + baseline model using application. Kaggle. Retrieved September 19, 2021, from https://www.kaggle.com/sz8416/eda-baseline-model-using-application?scriptVersionId=4207867amp;cellId=1.

[2] Koehrsen, W. (2018, July 31). Introduction to manual feature engineering. Kaggle. Retrieved September 19, 2021, from https://www.kaggle.com/willkoehrsen/introduction-to-manual-feature-engineering?scriptVersionId=4852087amp;cellId=1.

Kaggle score

group link: https://www.kaggle.com/junkaipeng/mafs6010z-warmup-project



Group work Contribution

We all actively participated in the initial discussion and model selection process. For the Feature Engineering part, all groupmates have come up with different thoughts. At last, we have incorporated different ideas from different people.

After the outline is settled, PENG Junkai and NI Xiaohan are mainly responsible for coding, and MA Rongyue and YE Mengxiang are mainly responsible for report writing. But there is a lot of shared work.