HOME CREDIT

DEFAULT RISK PREDICTION

MSBA 6420 Final Project Report

Group: Jiatong Li, Jingwen(Echo) Pei, Sirui Jiang, Xinbo Wang, Zhaoyan Zhi

Date: April 22nd 2022

Table of Contents

Project Introduction

Background

Founded in 1997, Home Credit B.V. is an international non-bank financial institution in the Czech Republic and headquartered in the Netherlands. The company primarily offers affordable point-of-sales (“POS”) loan, cash loan, and revolving loan products to underserved borrowers in nine countries. As of June 2019, they have served over 123 million borrowers, earning 2.1 million operating income. [1]

The company mainly focuses on borrowers in the blue-collar and junior white-collar segments who earn regular income from their employment or micro-businesses, but are less likely to access financing from banks and other traditional lenders. Customers typically start with POS financing in stores, then reliable customers can adopt broader consumer credit products, and ultimately receive fully-fledged branch-based consumer lending from Home Credit. [2]

Problem Statement

While many people struggle to get loans due to insufficient or non-existent credit histories, Home Credit strives to broaden financial inclusion for these unbanked population by providing a positive and safe borrowing experience. But, unfortunately, this population is often taken advantage of by untrustworthy lenders.

In order to correctly identify whether a customer with little or no credit history is trustworthy to start service with, Home Credit is currently using various statistical and machine learning methods to make predictions on customers’ repayment abilities. The idea is about predicting future payment behavior of clients from application, demographic and historical credit behavior data (such as telco and transactional information).

As a consulting team, our goal is to further explore the full potential of the data provided by Home Credit. To make more accurate predictions, we will be working on feature engineering to create more meaningful data, and also train different prediction models and methods to achieve the highest accuracy possible. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Data Source

All of the data was provided by Home Credit on Kaggle[3], including clients' application data, previous credit data from other financial institutions, as well as previous loan data and payment data at Home Credit.
There are 7 data sources:

- application_train/application_test: the main training and testing data with information about each loan application at Home Credit.
- bureau: data concerning the client's previous credits from other financial institutions.
- bureau_balance: monthly data about the previous credits in the bureau.
- previous_application: previous applications for loans at Home Credit of clients who have loans in the application data.
- POS_CASH_BALANCE: monthly data about previous point of sale or cash loans clients have had with Home Credit.
- credit_card_balance: monthly data about previous credit cards clients have had with Home Credit.
- installments_payment: payment history for previous loans at Home Credit.

Solution Overview

Technical Specification

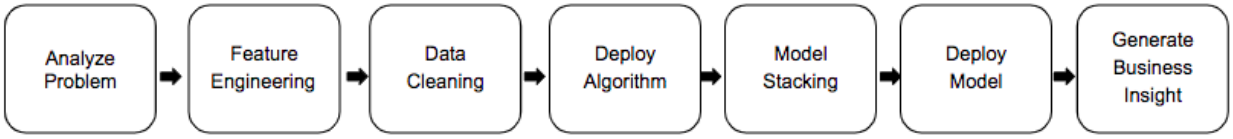
The code provided in this analysis is developed in Python 3.8 with Jupyter Notebook, with the following library specifications.

Library	**Version**
Pandas	1.3.5
Numpy	1.21.5
Matplotlib	3.5.1
Sklearn	1.0.2
Lightgbm	2.2.3
Xgboost	1.3.0
bayesian-optimization	1.2.0

Methodology

Six steps are taken to tackle our main task:

1) Analyze the problem: Define the business goal as prediction and exploration of features importance.
2) Feature engineering: Create new features based on business context and generate new features at application level utilizing statistical methods.
3) Data cleaning: Remove outliers and null value and transform the data into correct format to make sure the data is clean and suitable for modeling.
4) Deploy algorithm: Boosting and bagging models, which performs well in reducing bias and variance respectively, are used to make predictions. Specifically, models used include XGboost, LightGBM and Logistic Regression.
5) Model stacking: Conduct model staking by combining the prediction result of different well-performed models to improve prediction performance.
6) Deploy Model: Choose the final model which provides the best performance and make predictions.
7) Generate business insight: Generate feature importance from modeling results and compare the importance of different types of features. Make suggestions on how to better utilize non-credit information to predict fraud.



Detailed Implementation

Feature Engineering

Our implementation starts with feature engineering, a crucial step which largely might decide our model performance. The main features we used to train our model are from the application train dataset. Each row represents one loan record with corresponding client's personal information such as gender, property, number of children, etc.. We included all of the features from this main dataset. The following features listed are a part of representative features and transformated features we used in our model.

Application train

Gender: The applicant's gender including male, female
Ownership of Car: Whether the applicant owns a car or not
Number of children: How many children does the applicant has
The amount of income: how much does the applicant earns annually

Marital status: What is the marital status of the applicant including married, single, divorced, etc..
The ratio of credit to income: the ratio of debt and the total amount of annual income of the applicant
The ratio of monthly due to income: the ratio of monthly due amount to the total amount of annual income of the applicant
Terms of loans: The number of terms of the current loan
The ratio of employed days to age: the ratio of employed days to the applicant's age

A preliminary helped us select four supporting datasets out of the six related ones - the bureau, credit card balance, previous POS, and cash loans balance - that could probably make our prediction more reliable. Next, we transformed part of the raw features to receive more readable and informative output, such as one-hot encoding, simple addition, subtraction, average, etc.. Only a part of representative features are included in the following list.

Bureau

Total number of loans: the total number of loans the applicants have applied and updated in Credit Bureau
The percentage of active loans: the percentage of active loans among all loans the applicants have applied for
Types of loans: different types of loans including credit card, consumer credit, car loan, Mortgage etc.
Average number of loans per type: the average number of loans of every type of loan.
The number of loans within months: How many loans the applicant has applied for within 1 month, 3 month, 6 month, 1 year, 5 years.
Days since last loan: The number of days has passed since the applicants applied for the last loan
The percentage of loans prolonged: The percentage of loans that have been prolonged among all loans the applicant has applied for
The percentage of the amount of loans overdue: the percentage of the amount of loans overdue among the total remaining amount of loans

POS Cash

The number of POS loans: The total number of POS loans for each applicant
The number of POS loans per status: the total number of POS loans that are active, approved, canceled, completed, etc.
The percentage of POS loans per status: the percentage of loans of each status among all POS loans for the applicant
The number of installments: the total number of installments among all POS loans the applicant has applied for

The number of future installments: the total number of installments that the applicant has not paid for yet
The percentage of future installments: the percentage of installments that the applicant has not paid for yet among all installments
The max and avg days overdue: the maximum and average days past due among all POS loans that the applicant has applied for
The max and avg days overdue for low loan amount: the maximum and average days past due among all POS loans that have low amount that can be tolerated

Credit Card

The number of credit cards: The total number of credit cards for each applicant
The number of credit cards per status: the total number of credit cards that are active, completed, demand, signed
The percentage of credit cards per status: the percentage of credit cards of each status among all credit cards for the applicant
The balance of credit cards: the total amount of money the applicant owe the credit card companies for all credit cards
T**he amount of credit card limits**: the total amount of credit cards limits for each applicant
The drawing amount of credit cards: the total amount that the applicant has drew overall and from ATM
The number of drawings: the total number of drawings that the applicant has done overall and from ATM
The payment of credit cards: the total amount of payment the applicant has done
The percentage of usage: the percentage of limits that the applicant has used among the overall credit cards' limit
The percentage of drawing: the percentage of drawing among the applicant's total balance
the percentage of payment: the percentage of payment that the applicant has made among the total balance
The amount of interest: the total amount of interest that the applicant has to pay based on the difference between the amount of receivable and receivable principal
The percentage of receivable principal: the percentage of receivable principal among the total amount of receivable
The average amount of drawing: the average amount of drawing/ATM drawing each time

Model Tuning

LightGBM

To tune the models and select the best model, we use AUC score from 5-folder cross validation on the training data as metrics, and use Bayesian Optimization to search through the hyper-parameter space to maximize the score. Below is a sample code from our notebook to walk through the process of tuning a LightGBM model.

Step 1: Setup the objective function for the bayesian optimization. The optimizer will use this function to try out multiple combinations of hyper-parameters and try to maximize the output, which is defined as average AUC score from 5-folder cross validation.

```
from sklearn.model_selection import StratifiedKFold, cross_val_score
from lightgbm import LGBMClassifier

folds = StratifiedKFold(n_splits=5, shuffle=True)

# defining the goal function, which takes hyper-parameters to train and evaluate the model
def lgbm_bays_opt(learning_rate, num_leaves, n_estimators, min_split_gain,
                  bagging_freq, bagging_fraction, feature_fraction, evaluation=True):
    # set up LGBM model with the given parameters.
    clf = LGBMClassifier(learning_rate=learning_rate,
                         num_leaves=int(num_leaves),
                         n_estimators=int(n_estimators),
                         min_split_gain=min_split_gain,
                         bagging_freq=int(bagging_freq),
                         pos_bagging_fraction=bagging_fraction,
                         neg_bagging_fraction=bagging_fraction/10,
                         feature_fraction=feature_fraction,
                         n_jobs=-1,
                         is_unbalance=True,
                         metric='auc')
    if evaluation:
        # Train and calculate AUC of the given model with 5-folder cross-validation
        score = cross_val_score(clf, X=train_X, y=train_y, cv=folds, scoring='roc_auc')
        return score.mean()
    else:
        # if evaluation is set to False, will output the model directly
        return clf
```

Step 2: Setup the hyper-parameter searching space and the bayesian optimizer to perform the tuning. Here we run in total of 55 iterations to find the optimal combination of hyper-parameters.

```
# !pip install bayesian-optimization
from bayes_opt import BayesianOptimization

param_space = {'learning_rate': (0.001, 0.1),  # learning rate (low num of tree -> high learning rate, vice versa)
               'num_leaves': (100, 1000),      # max leaves each tree
               'n_estimators': (100, 1000),    # num of trees
               'min_split_gain': (0,0.1),      # min gain each split
               'bagging_freq': (1, 10),        # bagging reperformed every n rounds
               'bagging_fraction': (0.2, 1),   # fraction of data sampled each bagging
               'feature_fraction': (0.2, 1),   # fraction of feature used each round
               }

optimizer = BayesianOptimization(lgbm_bays_opt, param_space)
optimizer.maximize(init_points=5, n_iter=50, alpha=1e-3)
```

Step 3: Use the best hyper-parameters combination from the last step to train the model and save the tuned model.

```
best_params = optimizer.max['params']
# evaluation = False will return the model itself without training and evaluation
clf = lgbm_bays_opt(**best_params, evaluation=False)
clf.fit(train_X, train_y)

# save the model
import datetime
clf.booster_.save_model('lgbm_{}.txt'.format(datetime.datetime.now().strftime('%Y_%m_%d_%H_%M_%S')))
```

XGBoost

We use the random search method in tuning the XGBoost model. We first started with creating a XGB Classifier model and setup the hyperparameter searching space.

```
# Create a XGBoost Model
classifier = xgboost.XGBClassifier()
# Set hyperparameters for Random Search
params = {
    'learning_rate' : [0.05,0.10,0.20,0.30],
    'max_depth' : [ 3, 6, 10, 15],
    'n_estimators': [300, 500],
    'colsample_bytree' : [ 0.3, 0.7 ],
}
```

Then use “roc_auc” as the scoring method to execute the search, and save the tuned model.

```
folds = StratifiedKFold(n_splits=3, shuffle=True, random_state=2)

search_xgb=RandomizedSearchCV(classifier, param_distributions=params,
                              cv=folds, n_iter=20, scoring="roc_auc", verbose=3)
# Execute search
result_xgb=search_xgb.fit(xtrain,ytrain)
best_model_xgboost = result_xgb.best_estimator_
```

Logistic regression

We followed similar steps in developing the Logistic Regression model. We started with creating the model pipeline and the hyperparameter searching space.

```
# Pipeline created using Logistic Regression
pipeline_lr = make_pipeline(StandardScaler(),
                      LogisticRegression(penalty='l2', solver='lbfgs', max_iter=10000, random_state=1))
# Create the parameter grid
param_grid_lr = [{'logisticregression__C': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 10.0, 100.0, 1000.0]}]
```

Then use random search to get the optimal result, and save the tuned model.

```
folds = StratifiedKFold(n_splits=3, shuffle=True, random_state=2)

search_lr = RandomizedSearchCV(pipeline_lr, param_grid_lr, scoring='roc_auc', cv=folds, verbose=1)
# Execute search
result_lr = search_lr.fit(X, y)
# Get the best performing model fit on the whole training set
best_model_lr = result_lr.best_estimator_
```

Predicting

Now we have got the tuned models with different algorithms, the next step is to use them to make predictions on our test set. Upon submission we found that our tuned XGBoost and LightGBM models are relatively better than other models (see Appendix. Model Results), with AUC score of 0.783 and 0.780 on test set respectively.

However, to further improve the results by combining the merits of our best models, we also explored ensembled ideas such as stacking (training a logistic regression model on top of the final results) and averaging (combining the results by certain proportions). The results of stacking, however, weren't as good as the individual models. The reason might be we don't have enough diverse models and the results were overfitting. However, we did achieve an improvement of 0.001 increase (not much) in AUC through the averaging method.

Below is the simple sample code from our predicting notebook for finding the optimal portion to blend the results from XGBoost and LightGBM, again with bayesian optimization.

```
# train_proba_lgbm is the predicted proba on training data by our lgbm model
# train_proba_xgb is the predicted proba on training data by our xgb model
# here we use percentage rank to first normalize the results since these
# two models have different variance in output. This will not affect AUC
train_proba_lgbm = train_proba_lgbm.rank(pct=True)
train_proba_xgb = train_proba_xgb.rank(pct=True)

# define the objective function that take the portion and evaluate auc on training data
def portion_opt(p):
    # Use the input p to blend the results
    train_proba = train_proba_lgbm * p + train_proba_xgb * (1-p)
    return roc_auc_score(train_y, train_proba)

param_space = {'p': (0, 1)}

optimizer = BayesianOptimization(portion_opt, param_space)
optimizer.maximize(init_points=3, n_iter=10, alpha=1e-4)
```

Here we first use percentage rank to normalize the probability predicted by both models on the same training data. This will make sure the different predicting variance doesn't affect the blending. Then we define an objective function and utilize the optimizer to find the best

proportion for blending the results. The optimizer found 72% XGBoost and 28% LightGBM is the perfect blend on the training data, so we used this proportion when on the test data.

The prediction probabilities went through the same normalization and blending steps and resulted in a 0.001 increase in final AUC. This is not much increase but gives a general idea on how we can potentially further improve by including more results from good models.

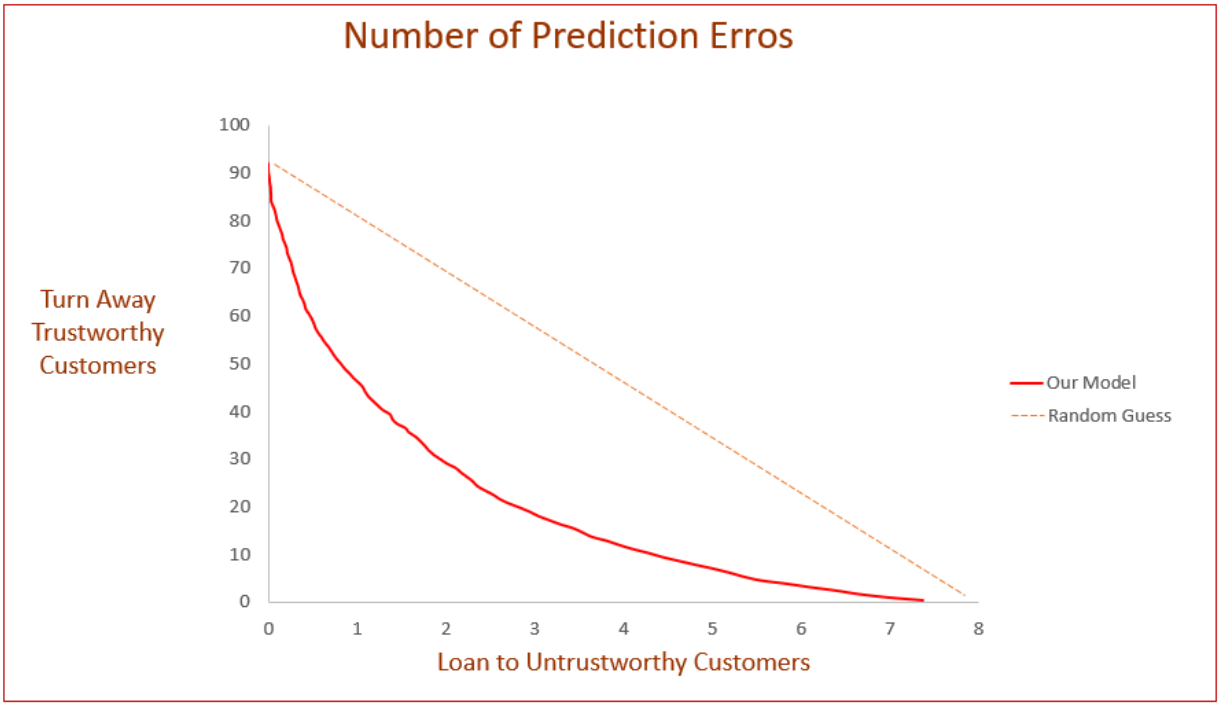
Results and Business Insights

Evaluations on Model Performance

To make a prediction, there's a risk of making two kinds of incorrect decisions: one is FP: to wrongly predict untrustworthy customers into trustworthy, the other is FN: to distrust reliable customers. The first one will result in financial loss to the company, as money would be loan to the group of customers that are less likely to repay; the later one, however, may not bring financial loss but will turn those underserved but trustworthy customers away, which contradicts the goal of Home Credit, and in the long run may be detrimental to the company reputation.

To provide a better illustration to the company on how to implement our model, and some more intuitive business insights into the model performance, we created a graph based on the rate of making each of these two kinds of mistakes. Since the model is used for predicting the probability of each customer's credibility, the company needs to set a percentage threshold on how much (from 0 to 1) should be considered as trustworthy.

In a sample group containing 100 customers, 92 are trustworthy customers and 8 are untrustworthy. From the figure shown below, while implementing our model, if the company wants to make no "False Negative" prediction (turn away trustworthy customers), there'll be 8 "False Positive" predictions (loan to untrustworthy customers). On the contrary, the number of false negative predictions will be as high as 92 if the company wishes to eliminate false positive predictions. But if the company chooses a threshold that will make 4 FP predictions (precision = 4/8 = 50%), there'll only be 12 FN predictions (precision = 12/92 = 13%). Overall, the likelihood of making wrong predictions is significantly lower than random guesses. Home Credit should decide based on their business strategies on how to interpret the model and set corresponding thresholds for predicting.

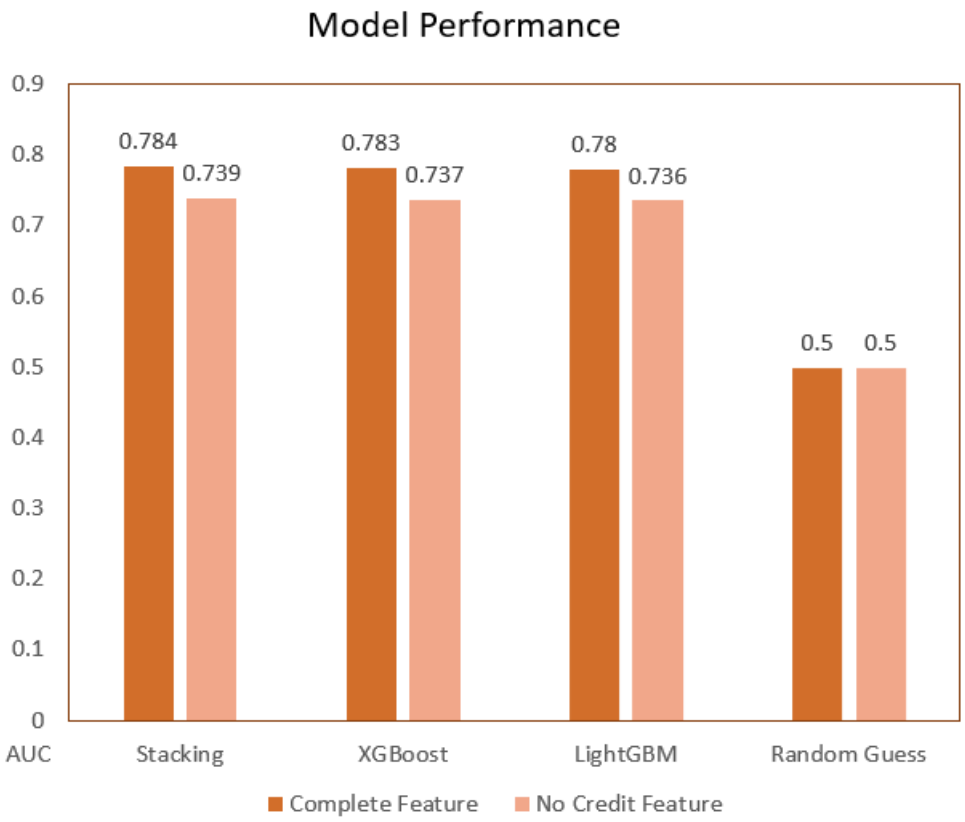


Evaluations on Predicting Non-History Customers

For this project, to evaluate the accuracy of our prediction model, we're using a metric called AUC, which is a useful tool for evaluating the performance of a prediction model that classifies results into "yes" or "no" categories, in this case, whether or not a customer is trustworthy in credit behaviors. AUC stands for "the Area Under ROC", and ROC is a graphical representation of making correct predictions versus mistakenly predicting untrustworthy customers into trustworthy, which means, the higher the AUC, the better our prediction performance will be.

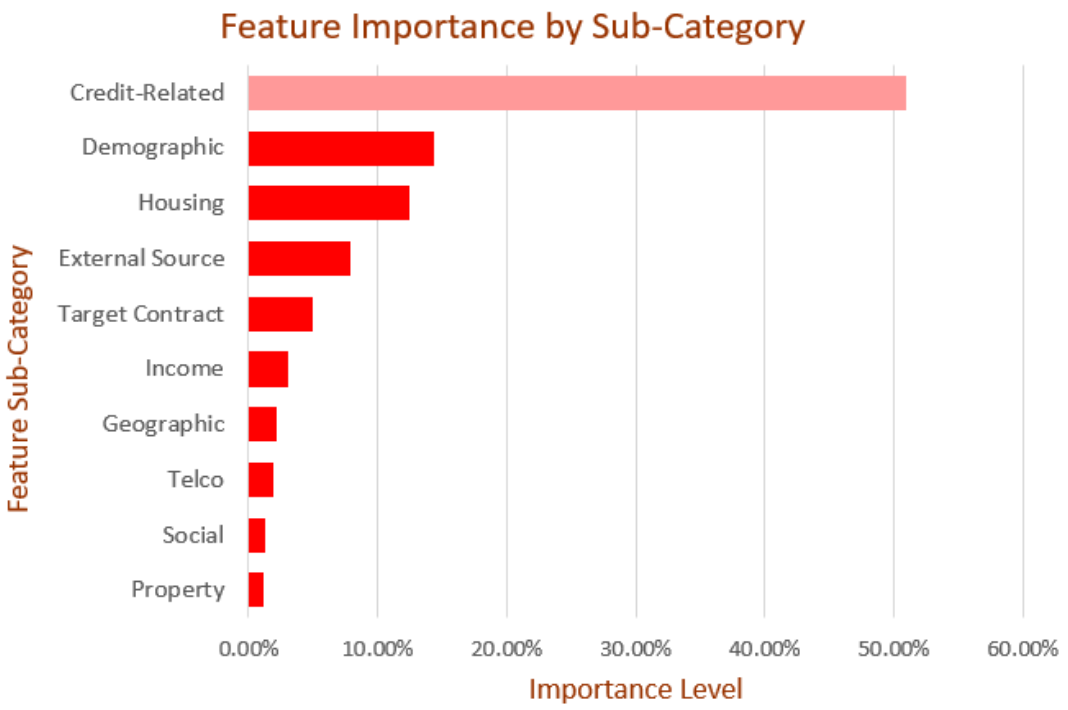
The dataset provided by Home Credit included both customers with or without previous credit history. To evaluate the performance of our prediction on those customers with no previous credit history, we ran the model again on this subgroup of people, based on the models we have developed so far. As the final result, we have achieved a 0.784 AUC with our best model on all customers, and in general, for those who don't have credit info, we can still get 94% of the performance compared to those who have a credit history.

This shows that our model has increased the chance of successfully predicting customer behaviors, and it could work properly for non-history customers as well, saving loss from frauds.

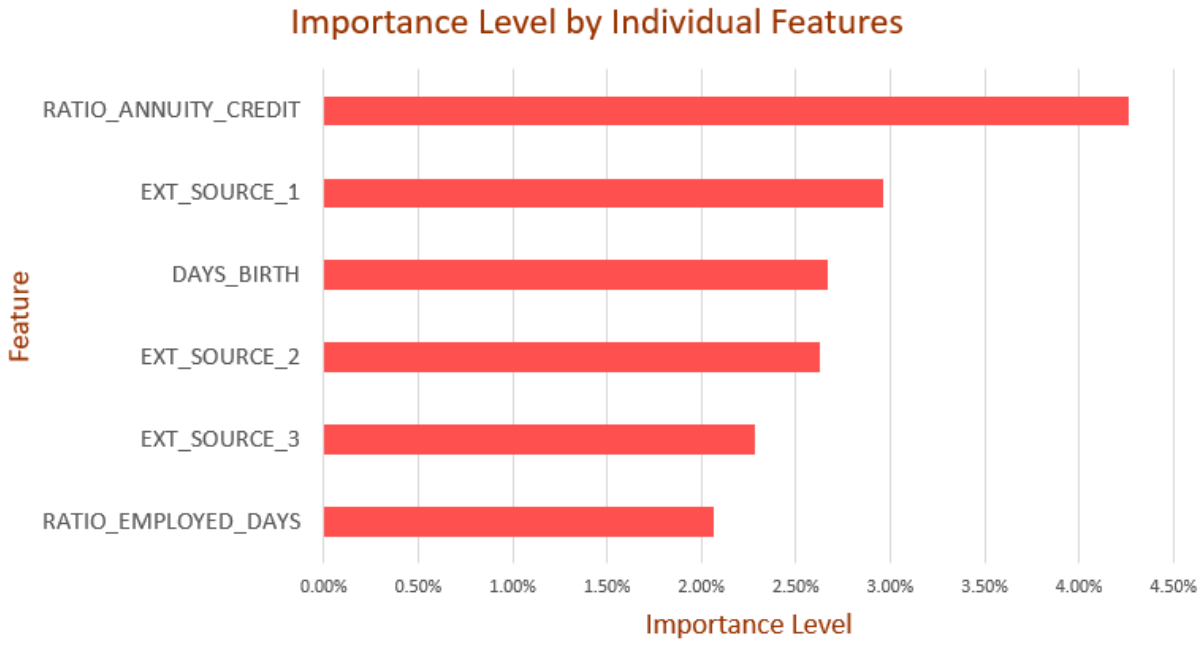


Feature Importance

From our best-performing model (a combination of XGBoost and LightGBM), we extracted the importance level of each variable to discover the features that contribute the most to the prediction process. We have classified the features into their representative sub-categories, such as demographics, credit history-related features, housing, income, etc. The figure below shows the importance level by these subcategories.



From the figure above we can see that for customers with previous credit history, their credit-related data play the most significant role in predicting. While for those with no credit history, demographics, housing and some external sources are the most important. Then let's take a deeper dive into the level of importance by individual features (shown below).



The most contributing feature RATIO_ANNUITY_CREDIT is the one we have created. It is the ratio of the amount of credit issued by Home Credit divided by the number of annuity payments, which refers to the number of terms in the current loan plan. The other five features in the graph are all non-credit-related, which can be applied to customers with no credit history. We do not have clear information on features from external sources. Home Credit may need to interpret these features from their perspective. DAYS_BIRTH refers to the age of the applicant, and RATIO_EMPLOYED_DAYS is also another feature created by us, meaning the ratio of days of employment and applicant's age. When making predictions on new coming customers, Home Credit could potentially prioritize these important features as they have more predictive value, to make better decisions.

Limitations in Deployment

There are two major limitations that might exist in actual deployment:

1. Since there are a lot of null values in the dataset, we have implemented different methods to fill these data, such as filling with zero or median in different scenarios, based on value distribution in each column. Some of the methods may not be suitable in real-world scenarios.
2. We have done a variety of feature engineering based on our understanding of the business problem. We didn't fully use all the data sources that were provided by Home Credit, as we consider some of the features are not intuitive enough for the analysis. The limitation of our domain knowledge may hinder the model from reaching its full potential.

Appendix

Model Results

AUC Score in each stage of feature engineering using the default XGBoost classifier. The more information included, the better the results.

Features Engineering Steps	Model	AUC Score
Application Original Features	Default XGBoost	0.743
`+ Calculate Ratios from Original	Default XGBoost	0.751

`+ Features Engineered from Bureau Data	Default XGBoost	0.757
`+ Features Engineered from Credit Card	Default XGBoost	0.764
`+ Features Engineered from POS Cash	Default XGBoost	0.772

After finishing feature engineering, we tried out tuning various models, here's the top models we got:

Model	Best Params	Best AUC Score
Stacking	Combined results from XGBoost: 72% LightGBM: 28%	0.784
XGBoost	n_estimators: 500 max_depth: 6 learning_rate: 0.05 colsample_bytree: 0.3	0.783
LightGBM	num_iterations: 1201 learning_rate: 0.057 num_leaves: 8 max_depth: 3 min_data_in_leaf: 20 pos_bagging_fraction: 0.91 neg_bagging_fraction: 0.09 bagging_freq: 3 feature_fraction: 0.177 min_gain_to_split: 0.032 is_unbalance: 1	0.78
Logistic Regression	penalty='l2' solver='lbfgs' C=10	0.753
Random Forest	max_depth: 20 min_samples_leaf: 10 n_estimators: 1001	0.741

File System

Raw Data Files

HomeCredit_columns_description.csv - Data description file for raw data files
application_train.csv - Main training data with original features and labels
application_test.csv - Testing data with original features
bureau.csv - Bureau data at the point of each loan application. Bureau features are engineered from this file
credit_card_balance.csv - Credit card balance data for each month for each applicant. Credit Card features are engineered form this file
POS_CASH_balance.csv - Credit card transactions and loan repayment data for each month for each applicant. POS Cash features are engineered from this file

Generated Data Files

training.csv - Training data generated from Feature Engineering.ipynb, contains all features and labels needed for training the models
testing.csv - Training data generated from Feature Engineering.ipynb, contains all features needed for making final predictions
xgb_predictions.csv - Predictions from XGBoost model generated by Predicting.ipynb
lgbm_predictions.csv - Predictions from LightGBM model generated by Predicting.ipynb
final_predictions.csv - The final predictions from combined results of xgb_predictions.csv and lgbm_predictions.csv, generated by Predicting.ipynb

Model Files

xgb.json - Contains the trained XGBoost model for predictions.
lgbm.txt - Contains the trained LightGBM model for predictions.

Code Files

Feature Engineering.ipynb - Contains code that takes raw data files and generates the training.csv and testing.csv for training and predictions
Modeling.ipynb - Contains code for model tuning and training, generates xgb.json and lgbm.txt
Predicting.ipynb - Contains code for making the final predictions. First uses the two models to make individual predictions, then find the best parameters to combine the prediction results from the two models to generate the final results.

References

[1] https://en.wikipedia.org/wiki/Home_Credit
[2] https://www.homecredit.net/about-us.aspx
[3] https://www.kaggle.com/c/home-credit-default-risk