MSBD5013 MIDTERM PROJECT: HOME CREDIT DEFAULT RISK

TECHNICAL REPORT

Shihan BU
Big Data Technology
Department of Engineering
The Hong Kong University of Science and Technology
Hong Kong
sbuaa@connect.ust.hk

Yilin LI
Big Data Technology
Department of Engineering
The Hong Kong University of Science and Technology
Hong Kong
ylime@connect.ust.hk

Yuxin YANG
Big Data Technology
Department of Engineering
The Hong Kong University of Science and Technology
Hong Kong
yyanget@connect.ust.hk

March 25, 2022

ABSTRACT

Home Credit is a consumer finance provider which focuses on providing a positive borrowing experience for underserved population with little credit history. It holds a competition on Kaggle in hope that participants can leverage a variety of alternative data to help predict their customers' repayment abilities.We explored the given data with both statistical and financial knowledge and build a leaf-wise GBDTLightGBM) model to classify whether a specific data example is default or not.

Keywords Home Credit · Statistics · Finance · LightGBM

1 Introduction

When it comes to loan applications, credit history plays an irreplaceable part. This means that people with little to no credit history struggles a lot when trying to apply for loans and can potentially be exposed to untrustworthy lenders. In order to tackle this problem and provide loans to those in need in a timely manner, we would like to build a model that accurately predicts an applicant's repayment abilities by utilizing various alternative data.

2 Methodology

Before we dive into the models and make predictions, we need to first understand our data and process it to make it more usable. We would like to first perform some exploratory data analysis on each table to understand the content and quality of our data. And then, starting from the 2 main tables as the baseline, experiment with incorporating the remaining 6 tables to see if they add value.

2.1 Data Set

The data set that was provided consists of 8 data tables and a descriptive table. Of the 8 data tables, application_{train|test}.csv(hightlighted in green) are the main tables. There are 307,511 rows and 122 columns (with target) in the train table where each row represents one loan. And there are 48,744 rows and 121 columns in the test table(without target). The remaining 6 supporting tables contains various bureaucratic and transaction information on the loan applicants and their previous loans. These tables are linked to the main tables directly and indirectly through keys.

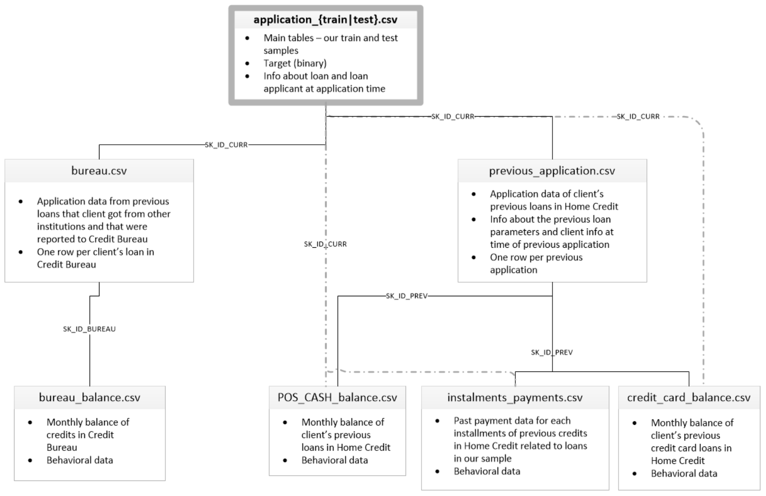

application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
SK_ID_BUREAU
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
previous_application.csv
Application data of client's previous loans in Home Credit
Info about the previous loan parameters and client info at time of previous application
One row per previous application
SK_ID_PREV
POS_CASH_balance.csv
Monthly balance of client's previous loans in Home Credit
Behavioral data
instalments_payments.csv
Past payment data for each installments of previous credits in Home Credit related to loans in our sample
Behavioral data
credit_card_balance.csv
Monthly balance of client's previous credit card loans in Home Credit
Behavioral data

Figure 1: Data Structure.

2.2 Data Processing

2.2.1 Exploratory Data Analysis (EDA)

Target Distribution Since our goal is to predict whether a client will default on the loan or not, we would like to first check out the distribution of default in our training set. And the result shows that our data is imbalanced, only a small portion of the loans are default.

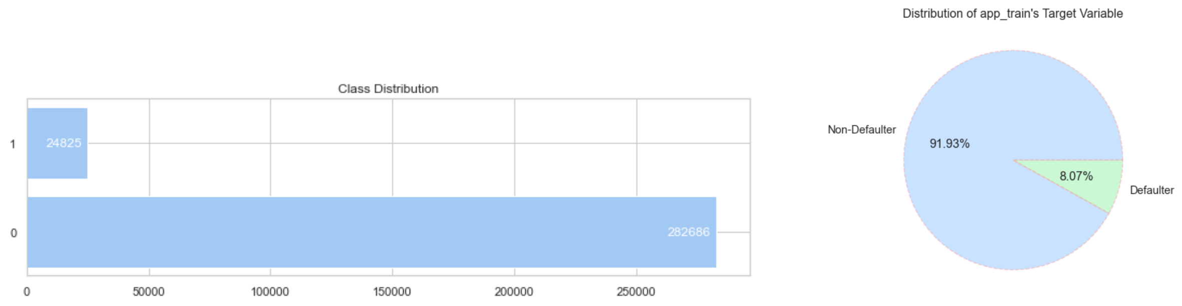


Figure 2: Missing Values in Train Set

Data and Feature Types The training and testing set contains 120 features for each loan. These are a mix of numerical and categorical values. The numerical values consists of float and integer data, and the categorical values consists of both object and integer data. Take FLAG_MOBIL as an example, 1 means that the client provided a mobile phone number and 0 means the opposite. Therefore we need to be careful with integer values since some of them are used as categorical labels.

Data Type	Number of Features
float64	65
int64	39
object	16

Feature Type	Number of Features
Numerical	70
Categorical	50

Table 1: Data and Feature Types

Missing Values We find that some features in our data has the problem of missing values. As shown in Figure 3 below, some features' missing percentage in the training set reached as high as close to 70% while others are less than 0.1%.

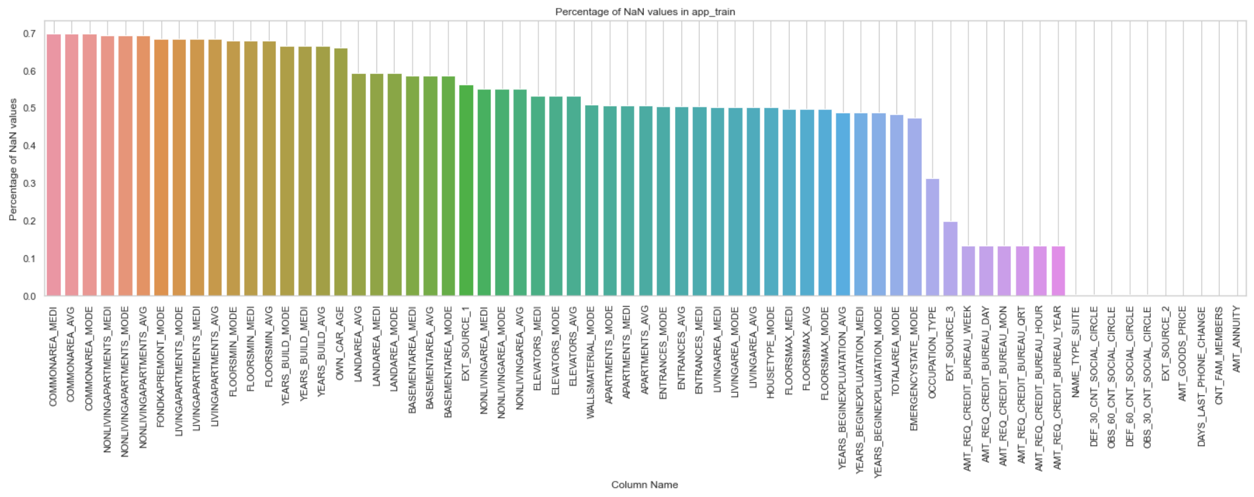
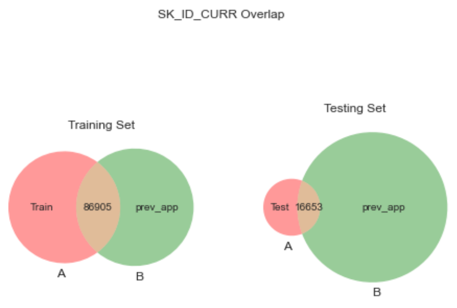
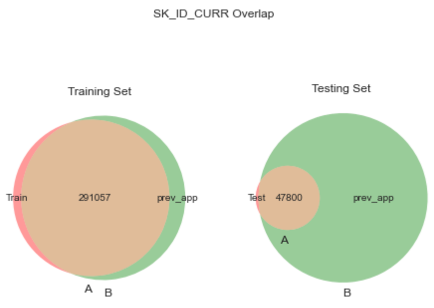


Figure 3: Missing Values in Train Set

Loan Instance Overlap The 6 supporting tables provide us with additional information on the loan applicants and their previous loans. However, the loan instances in these supporting tables are not a complete match with the main train/test tables. Figure 4 shows that table credit_card_balance.csv only shares a small percentage of loan instance overlap with the train/test set while previous_application.csv shares much bigger percentage. We would like to have as much relevant information for our final prediction as possible, therefore tables with higher levels of overlap are given priorities in analysis.



(a) credit_card_balance.csv



(b) previous_application.csv

Figure 4: Loan Instance Overlap

2.2.2 Data Cleaning

Illogical Values Some values in the tables do not make any sense. For example, in previous_application.csv, some rows of feature DAYS_FIRST_DUE have value 365,243 which is equivalent to 1,000 years. We've identified these values to the best of our knowledge, and treated them as missing values.

Missing Values Filling There are a number of ways to deal with missing values. And the most straight forward way is to remove the feature entirely when its missing percentage goes over the threshold. However, we decided to not directly delete those instances with NaN value, since the missing rate in our data is relatively high. Instead we decide to fill categorical features with string "Missing" as an additional category, and fill numerical features with mean values.

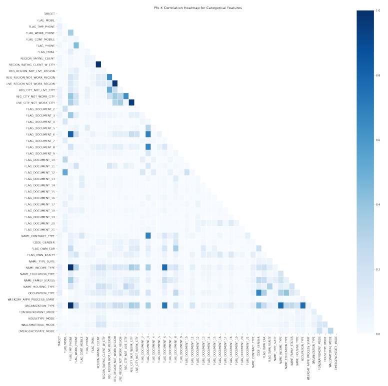
Normalization We used the StandardScaler() function from sklearn package to center the mean of train data to 0 and set variance to 1.

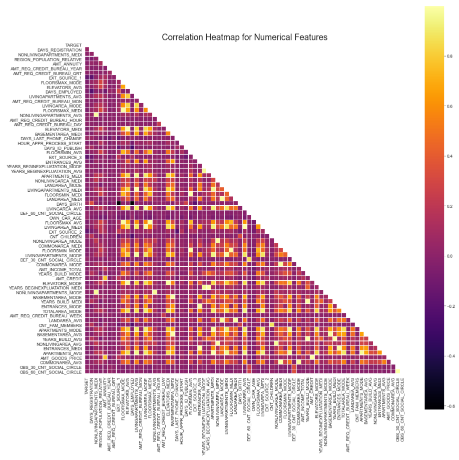
2.3 Feature Engineering

2.3.1 Dimension Reduction

Categorical Features Selection We choose ϕ_k as the correlation coefficient for selecting categorical features. Compared with other existing coefficients ϕ_k has the advantage of capturing non-linear dependency and working consistently between categorical, ordinal and interval variables. We computed the correlation matrix for all feature including target and ranked the features from highest to lowest. We selected the top 12 categorical features into our model.

Numerical Features Selection We calculated the Pearson correlation between all numerical features and the target. Then we ranked the numerical features from highest correlation with the target to lowest. The top 6 numerical features were selected into our model.


(a) ϕ_k Correlation for Categorical Features



(b) Pearson Correlation for Numerical Features

2.3.2 Feature Encoding

Since categorical features cannot be directly processed, we need to encode them so that they can be fed into our model.

Bi-classes Categorical Features For this type of feature, we use label encoder to turn them into 0/1 numbers.

Multi-classes Categorical Features For this type of feature, we created dummy variables and one-hot coded the categorical feature.

2.3.3 Feature Enrichment

Based on domain knowledge, we constructed 8 new numerical features for our model.

2.4 Model Construction

As there are numerous categorical features in our fully aggregated data, a more flexible model should be utilized here to find the expected non-linear relationship between features and the final default probability target. Our model mainly focus on LightGBM (Ke et al. [2017]), which is an machine learning approach of Gradient Boost Decision Tree (GBDT) to perform home credit classification task. GBDT utilizes boosting as an ensemble method for combining multiple individual decision trees. The updating rule of GBDT model can be illustrated by the following formulas.

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}_{R_{jm}}(x), \quad \gamma_{jm} = \arg\min \gamma \sum_{x_i \in R_{jm}} L\left(y_i, F_{m-1}\left(x_i\right) + \gamma\right)$$

where J_m represents the number of its leaves, the input tree space will be partitioned into J_m disjoint regions and a separate optimal value γ_{jm} will be predicted for each region $R_{1m}, \ldots, R_{J_m m}$. In this greedy way, the model function

F_m can be updated step by step and optimized to convergence. Figure 6 shows the basic framework of constructing a GBDT model.

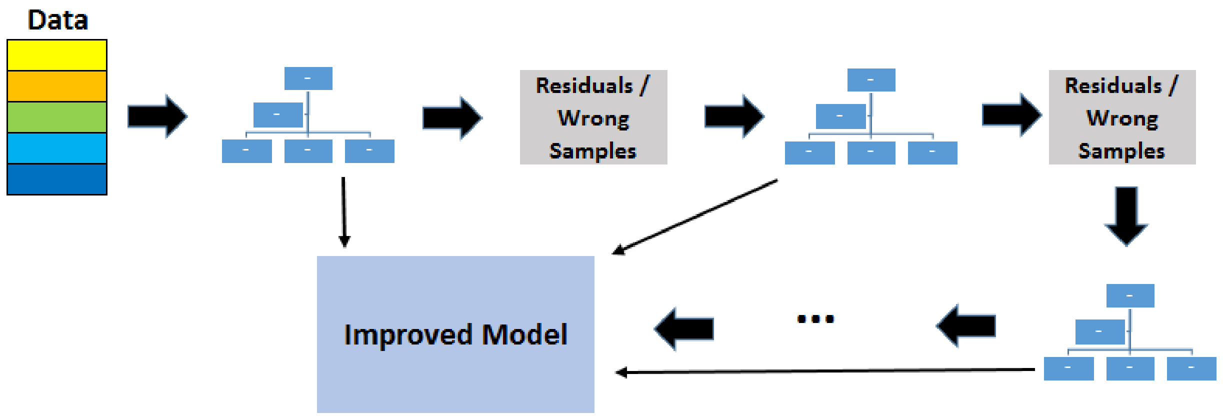


Figure 6: Gradient Boosting Decision Tree (GBDT) construction framework.

As shown in Figure 7, using leaf-wise growth strategy, LightGBM finds the leaf with the largest splitting gain from all the current leaves at a time, and then splits it, and so on. Therefore, compared with level-wise, the advantage of leaf-wise is that it can reduce more errors and get better accuracy with the same number of splits. Meanwhile, it retains large gradient samples while randomly retaining some small gradient samples, while amplifying the information gain from the small gradient samples.

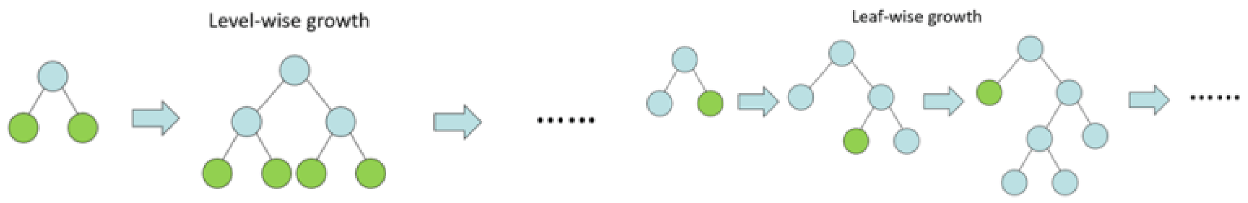


Figure 7: Process of LightGBM (a leaf-wise based GBDT model).

Considering several defects of leaf-wise that it may grow a deeper decision tree and produce overfitting, we added a maximum depth limit, early stopping round and regularization on top of leaf-wise to ensure high efficiency and prevent overfitting at the same time. For finding the best model, we validated our model with stratified k-fold cross validation method and optimized the best parameter of our LightGBM model with Bayesian optimizer.

3 Experiments

3.1 K-fold Validation

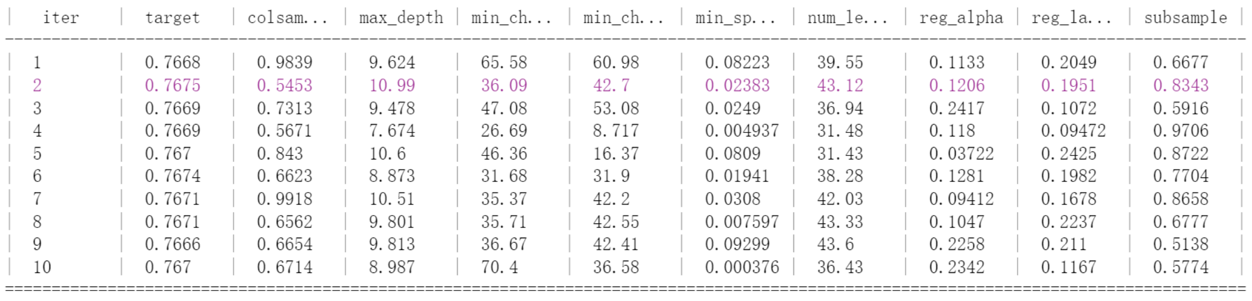
Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.
In this project, to avoid the problem of over-fitting caused by excessive number of variables. We use 5-fold cross-validation method to divide the original data-set into 5 groups. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, second fold is used as the testing set while the rest serve as the training set. Finally, we got 5 models and their evaluation metrics showed in the table 2. From the table, we can see that the training accuracy concentrated in 0.81-0.82, and validation in 0.76-0.78. Therefore, it can be concluded that there does not exist the problem of low AUC value caused by over-fitting problem in our process, since partial of the performance effect of the validation data-set and the prediction effect score of the training data set are almost at the same level. To change a word, Although the number of features is relatively large, there does not exist the over-fitting problem in our prediction model, which also proves that our prediction model has certain degree of stability.

Fold No.	Training	Validation
1	20.811937	0.76838
2	0.817108	0.765332
3	0.812574	0.768703
4	0.812276	0.761064
5	0.823392	0.771596

Table 2: Result of 5-fold validation

3.2 Model Tuning and Parameter settings

Although machine learning process is always a black box, we still want to find the set of model weights that best minimizes the loss function. Bayesian optimization is a machine learning based optimization algorithm used to find the parameters that globally optimizes a given black box function. It is an approach that is most useful for objective functions that are complex, noisy, or expensive to evaluate. Bayesian optimization is a powerful strategy for finding the extrema of objective functions that are expensive to evaluate. We had a simple understanding of the theoretical knowledge in this area that it used Gaussian process model and iterations.
In python, we used Bayes_opt library to implement the optimization. After 10 iterations, we found the best parameter setting, shows in purple color in the Figure 8.



```
|   iter    |  target   | colsam... | max_depth | min_ch... | min_ch... | min_sp... | num_le... | reg_alpha | reg_la... | subsample |
-------------------------------------------------------------------------------------------------------------------------------------
|  1        |  0.7668   |  0.9839   |  9.624    |  65.58    |  60.98    |  0.08223  |  39.55    |  0.1133   |  0.2049   |  0.6677   |
|  2        |  0.7675   |  0.5453   |  10.99    |  36.09    |  42.7     |  0.02383  |  43.12    |  0.1206   |  0.1951   |  0.8343   |
|  3        |  0.7669   |  0.7313   |  9.478    |  47.08    |  53.08    |  0.0249   |  36.94    |  0.2417   |  0.1072   |  0.5916   |
|  4        |  0.7669   |  0.5671   |  7.674    |  26.69    |  8.717    |  0.004937 |  31.48    |  0.118    |  0.09472  |  0.9706   |
|  5        |  0.767    |  0.843    |  10.6     |  46.36    |  16.37    |  0.0809   |  31.43    |  0.03722  |  0.2425   |  0.8722   |
|  6        |  0.7674   |  0.6623   |  8.873    |  31.68    |  31.9     |  0.01941  |  38.28    |  0.1281   |  0.1982   |  0.7704   |
|  7        |  0.7671   |  0.9918   |  10.51    |  35.37    |  42.2     |  0.0308   |  42.03    |  0.09412  |  0.1678   |  0.8658   |
|  8        |  0.7671   |  0.6562   |  9.801    |  35.71    |  42.55    |  0.007597 |  43.33    |  0.1047   |  0.2237   |  0.6777   |
|  9        |  0.7666   |  0.6654   |  9.813    |  36.67    |  42.41    |  0.09299  |  43.6     |  0.2258   |  0.211    |  0.5138   |
|  10       |  0.767    |  0.6714   |  8.987    |  70.4     |  36.58    |  0.000376 |  36.43    |  0.2342   |  0.1167   |  0.5774   |
=====================================================================================================================================
```

Figure 8: Hyperparameter Tuning

Although it only makes a small influence on the result, it shows that our model is stable, and it is also one part of our exploration.
Besides, we also early stop the iteration and find the best iteration during the training process to tune the model.

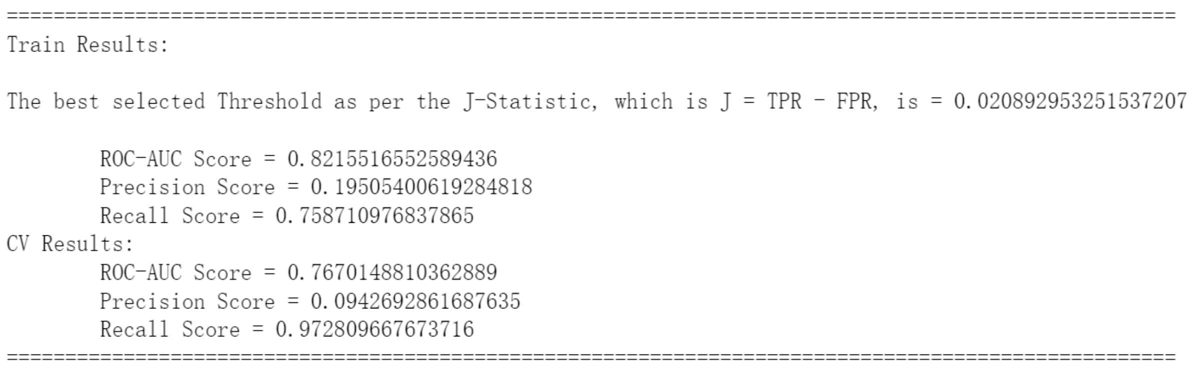
3.3 Evaluation Metrics

After using Bayesian Optimization to extract the best parameters for our model and using 5-fold validation to avoid over-fitting, we got the final evaluation metrics of the fitted model as Figure 9 shows.

The AUC value of the fitted model is 0.744. During our EDA, we found that our dataset is unbalanced. So, we should have high recall and low precision when doing machine learning, high recall relates to a low false negative rate, and low precision relates to a high false positive rate. From the result, we can see our model do fit these two requirements.

4 Results on Kaggle and Analysis

The final score we submitted to kaggle showed in figure 10.
Models for this problem are very important, features also influence the computation cost and accuracy. In order to explore whether our early characteristic engineering is effective, we plot the top 20 features that contribute more for the classification, which showed in figure 11.
From the figure, we are surprised to find that the top two features are depend on the financial domain knowledge, for example extsourcemean is to calculate the mean of some original important features contains extsource1, extsource2 and exsource3. It told us domain knowledge of specific field sometimes improve the accuracy and computation cost effectively.

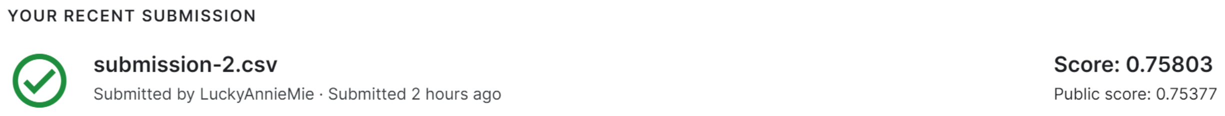


```
===================================================================================================
Train Results:

The best selected Threshold as per the J-Statistic, which is J = TPR - FPR, is = 0.020892953251537207

        ROC-AUC Score = 0.8215516552589436
        Precision Score = 0.19505400619284818
        Recall Score = 0.758710976837865
CV Results:
        ROC-AUC Score = 0.7670148810362889
        Precision Score = 0.0942692861687635
        Recall Score = 0.972809667673716
===================================================================================================
```

Figure 9: Evaluation Metrics of the result



YOUR RECENT SUBMISSION

submission-2.csv
Submitted by LuckyAnnieMie · Submitted 2 hours ago

Score: 0.75803
Public score: 0.75377

Figure 10: Kaggle Score

Besides, Statistical knowledge is also import. In this project, we first used p-value of Chi-square test to judge the importance of each categorical feature, then used correlation heatmap to analysis Numerical features. Finally extracted 18 feature that we thought which are important to predict the result. From the figure, these features do take a part of importance.
To conclude, choices of models, both statistics or professional knowledge to choose features that influence the final cost and accuracy of the result.

5 Conclusion and Future Work

We first do the EDA of the datasets. EDA contains data type exploration, missing value, distribution of the target. Then we extracted 18 features from the main table, by using p-value of Chi-square test to judge the importance of each categorical feature and correlation heatmap for numerical feature. We also added 8 features which depend on financial knowledge.
After preprocessing, we used LightGBM Model and 5-fold validation method to train the data. We also used Bayes-optimization to tune the model and find the best parameter setting.
About the future work, one side, we need to learn more about finance domain and add more useful features from other tables. Other side, we can also try to add some deep learning models and even ensemble these models to improve the score .

6 Contribution

The following table shows our group member's basic information and contributions.

Group Members	Student ID	Work
Shihan BU	20787953	Coding, LaTeX Formatting, Report Writing
Yilin LI	20829426	Coding Support, Report Writing
Yuxin YANG	20787850	Coding Support, Report Writing

Table 3: Contribution of group members

All of our team members have fully participated in the above sections of coding implementation and report writing.

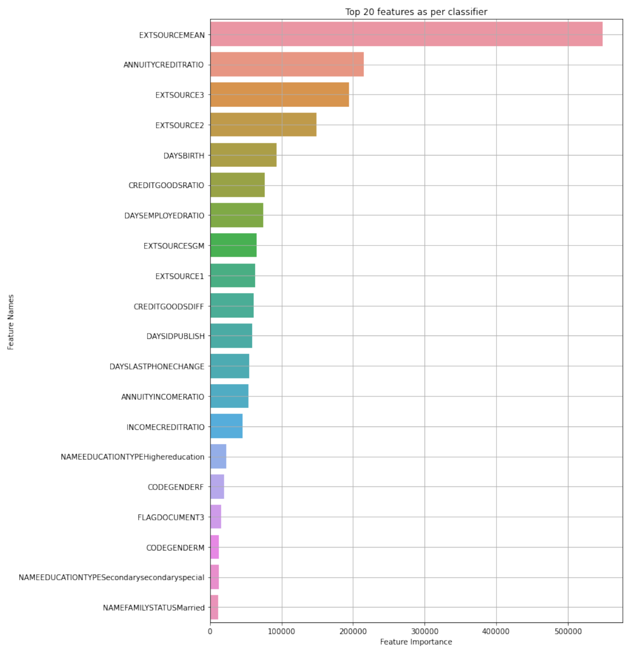


Figure 11: Top20 Features

References

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, 2017.