ICT Proj3 : Home Credit Risk

Lei TAN
Lefei ZHANG
Donglin CHEN
Bo LI

21/12/2020

Contents

1 Introduction

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders. It is very important for lenders to predict whether or not a client will repay a loan or have difficulty, this has been a critical business demand in loan practice.

This project aims to provide a model which helps to make repayment predictions. The data are provided by Home Credit, a non-banking financial institution dedicated to provide lines of credit (loans) to the unbanked population. Using various statistical and machine learning methods, our model finally reached an accuracy of 79.135%.

2 Exploratory Data Analysis

2.1 Data Description

The data are provided by Home Credit, a service dedicated to provided lines of credit to the unbanked population.

There are 7 different sources of data :

- **application_train/application_test** : the main training and testing data with information about each loan application at Home Credit.

- **bureau** : data concerning client's previous credits from other financial institutions. Each previous credit has its own row in bureau, but one loan in the application data can have multiple previous credits.

- **bureau_balance** : monthly data about the previous credits in bureau. Each row is one month of a previous credit, and a single previous credit can have multiple rows, one for each month of the credit length.

- **previous_application** : previous applications for loans at Home Credit of clients who have loans in the application data. Each current loan in the application data can have multiple previous loans.

- **POS_CASH_BALANCE** : monthly data about previous credit cards clients have had with Home Credit. Each row is one month of a credit card balance, and a single credit card can have many rows.

- **credit_card_balance** : monthly data about the previous credits in bureau. Each row is one month of a previous credit, and a single previous credit can have multiple rows, one for each month of the credit length.

- **installments_payment** : payment history for previous loans at Home Credit. There is one row for every made payment and one row for every missed payment.

Fig.1 shows how all data files are related :

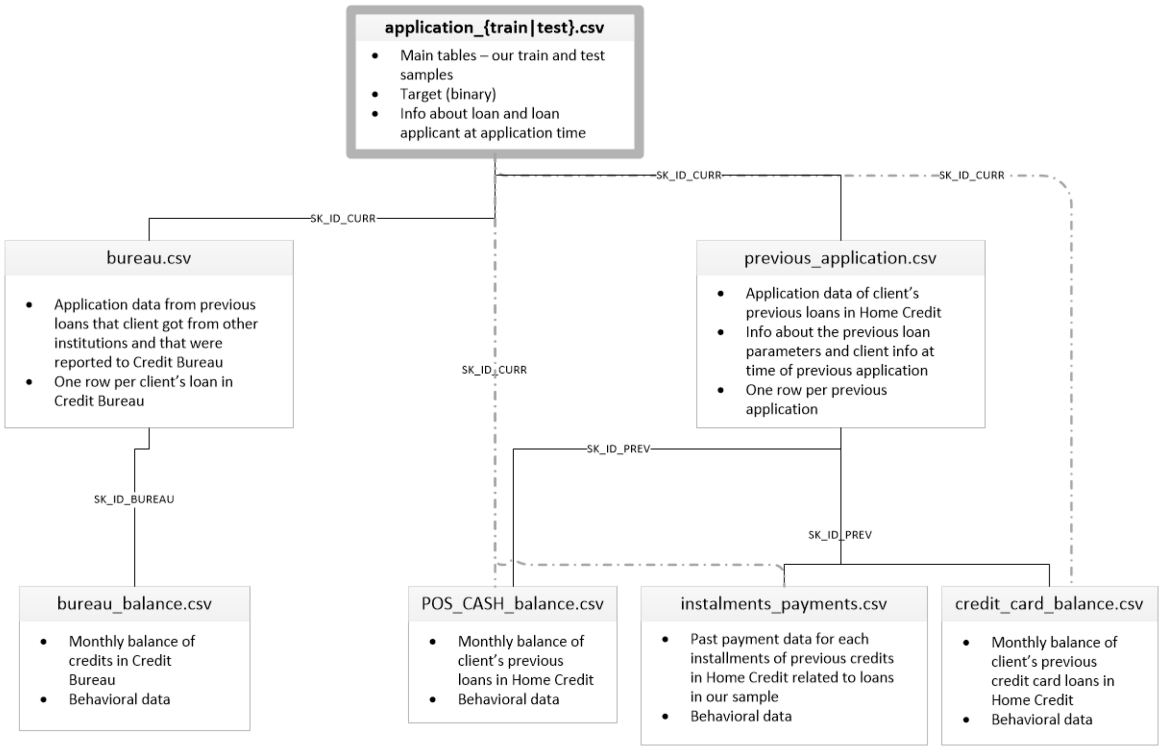
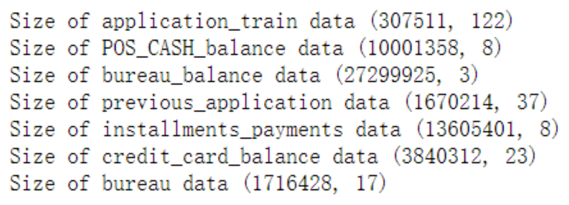

application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
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_BUREAU
SK_ID_PREV
SK_ID_PREV
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
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: Relations of data files

We read in all data files and look at their size.

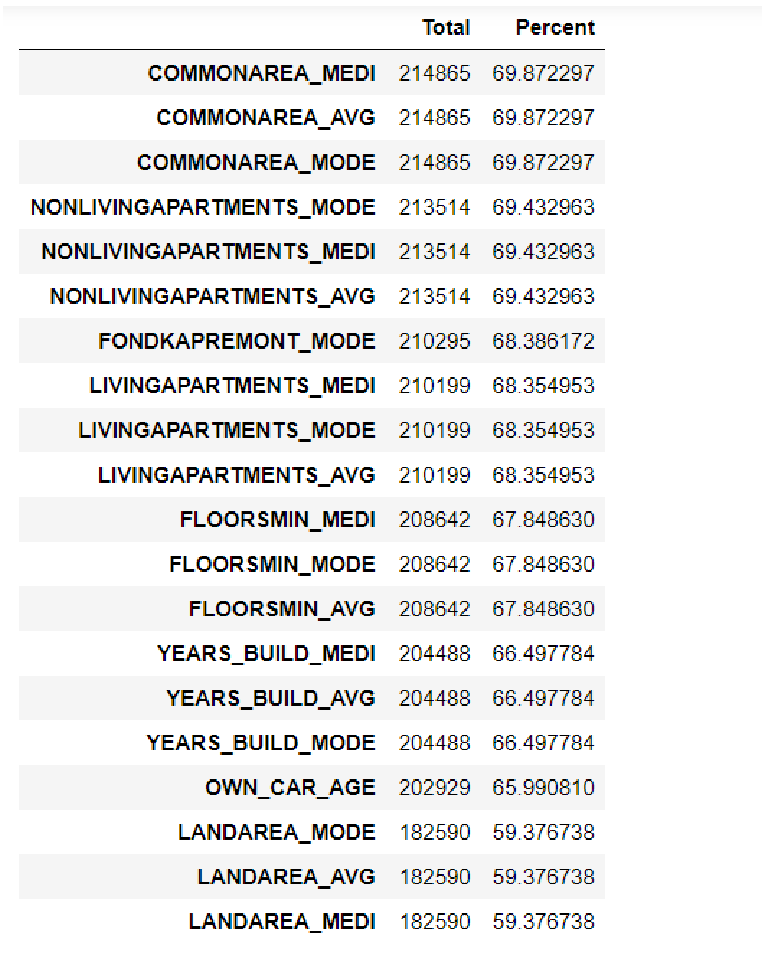


```
Size of application_train data (307511, 122)
Size of POS_CASH_balance data (10001358, 8)
Size of bureau_balance data (27299925, 3)
Size of previous_application data (1670214, 37)
Size of installments_payments data (13605401, 8)
Size of credit_card_balance data (3840312, 23)
Size of bureau data (1716428, 17)
```

Figure 2: Size of data files

2.2 Check for missing data

First, we take a look at the missing values. Fig.3 is an example of the number and percentage of missing values in **application_train**



	Total	Percent
COMMONAREA_MEDI	214865	69.872297
COMMONAREA_AVG	214865	69.872297
COMMONAREA_MODE	214865	69.872297
NONLIVINGAPARTMENTS_MODE	213514	69.432963
NONLIVINGAPARTMENTS_MEDI	213514	69.432963
NONLIVINGAPARTMENTS_AVG	213514	69.432963
FONDKAPREMONT_MODE	210295	68.386172
LIVINGAPARTMENTS_MEDI	210199	68.354953
LIVINGAPARTMENTS_MODE	210199	68.354953
LIVINGAPARTMENTS_AVG	210199	68.354953
FLOORSMIN_MEDI	208642	67.848630
FLOORSMIN_MODE	208642	67.848630
FLOORSMIN_AVG	208642	67.848630
YEARS_BUILD_MEDI	204488	66.497784
YEARS_BUILD_AVG	204488	66.497784
YEARS_BUILD_MODE	204488	66.497784
OWN_CAR_AGE	202929	65.990810
LANDAREA_MODE	182590	59.376738
LANDAREA_AVG	182590	59.376738
LANDAREA_MEDI	182590	59.376738

Figure 3: Missing features of "application_train"

When it comes to build our machine learning models, missing values need to be treated before processing. There are 3 options to treat the missing values:

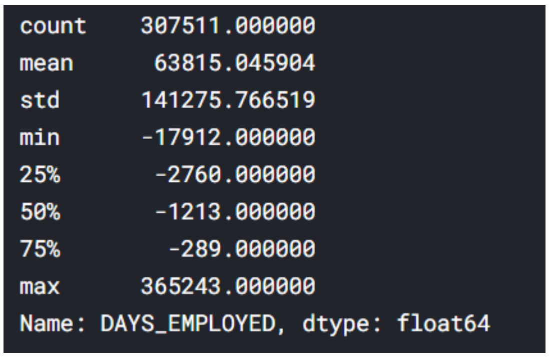
- Fill in these missing values (known as imputation).
- Use models such as XGBoost that can handle missing values with no need for imputation.
- Drop columns with a high percentage of missing values, although it is impossible to know ahead of time if these columns will be helpful to our model.

In our model, we choose to drop columns with a percentage of missing values higher than 70%.

2.3 Anomalies

We have a look at the anomalies within the data, these may be due to mistyped numbers, errors in measuring equipment, or they could be valid but extreme measurements. We support anomalies quantitatively by looking at the statistics of a column.

We found anomalies in column **DAYS_EMPLOYED**, which indicates how longs an employee has been employed :



```
count    307511.000000
mean      63815.045904
std      141275.766519
min      -17912.000000
25%       -2760.000000
50%       -1213.000000
75%        -289.000000
max      365243.000000
Name: DAYS_EMPLOYED, dtype: float64
```

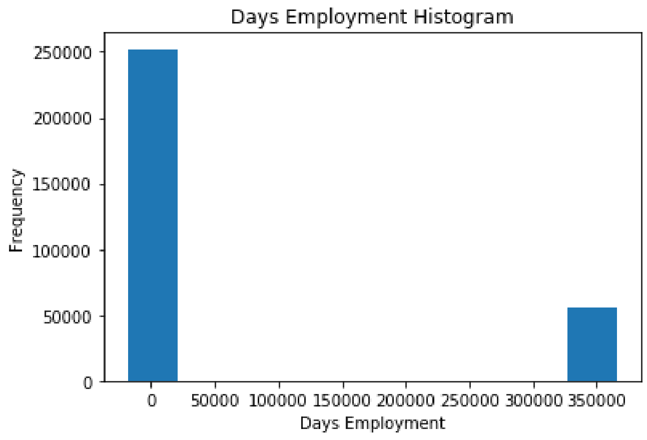


Figure 4: Types of loan

The maximum value is about 1000 years, which is absolutely impossible. The histogram of this label is shown in Fig.4.

We also subset the anomalous data to see if they tend to have higher or lower rates of default than the rest of the clients. It turns out that the non-anomalies default on 8.66% of loans, while the anomalies default on 5.40% of loans. The anomalies have a lower rate of default.

In this case, since all the anomalies have the exact same value, we want to fill them in with the same value in case all of these loans share something in common. Thus, we fill in the anomalous values with *np.nan* and then create a new Boolean column indicating whether or not the value was anomalous.

2.4 Data Distribution

We first take a look at the distribution of loan repaid or not, where 0 means repaid and 1 means not.

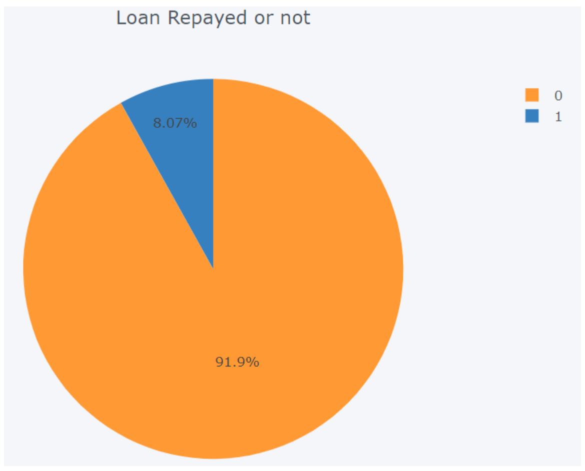


Figure 5: loan repaid or not

As shown in Fig.4, the data is highly imbalanced, most of the loans are repaid. Once we get into more sophisticated machine learning models, we can weight the classes by their representation in the data to reflect this imbalance.

Next we plot the distribution of types of loan.

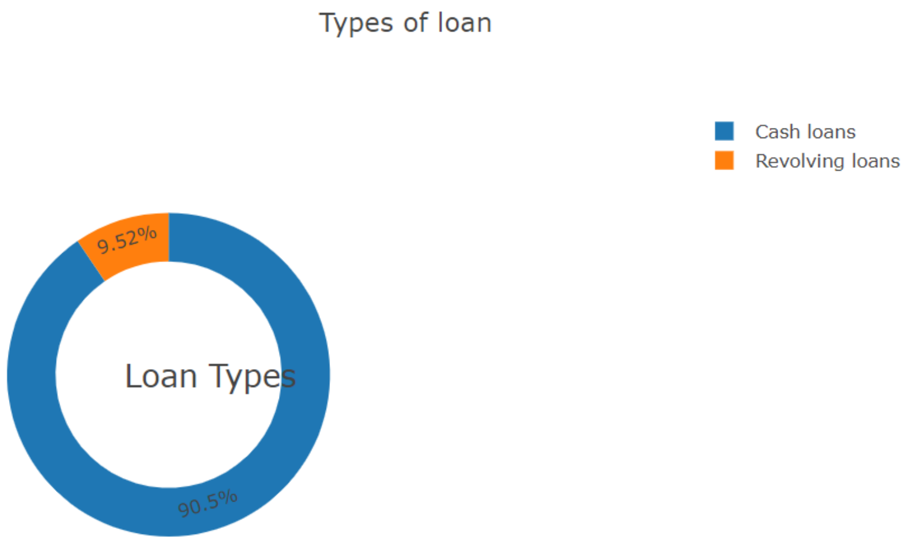


Figure 6: Types of loan

As shown in Fig.5, most of the loans are Cash loans which were taken by applicants 90.5 % loans. Revolving loan is an arrangement which allows for the loan amount to be withdrawn, repaid, and redrawn again in any manner and any number of times, until the arrangement expires. Credit card loans and overdrafts are revolving loans.

Fig.6 shows the income sources of applicants, 51.6 % Applicants mentioned that they are working. 23.3 % are Commercial Associate and 18 % are Pensioner etc.

Fig.7 shows the education of applicants. 71 % applicants have secondary and 24.3 % having higher education.

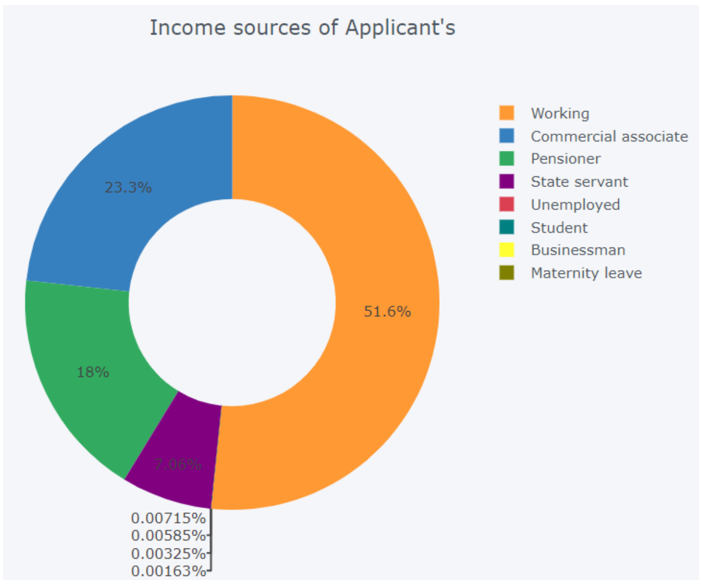


Figure 7: Income source

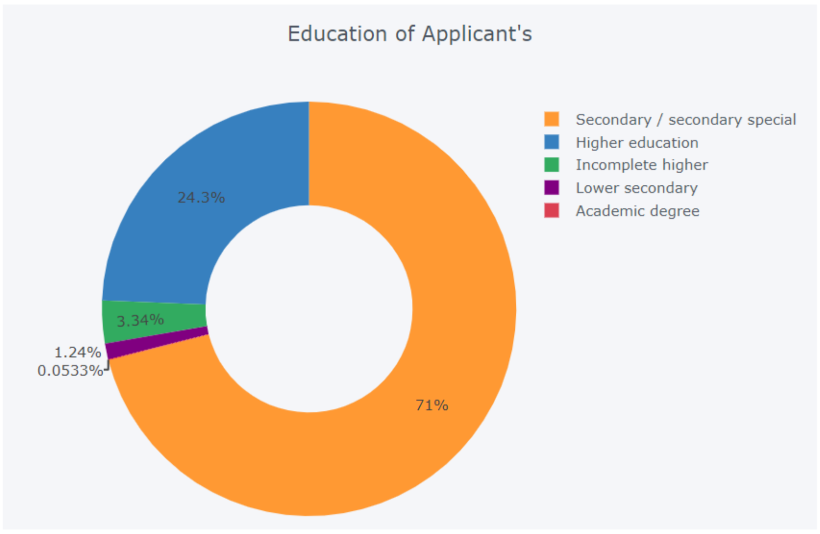


Figure 8: AMT credit

We are also interested in the distribution of labels separated by target. This can give us more direct information of the relation between a label and its target.

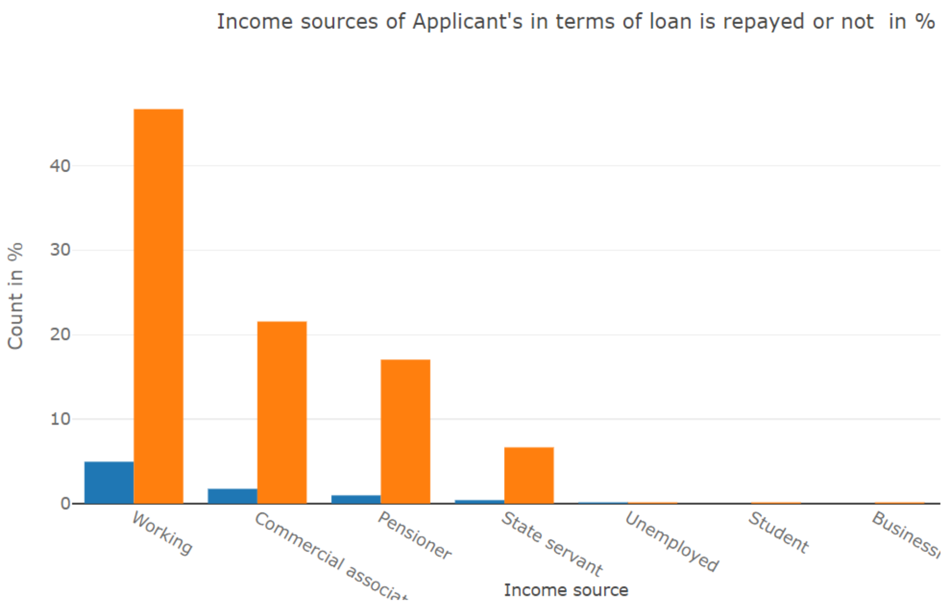


Figure 9: Income source

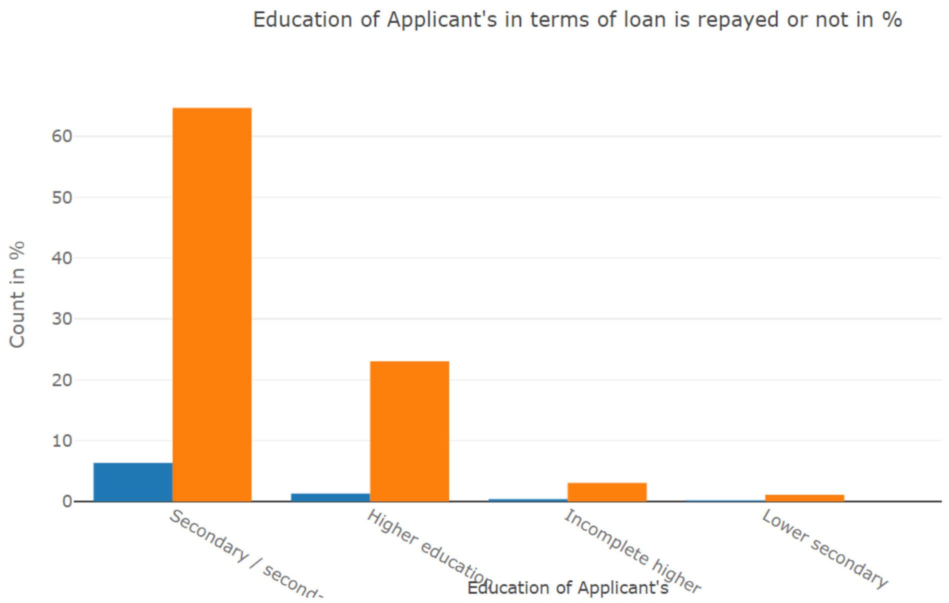


Figure 10: Income source

Fig.8 shows income sources of applicants in terms of loan is repaid or not. We can see that repaid cases and not repaid cases have nearly same distribution.

Fig.9 shows that people with higher education are more likely to repay loan (4% not repaid) while 9% people with secondary education didn't repay loan. Therefore, the information of education may be useful in further processing.

2.5 Research in previous applications

Previous applications are very helpful in loan prediction. We also separate repaid cases and not repaid cases.

Here are some labels that contribute in feature engineering.

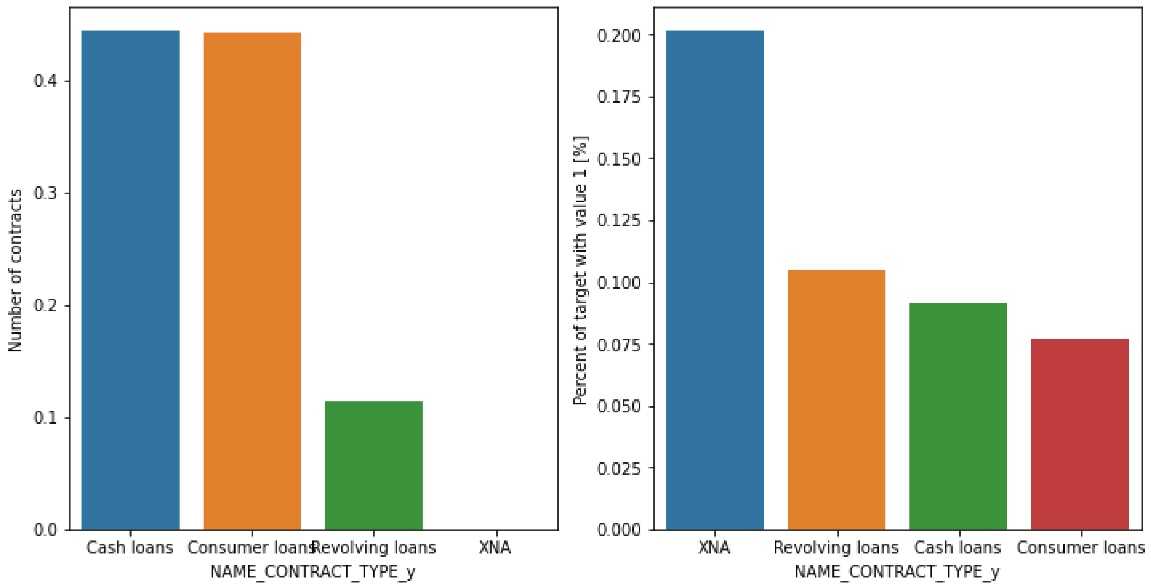


Figure 11: Income source

The percent of defaults loans for clients with previous applications is different for the type of previous applications contracts, decreasing from 10% for Revolving loans, then 9.5% for Cash loans and 8% for Consumer loans.

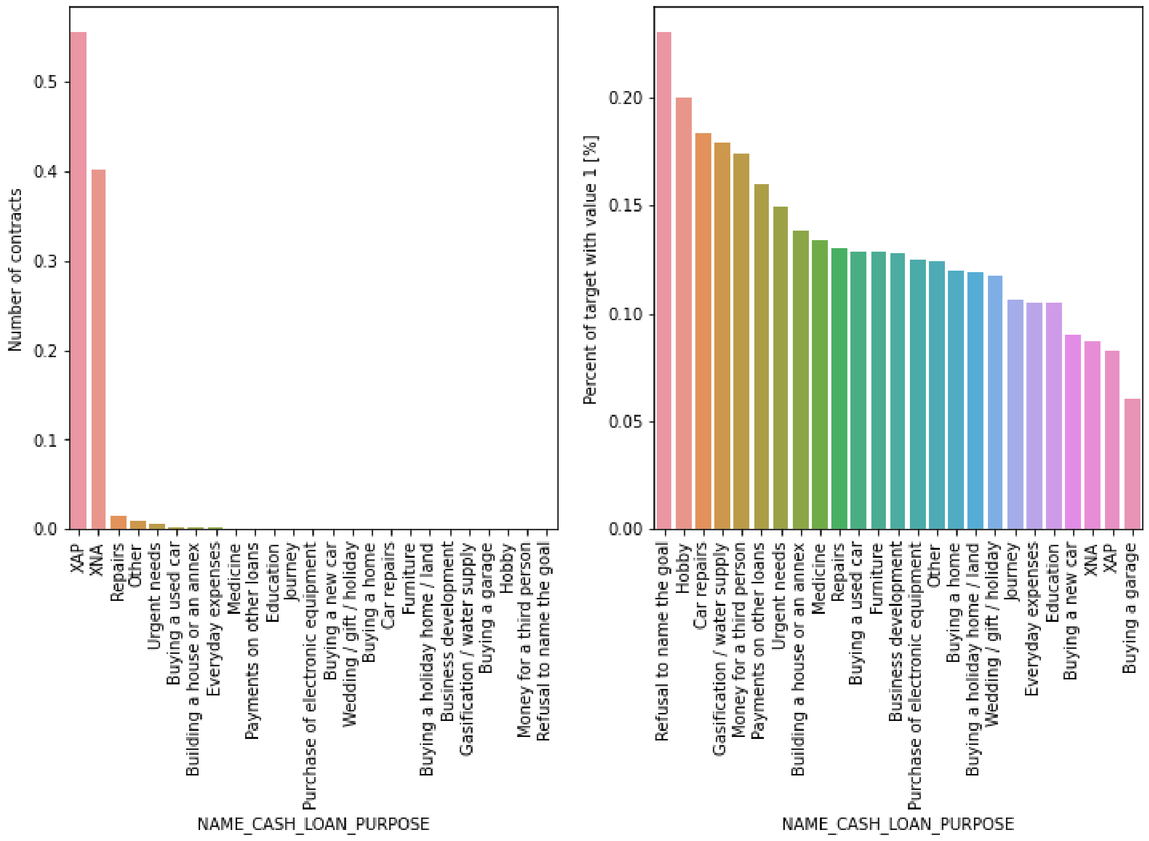


Figure 12: Income source

Clients with history of previous applications have largest percents of defaults when in their history are previous applications for cash loans for Refusal to name the goal (23%) , Hobby (20%), Car repairs ($\tilde{1}$8%).

In terms of percent of defaults for current applications in the sample (Fig.12), clients with history of previous applications have largest percents of defaults when in their history contract statuses are Refused (12%), followed by Canceled (9%), Unused offer ($\tilde{8}$%) and Approved (lowest percent of defaults in current applications, with less than 8%).

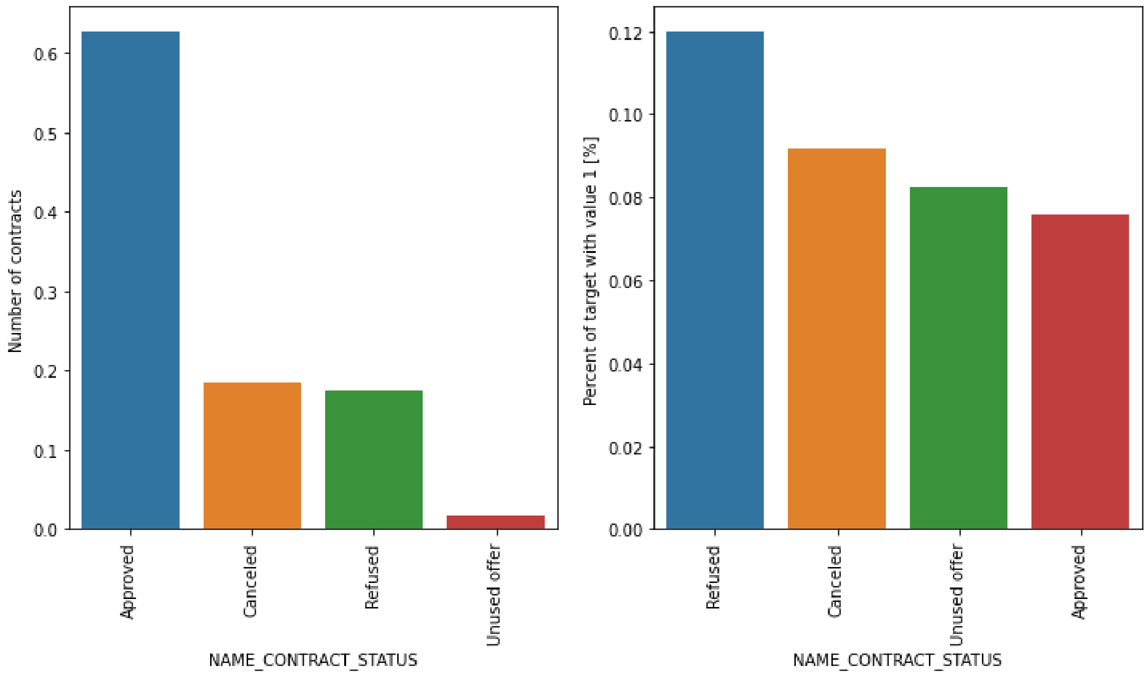
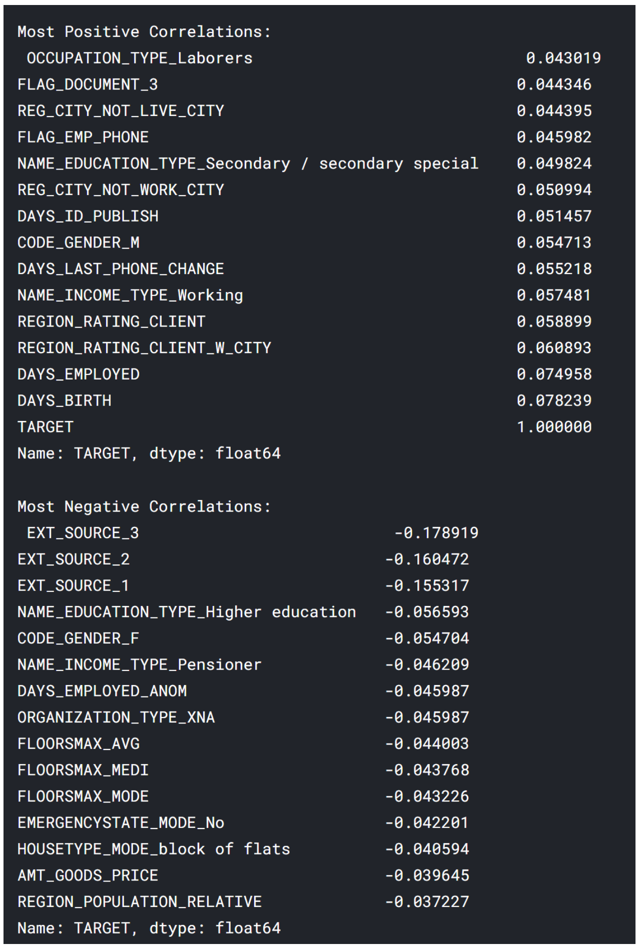


Figure 13: Income source

2.6 Correlation

One way to try and understand the data is looking for correlations between the features and the target. We calculate the Pearson correlation coefficient between the features and the target, which can give us an idea of possible relationships within the data.



```
Most Positive Correlations:
 OCCUPATION_TYPE_Laborers                              0.043019
FLAG_DOCUMENT_3                                       0.044346
REG_CITY_NOT_LIVE_CITY                                0.044395
FLAG_EMP_PHONE                                        0.045982
NAME_EDUCATION_TYPE_Secondary / secondary special     0.049824
REG_CITY_NOT_WORK_CITY                                0.050994
DAYS_ID_PUBLISH                                       0.051457
CODE_GENDER_M                                         0.054713
DAYS_LAST_PHONE_CHANGE                                0.055218
NAME_INCOME_TYPE_Working                              0.057481
REGION_RATING_CLIENT                                  0.058899
REGION_RATING_CLIENT_W_CITY                           0.060893
DAYS_EMPLOYED                                         0.074958
DAYS_BIRTH                                            0.078239
TARGET                                                1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
 EXT_SOURCE_3                           -0.178919
EXT_SOURCE_2                           -0.160472
EXT_SOURCE_1                           -0.155317
NAME_EDUCATION_TYPE_Higher education   -0.056593
CODE_GENDER_F                          -0.054704
NAME_INCOME_TYPE_Pensioner             -0.046209
DAYS_EMPLOYED_ANOM                     -0.045987
ORGANIZATION_TYPE_XNA                  -0.045987
FLOORSMAX_AVG                          -0.044003
FLOORSMAX_MEDI                         -0.043768
FLOORSMAX_MODE                         -0.043226
EMERGENCYSTATE_MODE_No                 -0.042201
HOUSETYPE_MODE_block of flats          -0.040594
AMT_GOODS_PRICE                        -0.039645
REGION_POPULATION_RELATIVE             -0.037227
Name: TARGET, dtype: float64
```

The DAYS_BIRTH is the most positive correlation, except for TARGET because the correlation of a variable with itself is always 1. DAYS_BIRTH is the age in days of the client at the time of the loan in negative days. The correlation is positive, but the value of this feature is actually negative, meaning that as the client gets older, they are less likely to default on their loan.

Furthermore, EXT_SOURCE_1,2,3 are the most negative, which indicates that EXT_SOURCE can also be very useful in prediction. Actually, we created polynomial features using the EXT_SOURCE variables.

2.7 Inspirations for new features

Existing labels in data files may not be the best features for prediction. By combining different labels, we can create some hand-craft features which may have a better performance.

Fig.14, 15, 16, 17 shows AMT credit, AMT total income, AMT goods price and number of family members.

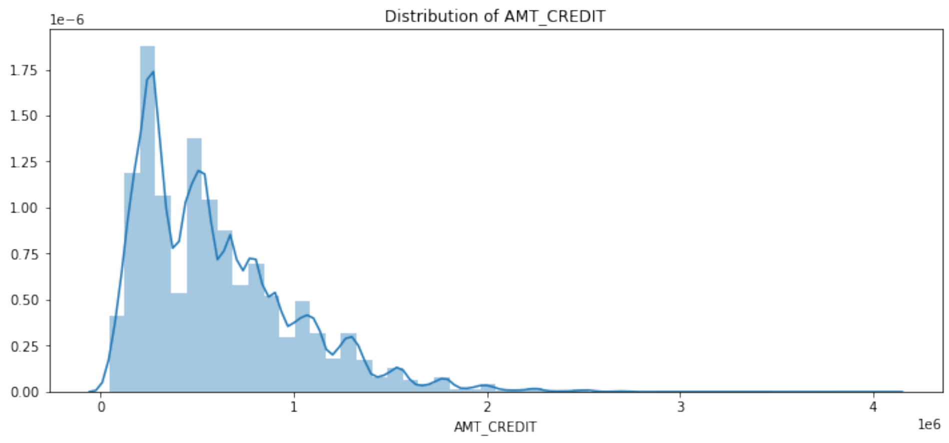


Figure 14: AMT credit

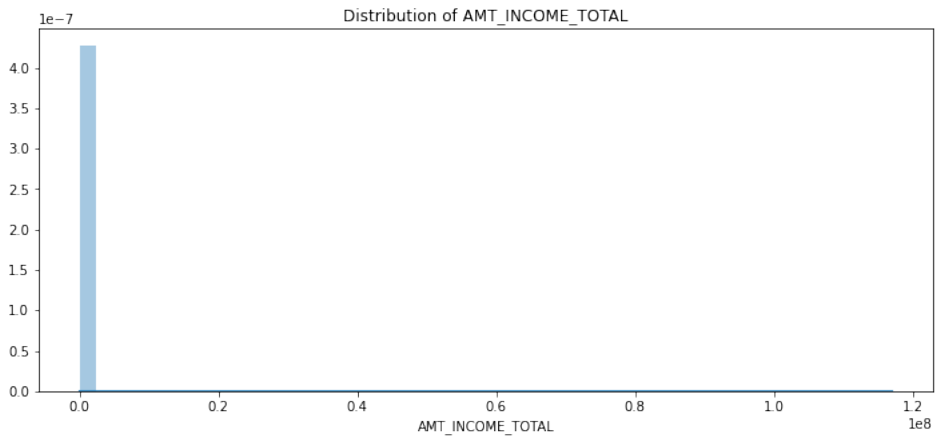


Figure 15: AMT total income

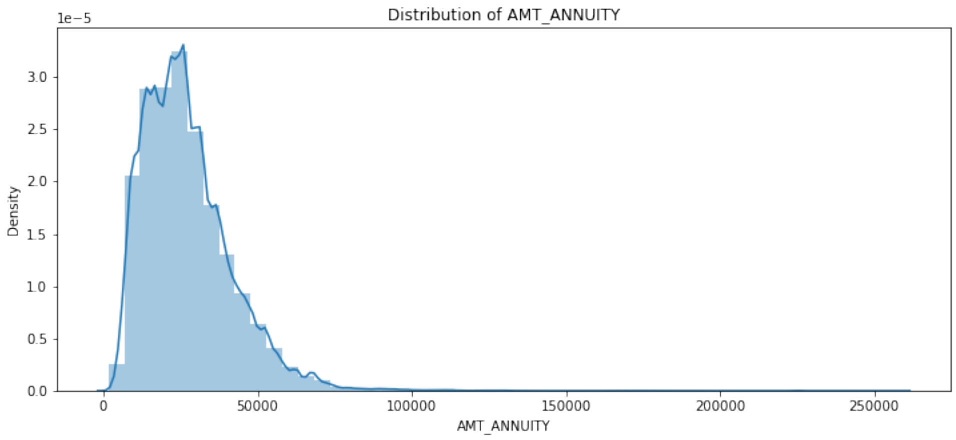


Figure 16: AMT annuity

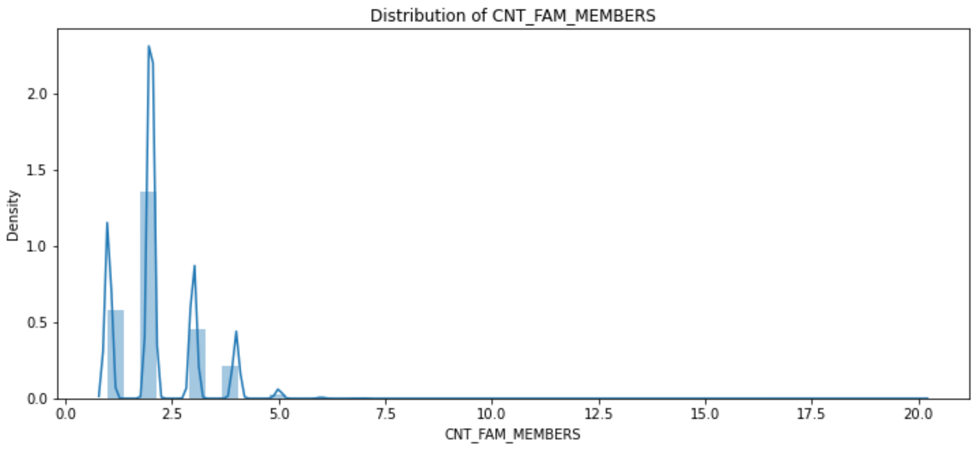


Figure 17: family members

Inspired by these distributions we propose some new features, for example:

- **Percentage of days employed** : Defined by $days\_employed/days\_birth$
- **Percentage of income credit** : Defined by $AMT\_total\_income/AMT\_credit$
- **Income per person** : Defined by $AMT\_total\_income/family\_members$
- **Percentage of annuity income** : Defined by $AMT\_Annuity/AMT\_total\_income$
- **e.t.c.**

3 Solution

In general, three of us worked on feature engineering separately, then we used similar classification model (LightGBM) with different hyper-parameters, and finally we did stacking on our three different models. Fig.18 shows the global architecture of our model.

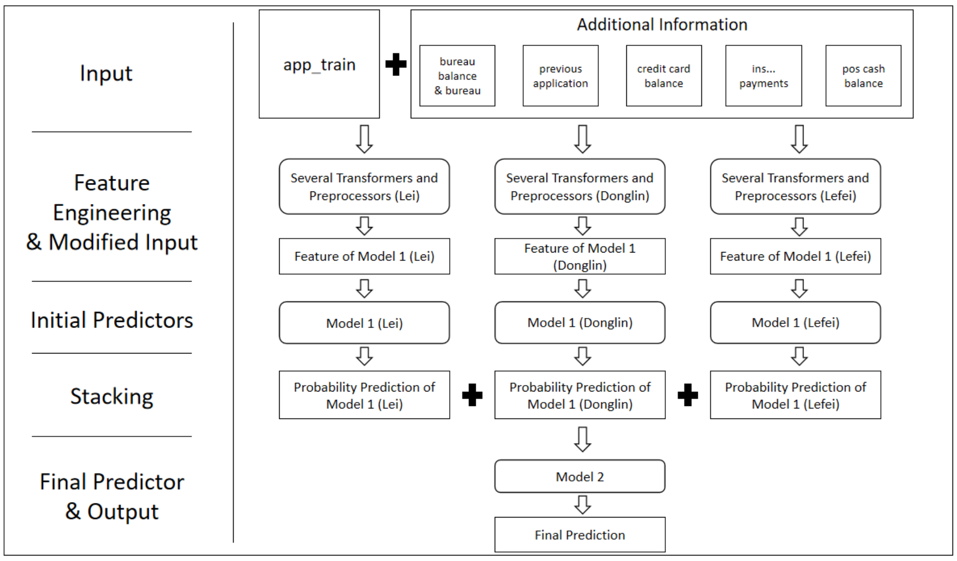

Input
Feature Engineering & Modified Input
Initial Predictors
Stacking
Final Predictor & Output
app_train
Additional Information
bureau balance & bureau
previous application
credit card balance
ins... payments
pos cash balance
Several Transformers and Preprocessors (Lei)
Several Transformers and Preprocessors (Donglin)
Several Transformers and Preprocessors (Lefei)
Feature of Model 1 (Lei)
Feature of Model 1 (Donglin)
Feature of Model 1 (Lefei)
Model 1 (Lei)
Model 1 (Donglin)
Model 1 (Lefei)
Probability Prediction of Model 1 (Lei)
Probability Prediction of Model 1 (Donglin)
Probability Prediction of Model 1 (Lefei)
Model 2
Final Prediction

Figure 18: The architecture of our algorithm

3.1 Feature Engineering

Firstly, for the application data “application_train.csv”, we used the results of EDA to perform some manual operations on the features, generating some new features that may be meaningful.

Anomalies: In EDA, we found that some features like “DAYS_EMPLOYED”, have anomalies. We replaced the anomalies with NAN and created a new feature to indicate whether an entry has anomaly or not.

Hand crafted features: Despite the lack of domain knowledge, we could still create some features attempting to capture what might be important for predicting whether a client will default on a loan. So we calculated some percentages to make new features. For example, we calculated days employed percentage, income credit percentage, income per person in a family, annuity income ratio, and so on.

Polynomial features: We also created polynomial features using the EXT_ SOURCE variables.

For the remaining data ("bureau_balance.csv", "bureau.csv", "credit_card_balan-- ce.csv", "installments_payments.csv", "pos_cash_balance.csv" and "previous_application.csv"), since we only have limited domain knowledge, we tried to make features as many as possible using groupby and aggregation on each table. After that, we join them together on the application data.

Aggregating Numeric Variables: To account for the numeric information in the data frames, we compute statistics for all the numeric columns. To do so, we groupby the client id, then aggregate the grouped data frame. We used statistics like 'mean', 'max', 'min', 'sum', etc.

Categorical Variables: We realized categorical variables encoding with the help of one-hot encoding, group by client ID and then aggregate by 'sum' and 'mean'.

Feature selection: Reducing the number of features can help the model learn during training and also generalize better to the test data. We got alittle bit too many features in the previous steps, so we did feature selection using LightGBM to delete the features with zero importance.

3.2 Classification Model

We used a k-folder LightGBM to do the classification after feature engineering. Since training a LightGBM and finding a good set of parameters takes much time, we directly used the parameters proposed by [1], found by Bayesian optimization.

Results: The results on test data of each model is presented in Table 1.

	Private Score	Public Score
Lei	0.78844	0.78845
Donglin	0.78960	0.79006
Lefei	0.78927	0.78982

Table 1: Results of each model on Kaggle

3.3 Stacking

With the predictions on the training set of the three models each, we did stacking using Logistic regression. The hyper-parameter of Logistic regression was obtained by cross validation.

Results: The results on test data of stacking is presented in Table 2. We can see that the accuracy has improved compared to each individual model.

	Private Score	Public Score
Stacking	0.79135	0.79135

Table 2: Results of stacking on Kaggle

4 API and Latency

In this section, we explain how we designed an HTTP POST API to expose our solution of Home Credit Default Risk as an online service. We have also done some basic performance tests on our HTTP API.

Above all, we have originally planned to launch our service on a Docker container and to upload the final image onto Dockerhub. However, as we could only use our PCs as server, our hard-disk space resources were very limited and we had no choice but to abandon the idea of creating a Docker image as the final demonstration. After a discussion with our mentor, we finally decided to start a local service directly on our PCs. In order that the others can use and share our result, we listed the environment requirement in our public repository on Github.

4.1 HTTP API Design

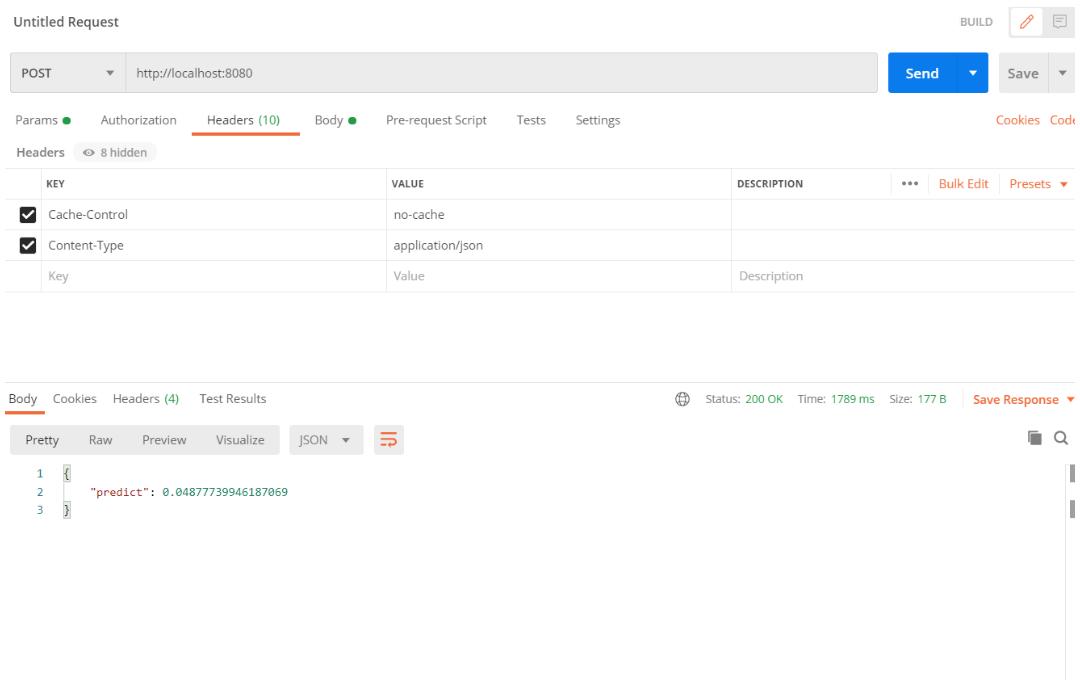
As it is discussed in the last section, our predictor follows the idea of stacking and contains mainly two parts: three initial predictors (as Model 1) designed by each of the group members and a final predictor (as Model 2) used to make a better prediction based on initial predictors' outputs. In order to avoid the high-cost training process of transformers and classifiers when launching the server, every group member saved his own pre-trained transformers and initial predictor in two files: preprocess{name}.joblib and clf{name}.joblib.

For the server side, we chose to use Flask, which is a micro web framework written in Python. We were asked to design an HTTP POST API, so the main part on the server side is a function named call(). It will be executed once a client raises an HTTP POST Request through the default path. While launching the server, we will directly load all csv files, which contain additional client information. At the same time, all pre-trained transformers and predictors will be imported, as well as other necessary functions in feature engineering. Different feature engineering processes on additional client information will also be done directly in this phase. Once the server receives a JSON request as input, the JSON request will be transformed into three pandas data-frames by joining additional client information and applying different transformers. Then, three pre-trained models predicts the probability that the client is unable to repay loans and a pre-trained logistic regression estimator, acting as Model 2, makes the final prediction of by applying the idea of stacking.

4.2 Performance Test

We launched the server on port 8080 of local host and we used Postman and Apache-Jmeter to realize the performance test. The following Postman Response Figure shows that our HTTP POST API works. However, we found that the latency is quite large (1794.48 ms) compared to our expectation (about 500ms). In Latency Detail Figure, we found out that a part of our latency is due to DNS Lookup which is not related with our algorithm but the Internet condition during testing. "Transfer Start" (about 1270.51) is the true latency of our algorithm. This latency can be interpreted from several aspects. First, we launched the server on our personal PC, which means that our computing power is very limited. Secondly, we were told that stacking is a necessary part of our algorithm. But, in fact, an algorithm without stacking can also obtain a relatively high accuracy (shown in the last section) but with a much lower time complexity (about 300 ms). Thirdly, our algorithm can still be improved. We can either choose classifiers with better hyper-parameters (thus decrease the number of estimators) or drop more features that are not closely related (thus decrease the running time of each estimator).

Besides, we have also used the Apache JMeter to realize stress testing on our API. We set 50 parallel threads applying our API at the same time. The result is that all requests were passed but the average response time was about 50s, which means that our API is not capable to deal with parallel requests. The computing power is of course a critical reason. But the algorithm itself can also be designed to obtain a better performance on stress testing. This could be a future work of this project. A very detailed summary of Stress Testing Performance could be found in our repository on Github.

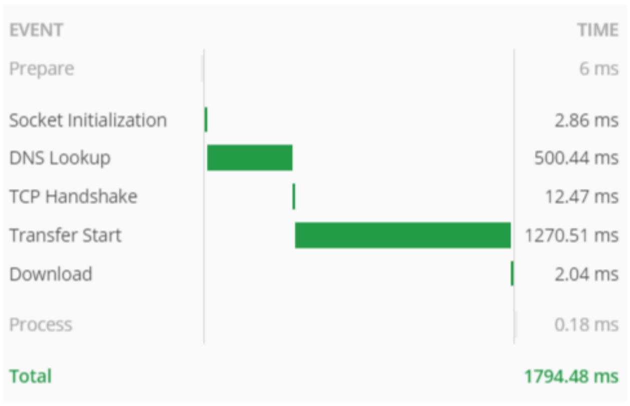


	KEY	VALUE	DESCRIPTION
✓	Cache-Control	no-cache	
✓	Content-Type	application/json	
	Key	Value	Description

Status: 200 OK Time: 1789 ms Size: 177 B

```
{
    "predict": 0.04877739946187069
}
```

Figure 19: Postman Response



EVENT	TIME
Prepare	6 ms
Socket Initialization	2.86 ms
DNS Lookup	500.44 ms
TCP Handshake	12.47 ms
Transfer Start	1270.51 ms
Download	2.04 ms
Process	0.18 ms
Total	1794.48 ms

Figure 20: Latency Detail

5 Conclusion

In this project, we have gone through a complete process of an industrial machine learning project: beginning with exploratory data analysis and data visualization, then feature engineering, followed by classification and ensemble learning, and finally exposing our machine learning solution for online prediction and conducting several performance tests. The accuracy of our model is 0.79135, ranking

2474/7140 on Kaggle.

References

[1] Aguiar. *LightGBM with Simple Features*. 2018.