Kaggle Contest: Home Credit Default Risk

He Weiwei wheal@connect.ust.hk
Li Xintong xlifw@connect.ust.hk
Ma Jingkun jmabg@connect.ust.hk
Xu Tongcan txuav@connect.ust.hk

Abstract

Home credit default risk is one of the major risks of the credit business of a bank, which contributes a large portion of the profit, so good credit control is of vital importance. In this project, we went through the variable selection, model selection and model implementation procedures. Among the four models we evaluate(Logistics Regression, KNN, Random Forest and Gradient Boosting), Gradient Boosting achieved the best performance in cross-validation, its area under curve score is 0.71 and accuracy is 0.91. However, the Kaggle public score of Gradient Boosting drops to 0.50. Random Forest got the highest Kaggle public score among the four models, which is 0.54, while its area under curve score is 0.69 and accuracy is 0.70.

1. Variable Selection

1.1 Data in application_train

For labels in the application_train data, we choose some meaningful data. From column C to column L, we chose "CODE_GENDER" "FLAG_OWN_CAR" and "FLAG_OWN_REALTY" because we thought these could measure a person's financial resources, while for other variables, we believe that they have little effect on the determination of the borrower's economic status or there is no apparent corresponding relationship. For example, for the data "CNT_CHILDREN," we believe that it does not have a good correspondence with personal economic conditions. At the same time, it overlaps with the variable we selected, so we choose to discard this factor. Furthermore, if the client owns a car or house, we take the 'Y' as 1. Otherwise, we take 'N' as 0. Similarly, we take '1' for males and '0' for females.

For variable “NAME_INCOME_TYPE”, we believe that the different types of income can reflect the stability of the borrower's income. Therefore, we try to sort the different income types by grouping numbers. We assign the numbers 0 to 7 to different income types. At the same time, the larger the number, the more stable the corresponding income. In this case, we choose ‘unemployed’ as ‘0’, ‘student’ as ‘1’, ‘maternity leave’ as ‘2’, ‘pensioner’ as ‘3’, ‘working’ as ‘4’, ‘state servant’ as ‘5’, ‘businessman’ as ‘6’ and ‘commercial associate’ as ‘7’ divided by income level.

For “NAME_EDUCATION_TYPE”, we believe that the level of education can also reflect the financial status and credit rating of the borrower to a certain extent. We adopt a similar method as before to sort them by numbers. Thus, we take ‘Lower secondary’ as ‘0’,’ Secondary / secondary special’ as ‘1’, ‘uncomplete higher’ as ‘2’, ‘Higher education’ as ‘3’ and ‘Academic degree’ as ‘4’.

The column “DAYS_EMPLOYED” can measure the length of work and judge the level of income. Compared with the time of birth, we believe that working hours are more representative, so we choose it. In addition, we noticed a problem. In the training set, there are a lot of "365243", which accounts for about one-sixth of the total data volume. We think this is a type of wrong data. However, on the one hand, the proportion of data cannot be ignored, so it cannot be deleted directly. On the other hand, we have observed that the same "365243" also appears in large numbers in the test set, and it is not necessary to convert it to other values. Therefore, we decided to keep this type of data directly and add it to the subsequent analysis.

From column W to column AA, some flags represent whether one can be contacted or not, so we calculated the average of these five columns of numbers and named "AVG_FLAG_PHONE". Email is easy to register and is not important compared to mobile phone numbers, so we do not choose it.

Then we take the column "CNT_FAM_MEMBERS" because the number of family members will also affect the standard of living and economic status.

For columns" REGION_RATING_CLIENT" and "REGION_RATING_CLIENT_W_CITY," we calculate the average number to measure the region's rating where the client lives with or without taking the city into account.

From column AI to column AN, we calculate the sum of these numbers named "SUM_LIVE_CITY_WORK_CITY." The sum indicates the matching degree of the client's information, which can reflect a person's credit level from the side.

From column AP to column AR, we also calculate the sum of these numbers named "SUM_EXT_SOURCE" to see the normalized score from external data sources.

There is normalized information about the building from column AS to column CM, but we have already chosen "FLAG_OWN_REALTY," so we ignore all their numbers.

Then from column CN to column CQ, the two groups of numbers are similar, so we choose one group such as "OBS_60_CNT_SOCIAL_CIRCLE" and "DEF_60_CNT_SOCIAL_CIRCLE" and calculate the sum named "SUM_OBS_DEF", which can show the social surroundings evaluation of the client.

The variable "DAYS_LAST_PHONE_CHANGE" can reflect the stability of the client's contact information. For example, if a person frequently changes his mobile phone number, he is more likely to have credit problems.

Finally, from column CS to column DL, we calculate the sum of these numbers named "SUM_FLAG_DOCUMENT" to judge whether the client provides documents. The higher the number, the more documents provided.

1.2 Data in buearu

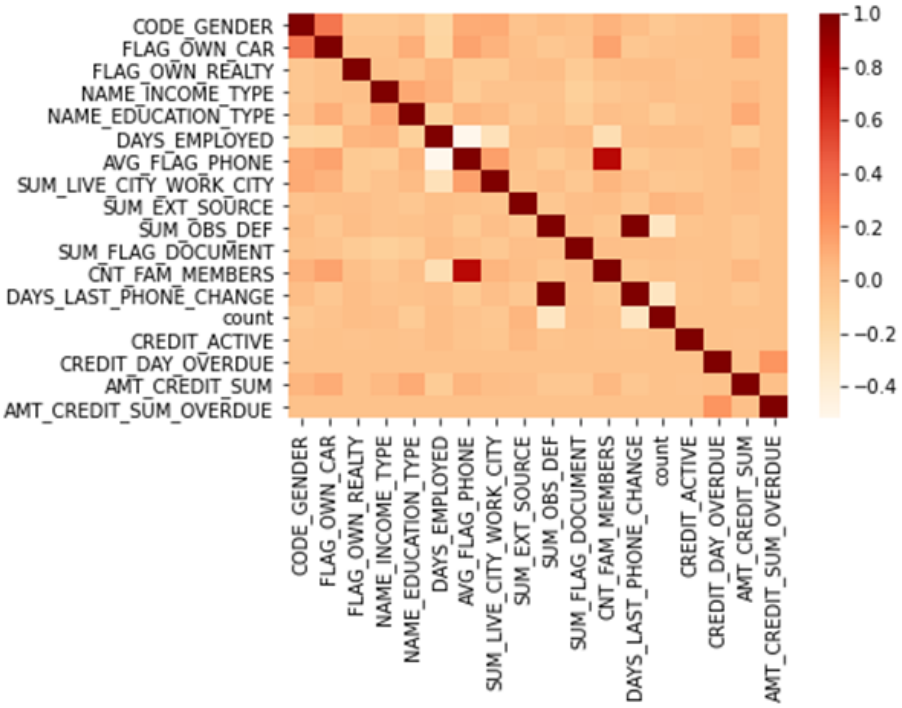
SK_ID_CURR	ID of loan in our sample - one loan in our sample can have 0,1,2 or more related previous credits in credit bureau
SK_BUREAU_ID	Recoded ID of previous Credit Bureau credit related to our loan (unique coding for each loan application)
CREDIT_ACTIVE	Status of the Credit Bureau (CB) reported credits
CREDIT_DAY_OVERDUE	Number of days past due on CB credit at the time of application for related loan in our sample
AMT_CREDIT_SUM	Current credit amount for the Credit Bureau credit
AMT_CREDIT_SUM_OVERDUE	Current amount overdue on Credit Bureau credit
DAYS_CREDIT_UPDATE	How many days before loan application did last information about the Credit Bureau credit come

The data from bureau.csv is the application data from previous loans that clients got from other institutions and that were reported to the Credit Bureau. We think the data from Credit Bureau, a government agency, should be quite reliable, so we decided to choose some variables from bureau.csv to train our model. Firstly, there are several SK_BUREAU_ID (Loan ID from credit bureau) with respect to one SK_ID_CURR(Loan ID from home credit). We chose the latest record by the column DAYS_CREDIT_UPDATE.

Then we screen the variables according to their definition and the number of NAs. To be specific, we chose the variables whose definition is not vague and NA values are relatively low as well as whose economic meanings may influence a lot to the default rate. Finally, we came up with CREDIT_ACTIVE, CREDIT_DAY_OVERDUE, AMT_CREDIT_SUM and DAYS_CREDIT_UPDATE 4 variables from bureau.csv to feed in the model. Considering the imbalance distribution of 0 and 1, we chose the 25% quantile to fill the NA values for last three variables and chose -1 to fill in NA values in CREDIT_ACTIVE because it’s the status.

Furthermore, we calculated the total number of records of every SK_ID_CURR from all the data files and also used this as an input to feed in the model. To sum up, we chose 5 more features outside the application.csv file.

1.3 Correlation Analysis



Before we put all the selected variables to train our model, we analyzed the correlation among the variables. According to the heatmap above, we found that two pairs of variables are highly correlated, which are: AVG_FLAG_PHONE and CNT_FAM_MEMBERS (with 0.78 correlation ratio) as well as DAYS_LAST_PHONE_CHANGE and SUM_OBS_DEF. So we dropped AVG_FLAG_PHONE and SUM_OBS_DEF and use the remaining 18 features to train our model.

2. Model Selection

2.1. Logistics Regression

We use Logistic Regression model and the Sklearn package to calculate. We choose penalty as ‘l2’ and test size as 0.3. The test accuracy is 0.92 but the ROC is only 0.58. In the application train test, the number of target ’0’ is larger than target ‘1’, roughly 14:1. Although the test accuracy is not low, the logistic regression model is not most suitable considering ROC.The score on Logostic: private score is 0.49923; public score is 0.49945.

```
Training accuracy: 0.9190587907365683
Test accuracy: 0.9197641269118619
```

```
ROC: 0.5794893266038316
```

2.2 k-nearest neighbors algorithm（kNN）

kNN is a nonparametric classification method, which is often used for classification and regression problems. Its input consists of the k closest training examples in the data set. When kNN deals with the classification problem, the output object is classified by a plurality of votes of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

The basic principle of kNN will not be repeated here; we mainly show the model processing process through specific code.

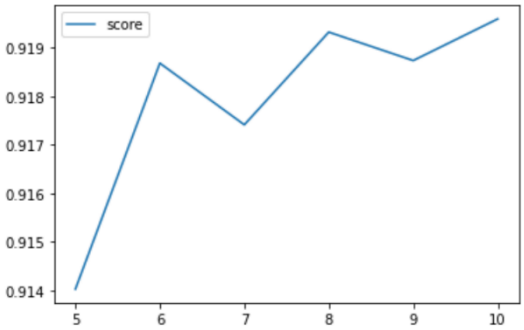
As an essential machine learning library in python, scikit-learn is efficient and straightforward. Here, we choose to use this library(sklearn).

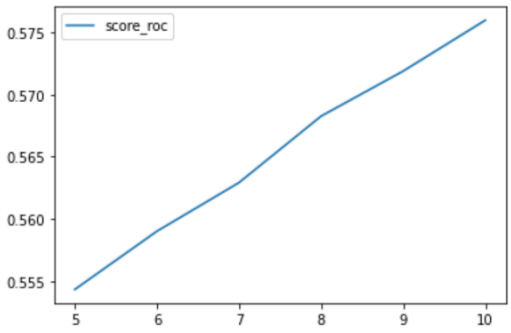
Since kNN uses distance for classification, if features have different units or very different scales, a feature may dominate the results. Therefore, we used data normalization and normalized data to the distribution with a mean of 0 and a variance of 1.

Here, we also need to pay special attention to the fact that in normalizing the test data set, we need to take the mean and variance of the training set. Because in a natural environment, on the one hand, we may not be able to obtain the mean or variance of all test data; on the other hand, it may lead to overfitting, making the results look efficient.

In order to choose a better value of k and build a more valuable model, we set up a for loop and test the value of k from 5-10(default_k_value=5). We try to evaluate by accuracy and Area Under Curve (Evaluate a score by cross-validation). Here, we use the functions accuracy_score () and cross_val_score () respectively. In general, the closer the value is to 1, the more effective the model is to obtain the k value we need.

In the end, we get the following results.





It can be seen that as the value of k becomes larger, the score of the model increases. Therefore, in the subsequent simulation, we may choose a k-value equal to 10.

In addition, for the hyperparameters of kNN, we can use the grid search method to obtain a model with a higher score. As a prevalent parameter adjustment method, there is a function GridSearchCV in sklearn, used to calculate the hyperparameters you want to adjust and return the best set of hyperparameter combinations.

In this model, we can obtain such effects through the following code:

```
param_grid = [
 {
  'n_neighbors':[i for i in range(1,11)],
  'weights' :['uniform'],
 },
 {
   'n_neighbors':[i for i in range(1,11)],
    'weights' :['distance'],
    'p':[p for p in range(1,11)]
 }

]
kNN_clf = KNeighborsClassifier()
grid_search = GridSearchCV(kNN_clf,param_grid=param_grid)
grid_search.fit(x_train,y_train)
grid_search.best_score_
grid_search.best_params_
```

Although we have made some adjustments to the kNN model, we can see that the roc_auc score of the model is still not up to expectations.

At the same time, when the amount of data is tremendous, the calculation amount of kNN is too large, the algorithm is slower than other models, and the overall efficiency is not high.

The score on Kagge: the private score is 0.50370; the public score is 0.50457.

2.3. Random Forest

Tree models are widely known to be good at classification. Compared to the decision tree, random forest has some advantages like bootstrapping and multiple trees to vote, and it's not so complicated compared to boosting model, so we decided to use random forest to do the default classification.After parameter tuning, we chose entropy to split the nodes and n_estimators=105, max_depth=11, min_samples_split=35.

```
RandomForestClassifier(n_estimators=105, max_depth=11,
                       criterion="entropy",class_weight="balanced",
                       min_samples_split=35)
```

We used cross validation to evaluate our model performance. The average fitting accuracy is about 0.70, and the average roc scores of this model are 0.69. The score on Kagge: private score is 0.538; public score is 0.542.

2.4. Gradient Boosting

Gradient Boosting is a kind of tree model which produces a prediction model in the form of an ensemble of weak predictors, for example, a single decision tree. Gradient Boosting is like an enhancement version of Random Forest, so we think it can have the best performance. We use the sklearn package of GradientBoostingClassifier to adjust different parameters to get the best fitting result.

Finally, we choose the learning rate to be 0.1, min_sample_split to be 5000, min_samples_leaf to be 10 and the max_depth to be 3. The fitting accuracy is about 0.91938, and the average roc score of this model is 0.7106 by using cross-validation to calculate the roc.

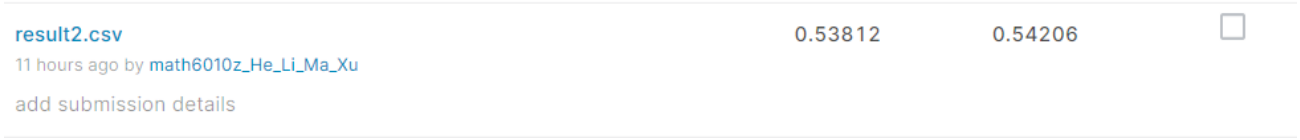
```
metrics.accuracy_score(y_test, y_pre)
0.9193847354557575
```

```
In [71]: score.mean()
Out[71]: 0.7105955472369826
```

However, the score on Kaggle is not very high, private score is 0.50039 and public score is 0.50050.

3. Final Score of Kaggle

At last, we choose the result of Random Forest to be our last submission.



4. Contribution of each member

Name	Code	Report
He Weiwei	The Logistic Regression and part of data cleaning in appilication_train.	Variable selection in application_train with Xu Tongcan. Model selection using Logistic.
Ma Jingkun	Data clean part and Gradient Boosting part.	Variable selection and the result of Gradient Boosting.
Xu Tongcan	kNN model Count and summarize some data	Variable selection in application_train with He Weiwei. Model selection using kNN.
Li Xintong	Cleaned and aggregated data from bureau.csv; Random Forest model implementation; Correlation analysis and plot heatmap	Variable selection in bureau.csv; Correlation Analysis; Model selection: Random Forest part; Abstract