Carlson School of Management
University of Minnesota

Home Credit Default Risk

Arkan Chatterjee, Beepa Bose, Emily Harvey, Francesca Rivera, Phuong Tran

CONTENTS

1. PROJECT DEFINITION & INTRODUCTION

1.1. Executive Summary

In this project, we predict whether or not Home Credit's clients have the ability to pay for the loans they are applying for. It is essential to be able to accurately predict this since Home Credit wants to maximize loan application acceptances for their clients who are truly able to repay so that they are empowered to be successful in their ventures. Our final model uses clients' various historical credit and application information fed into a stacking method, combining 5 different models with well-tuned hyper-parameters, resulting in an AUC score of 0.764. This means that there is a 76.4% chance that we are able to correctly predict whether or not the client has the ability to repay their loan. Having these predictions, we can reduce unnecessary rejections while still protecting the business from risk.

1.2. Background and Context

Home Credit is an international consumer finance provider, whose goal is to provide a safe experience for people who have little or no credit history. In order to create a positive borrowing experience for the customers, Home Credit utilizes a variety of data attributes to predict a customer' ability to repay loan. However, there are still many untapped data sources that could be employed to help Home Credit better serve its customers. By having a full understanding of its data, Home Credit could guarantee that the clients' ability to repay is not rejected and that loans are granted with principal and maturity.

2. DATASET & TECHNICAL SPECIFICATIONS

2.1. DATASET

The data files were taken from the Kaggle competition. Among all the data files, our team used the following to solve its business problem:

- *applicantion_train*: Application data for all applicants
- *bureau_balance*: Client's monthly balances of previous credits in Credit Bureau
- *previous_application:* Previous applications for Home Credit loans of clients who have loans

- *POS_cash_balance*: Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit
- *credit_card_balance:* Monthly balance snapshots of previous credit cards that the applicant has with Home Credit

2.2. TECHNICAL SPECIFICATIONS

The Data Platform we worked on was **Google Colab Notebook.**

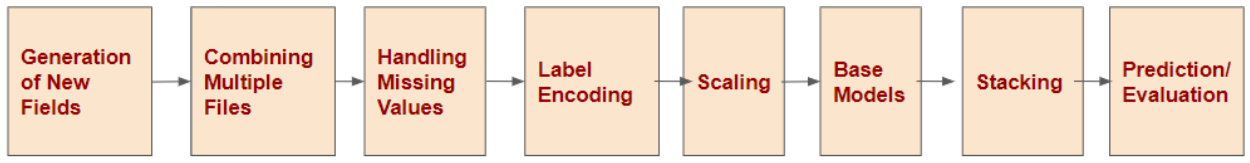
Specifications were:

```
GPU 0: Tesla K80 (UUID: GPU-205092bd-9e95-5853-64b8-b9b9d7ec2ff6)
Model name:          Intel(R) Xeon(R) CPU @ 2.30GHz
Socket(s):           1
Core(s) per socket:  1
Thread(s) per core:  2
L3 cache:            46080K
CPU MHz:             2299.998
Available memory : 13GB
Available hard disk space: 39GB
```

We used Python as the programming language and the libraries we used were:

```
sklearn.model_selection: GridSearchCV, KFold, RandomizedSearchCV
sklearn.preprocessing: StandardScaler, LabelEncoder
sklearn.linear_model: LogisticRegression
lightgbm: LGBMClassifier
sklearn.ensemble: RandomForestClassifier, AdaBoostClassifier
XGBoost: XGBClassifier
```

3. METHODOLOGY OVERVIEW



The figure above gives a general overview of the methodology of our predictive analysis. We start by generating new fields through feature engineering. Several files are then combined using the new features. After everything is combined, the missing values are removed. We encode the categorical data in order to be used for prediction analyses. Next, it is necessary to scale the data since several columns in the data are on different scales. We then train different base models to determine the best performing parameters and use these models with the ensemble modeling method of stacking. After combining these, we use the model to predict the clients' ability to repay with the testing data and submit these predictions to Kaggle to determine the model's final score. The steps of our predictive analysis are explained in further detail in the following section.

4. ANALYSIS

4.1 Data Preprocessing

4.1.1. Field Generations

We generated some new fields to add on to the primary application_train file:

a. *Credit_overdue*: Flag assigned to determine whether the applicant has had an overdue payment previously. If this has a value > 0, then there is some risk of late payments from this particular client.

b. *Debt_credit_ratio*: For each line of client's previous credit,

$$\frac{total\ debt}{total\ credit}$$

If this has a value > 1, then it can serve as a flag for at-risk customers that have incurred debt well over their credit limit.

c. *Success_ratio*: For every SK_ID_CURR,

$$\frac{\#successful\ applications}{\#\ total\ applications}$$

A ratio closer to 1 would signify that for the most part, this client has had successful loan applications in the past.

d. *Rejection_ratio*: For every SK_ID_CURR,

$$\frac{\#rejected\ applications}{\#\ total\ applications}$$

A ratio > 0.5 would signify that this client has had more rejections than successful loan applications in the past. A higher value could be a useful feature to flag rejection of current applications.

e. Avg_days_bet_rejection: Average days between each previous application rejection. A smaller value would indicate that the client has had more recent application rejections which could serve as a flag for determining that they are still not qualified for a loan.

f. *CreditCard Balance*: Average credit card balance per SK_ID_CURR

g. *Credit_Card_Limit:* Average credit card Limit balance per SK_ID_CURR

h. Avg_vs_Min_Pay: Average payment compared to minimum required payment. This feature would indicate if, on average, the client makes payments above the established or contracted minimum payment amount.

4.1.2. Handling of Missing Values

For all the stringtype fields, the missing values were replaced by N/A, for integer fields they were replaced by 0 and for float type fields they were replaced by 0.0.

4.1.3. Handling of Categorical Data

While Random Forest does not have trouble handling categorical data, Logistics Regression, LightGBM and XGBoost need the data to be in numerical format. So we did a numerical encoding of the categorical data using LabelEncoder.

LabelEncoder is a simple approach which involves converting each value in a column to a number. For example in our dataset,

CODE_GENDER	Label Encoder
F	1
M	2

One Hot Encoder on the other hand, removes the integer encoded variable and adds a new binary variable for each unique integer value. For example:

CODE_GENDER	M	F
F	0	1
M	1	0

Our chosen dataset has around 16 categorical columns with various values. Choosing one-hot encoding would result in a very sparse dataset. So we have chosen LabelEncoder. We also have experimented with One-hot-encoding followed by PCA and it gave us lower scores.

4.1.4. Scaling of Numeric Data

Many of our features are on different scales. When this happens, one feature can influence the outcome diminishing the effect of all other features. But when all features are in the same scale, it also helps algorithms to understand the relative relationship better. So we scaled the features using a StandardScaler. This scaler scales each input variable separately by subtracting the mean (called centering) and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one. It might have been risky to use Minmax as financial data might have outliers and it might make the majority of the data scaled down to 0. When we experimented with Minmax scaler the results matched the intuition.

4.1.5. Principal Component Analysis

We chose not to use PCA. Since, we are mostly choosing Ensemble methods like Random Forest, XGBoost, AdABoost and LightGBM. Ensemble methods do feature selection automatically. Using PCA on the other hand will give us the top n components which would give the highest variance in the data which not necessarily will have the highest discriminative power. In this case of our task being to come up with a binary classification model, having a model with excellent discriminative power is necessary. Moreover, since we have so many attributes we might end up having an enhanced effect of non-significant columns in the principal components as each principal component is a linear combination of all the original factors.

5. MODELING

5.1 Model 1: Logistic Regression

Starting things simple, we decided to fit a logistic regression model in order to set up a baseline from which we would further improve on by running different models as well as incorporating stacking.

To determine the best hyper-parameters to use for our model, we use Randomized Search with 5-fold cross validation to test different ranges C and whether to treat the data set balancing class weights or not.

- class_weight : balanced and None
- C: 0.01, 0.1, 1, 100, 1000

After tuning our model, the result shows that 'balanced' for class_weight and C = 100 are the best settings for our hyper-parameters.

The final score of Logistic Regression is 0.717

5.2 Model 2: Adaboost

Adaboost is a boosting algorithm in which multiple "weak" classifiers are combined into one "strong" classifier. The models are trained sequentially, by putting more weight to instances that are difficult to classify and less weight to those that are handled well.

We use Randomized Search cross validation to test different ranges of n_estimators, learning_rate, and different types of algorithm. Specifically, we tried:

- n_estimators: 50, 100, 150, 200, 300, 500
- learning_rate: 0.5, 1.0, 1.5
- algorithm: SAMME, SAMME. R

After tuning our model, the result shows that n_estimators = 500, learning_rate = 1.5, and SAMME for algorithm are the best settings for our hyper-parameters.

The final score of AdaBoost is 0.668

5.3 Model 3: Random Forest

The next model that we came up with is Random Forest, which is a classification algorithm that consists of many decision trees. The algorithm uses bagging technique and feature randomness when building each tree, meaning it only uses a random subset of all attributes, then finds which feature of the subset has the highest information gain to calculate each split.

To determine the best hyper-parameters to use for our model, we use Randomized Search cross validation to test different ranges of max_depth, min_sample_split, min_sample_leaf, and n_estimators:

- Max_depth: 5 and 15
- Min_sample_split: 2, 5, and 10
- Min_sample_leaf: 1, 2, and 4
- N_estimators: 100 evenly spaced values from 200 to 1000, inclusively

After tuning our model, the result shows that max_depth = 15, min_sample_split = 5, min_sample_leaf = 4, and n_estimators = 507 are the best setting for our hyper-parameters.

The final score of Random Forest is 0.652

5.4 Model 4: LightGBM

The next base model we used was a LightGBM Model It is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with faster training speed and higher efficiency and lower memory usage. We thought it was a good model to work with in the case of our dataset because the primary dataset had around 307511 data rows with 122 fields.

To determine the best hyper-parameters to use for our model, we useGridSearch with a 5 fold cross validation to test different ranges of boosting_type, n_estimators, learning_rate, num_leaves:

- Boosting_type: 'gbdt', 'goss'
- N_estimators:1000, 5000,...., 20000,
- Learning_rate: 0.0005, 0.001, 0.01, 0.05, 0.1, 0.5
- num_leaves:10, 20, 30, 40, 50

The best parameters among these were: boosting_type: ‘goss’, n_estimators: 15000, learning_rate: 0.001, num_leaves: 30 and max_height: 8.

The final score of LGBM came to 0.68

5.5 Model 5: XGBoost

XGBoost optimizes the gradient boosting algorithm (which uses gradient descent to minimize errors) through tree pruning, handling missing values, parallel processing, and regularization to avoid overfitting.

To determine the best hyper-parameters to use for our model, we use Randomized Search with 5-fold cross validation to test different ranges of gamma, max_depth, and n_estimators:

- gamma : 0.1, 1, 10
- max_depth: 2, 7, 12, 17
- n_estimators: 200, 700, 1200, 1700

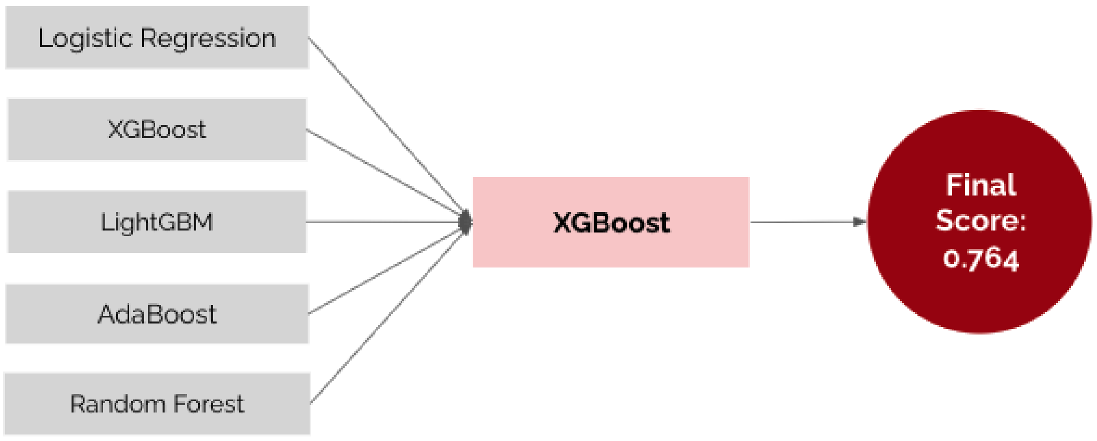
After tuning our model, the result shows that xgb__n_estimators = 700, xgb__max_depth = 2, and xgb__gamma = 0.1 are the best settings for our hyper-parameters.

The final score of XGBoost is 0.761.

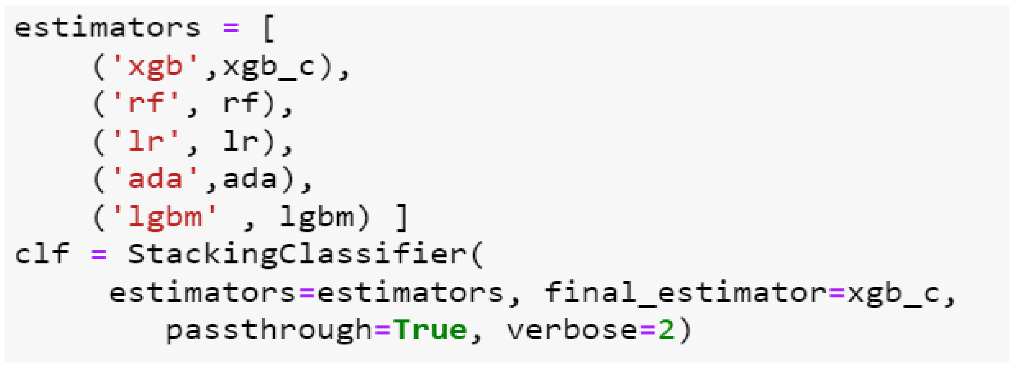
5.6 Stacking

In stacking we are stacking up Logistic Regression, XGboost, Adaboost, LightGBM and RandomForest as the base estimators using the best parameters that we got from running them individually. The final estimator is taking the predictions of the previously tuned models as input, flowing through another XGBoost model in order to conduct predictions.

This resulted in a final, and best performing, score of 0.764.



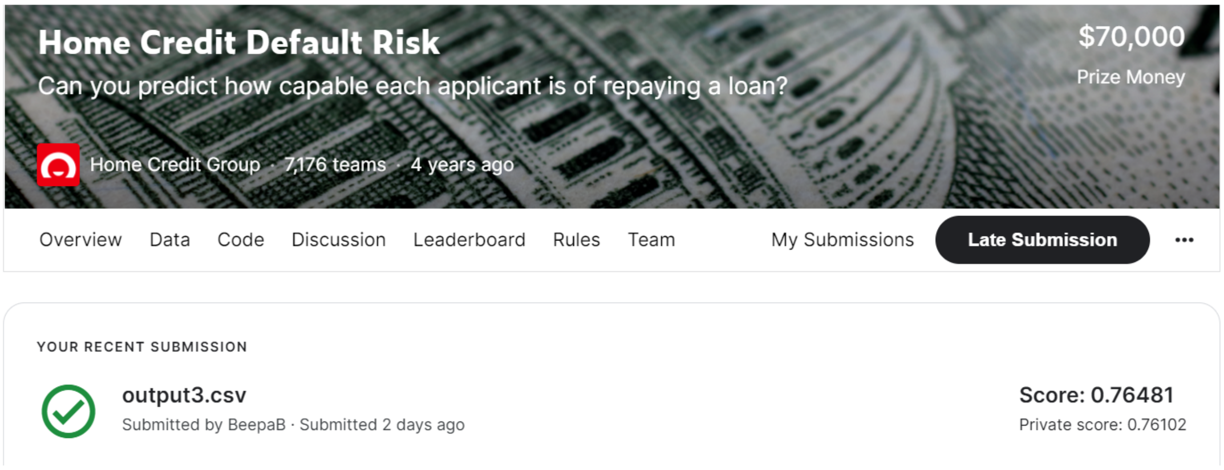
Code snippet : Here we can better see how the previous tuned models are fed into the final stacking classifier model.



```
estimators = [
    ('xgb',xgb_c),
    ('rf', rf),
    ('lr', lr),
    ('ada',ada),
    ('lgbm' , lgbm) ]
clf = StackingClassifier(
     estimators=estimators, final_estimator=xgb_c,
        passthrough=True, verbose=2)
```

6. RESULTS/ EVALUATION

We got our results by uploading the prediction probabilities file on the test dataset on Kaggle. Below is our final score:



Home Credit Default Risk
Can you predict how capable each applicant is of repaying a loan?
Home Credit Group · 7,176 teams · 4 years ago

$70,000
Prize Money

YOUR RECENT SUBMISSION

output3.csv
Submitted by BeepaB · Submitted 2 days ago

Score: 0.76481
Private score: 0.76102

Previously, Home credit had to go through the applications manually to identify whether a customer will be a loan defaulter or not. It was taxing, time-consuming, and also if not done correctly could lead to monetary losses for the company. The result above tells us that there is a 76.4% likelihood that our predictive model will better distinguish between the customers who are loan defaulters vs customers who are not.

Deploying this model gives Home Credit better decisive power when it comes to loan application acceptances or rejections. In line with their goal to support their clients through responsible loan grants, being able to better distinguish clients with the ability to pay will allow Home Credit to maximize successful loan applications that will further empower underserved clients financially. Reducing unnecessary rejections will also help foster the relationship between Home Credit and their retailers by improving the overall client experience.

7. OTHER EXPLORATIONS

Below is a summary of all other explorations we conducted while seeking to continue to improve our final model's predictive performance - from different preprocessing methods to including varying sets of features.

Model	Preprocessing	Data files	Score
Logistic Regression	LabelEncoder, StandardScaler	application_train only	0.7111

Logistic Regression	LabelEncoder, StandardScaler	application_train, credit_card_balance, previous_application_info	0.7152
Logistic Regression	LabelEncoder, StandardScaler	application_train, credit_card_balance, previous_application_info, bureau, POS_cash_balance	0.7246
XGBoost	LabelEncoder, StandardScaler	application_train, credit_card_balance, previous_application_info, bureau, POS_cash_balance	0.7615
LGBM	PCA(4)	application_train	0.588
LGBM	Without PCA OneHotEncoding	application_train	0.6114
LGBM	Without PCA LabelEncoder	Application_train, credit_card_balance,previous _application_info	0.6826
SVM	With PCA(4) OneHotEncoding	application_train	Took a very long time to execute so we stopped execution
KNN	With PCA(4) OneHotEncoding	application_train	0.54
Stacking	All 5 models Final Estimator - Logistic Regression	application_train, credit_card_balance, previous_application_info, bureau, POS_cash_balance	0.761
Stacking	All models except Random Forest Final Estimator -	application_train, credit_card_balance, previous_application_info,	0.76

	XGBoost	bureau, POS_cash_balance	
Stacking	All 5 models Final Estimator - XGBoost	application_train, credit_card_balance, previous_application_info, bureau, POS_cash_balance	0.764

8. LIMITATIONS AND FUTURE SCOPE

In our analysis there are several challenges and limitations that we encounter. It is difficult to determine which files and columns would result in the highest AUC score. There are also many ways to engineer features so we have to determine which features would provide the best results. In addition, several different predictive models could be used in our analysis. We chose to stack several models so that we can use the best predictions from each of these models in combination with each other. It is challenging to find which combination of these models used in parallel produces the best predictions.

There are several ways that we could expand on this analysis to try to achieve a higher AUC score. The following are some of the methods that could be utilized in further explorations:

1. Using more or fewer data files
2. Choosing different columns from the files
3. Create more features through feature engineering
4. Explore other predictive models
5. Try different combinations of models with stacking
6. Analyzing the cost benefit analysis of each of the models

REFERENCES:

1. https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/
2. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/
3. https://www.kaggle.com/competitions/home-credit-default-risk/overview