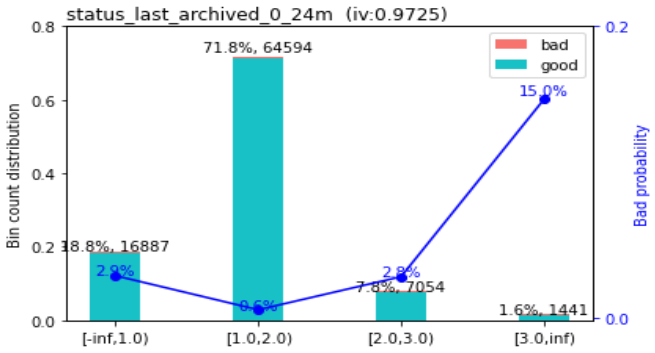
Exploratory Data Analysis:

Data is highly imbalanced (~100:1). Several parameters have significant number of missing values. Due to loan application software limitation, some parameters get lots of missing values. For example, new customers are likely to skip filling some forms. After doing exploratory analysis for missing variables, I kept all categorical parameters and replaced nan with zero. However, five numerical features with missing values are completely removed.

After looking at the defaults rates for merchant categories and groups, I found out that three categories got significantly different rates from their merchant groups. After creating separate groups for them, I recalculate defaults rates for merchant groups and re-encode merchant groups based on approximate default rates and treat the new merchant parameter as numerical parameter. The merchant group and category were dropped out.

I have seen the following example distribution for categorical parameters quite often. The first bins more likely represent new customers. However, the next bins show certain trends for the default rates



All the categorical parameters which needed one-hot encoding showed the above characteristics. So, I added additional binary parameter for the first bin for each feature with above characteristics and treated the categorical parameters as numerical parameters. Basically, we are creating piecewise parameters. The binary parameters are redundant for tree-based models because its not adding any value and more likely get the lowest feature importances. However, they are needed for logistic regression and can get significant weights. Also, I removed several parameters with no trend directly.

Feature Selection & Hyper Parameter Tuning

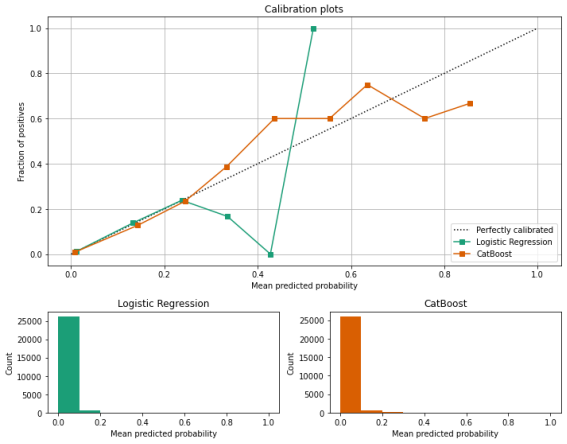
Using logistic regression, catboost classifier and combined with RFE rankings, I reduced features one-by-one iteratively. When I built scorecards before, I tested the approach using simulation. After selecting the certain parameters, I usually study interaction(non-linear) terms for logistic regression and build logistic regression model which is comparable to high-performance models. However, I did not do interaction term study this time because the logistic regression fits gave the comparable results with other models. Second, our goal is measuring the calibrated probability. After feature selection, I did hyper parameter tuning for logistic regression and catboost classifier with brier loss score and roc auc as scoring metric.

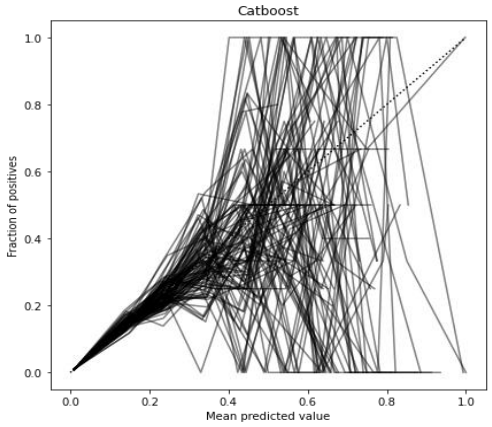
Probability Calibration

We are using CatBoost classifier and Logistic regression for the final probability calibration study. Both models have comparable roc auc. Catboost gets better brier score(lower) since its more important for predicting probability.

Classifier	Brier Loss	F1	Roc AUC (test)	Roc AUC (train)
Log Reg	0.013458	0.005063	0.854536	0.863022
Cat Boost	0.012833	0.035176	0.882436	0.905516

The first plot show calibration curves after calibration. Logistic regression model gave worse results for both uncalibrated and calibrated fits and had bias due to statistics in the high probability region. Since the final calibrated Catboost model gave better Brier score and unbiased results, I chose catboost for the final model. The second plot showed results for catboost model fits with 100 different random samples.





I stacked the first ten models of the above models to reduce statistical uncertainty and calculated probability for the template.

	uuid	pd
0	6f6e6c6a-2081-4e6b-8eb3-4fd89b54b2d7	0.004317
1	f6f6d9f3-ef2b-4329-a388-c6a687f27e70	0.002643
2	e9c39869-1bc5-4375-b627-a2df70b445ea	0.000032
3	6beb88a3-9641-4381-beb6-c9a208664dd0	0.009468
4	bb89b735-72fe-42a4-ba06-d63be0f4ca36	0.042487
5	e4eede99-76a3-4437-a540-3059a1eff67c	0.010298
6	a2af8d9e-9f81-4185-8fff-b2ec49d681a6	0.000181
7	ec910486-1e66-402a-80f2-08c6f04a9a1b	0.012166
8	08973cf0-646a-4fa7-9f1f-d03f76ffd59c	0.002231

Please check notebooks for the data analysis.

API :

I have an existing google cloud account and can setup free servers on google cloud easily. I am confident that I can easily pickup AWS. Nowadays, AWS console interface looks much more like GCP console.

I used fastapi and build the api on my own server. Fastapi is quite fast and easy to implement API without writing a lot of code. The validators for input information can be implemented easily.

First, you need to get token using the username and password in the following code. The tokens are valid in 30 minutes. To get token, send the following request:

POST https://batbold.mybox.mn/auth/token
Content-Type: application/json

{"username":"user1","password":"user123"}

After receiving the token, use it in the following code. All the necessary features need to be included and give error if not. Only the second parameter can get null value (can be missed). If you put numbers outside of the expected ranges, it will give error and explanation.

POST https://batbold.mybox.mn/req1
Content-Type: application/json
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAi


```
{  "uuid": "6f6e6c6a-2081-4e6b-8eb3-4fd89b54b2d7",
   "account_worst_status_0_3m": 1.0,
   "age": 20,
   "max_paid_inv_0_12m": 7225.0,
   "max_paid_inv_0_24m": 7225.0,
   "num_arch_dc_0_12m": 0,
   "num_arch_rem_0_12m": 0,
   "num_unpaid_bills": 1,
   "status_last_archived_0_24m": 1,
   "sum_capital_paid_account_0_12m": 8815,
   "sum_paid_inv_0_12m": 27157,
   "merchant_group": "Clothing & Shoes",
   "merchant_category": "Youthful Shoes & Clothing"
}
```


I created json file for testing api and attached to the email.

Codes : https://github.com/batboldsanghi/default_probability_prediction/

Data analysis and API codes can be found in notebook and fastapitest directories, respectively.