UNIVERSITY OF MINNESOTA
Driven to Discover℠

HOME CREDIT DEFAULT RISK

MSBA 6420: Predictive Analytics

Team: 5

Submitted by:

Anupam Shandilya

Jashyant Sikhakolli

Pahal Patangia

Pardha Pitchikala

Shiva Pabbathi

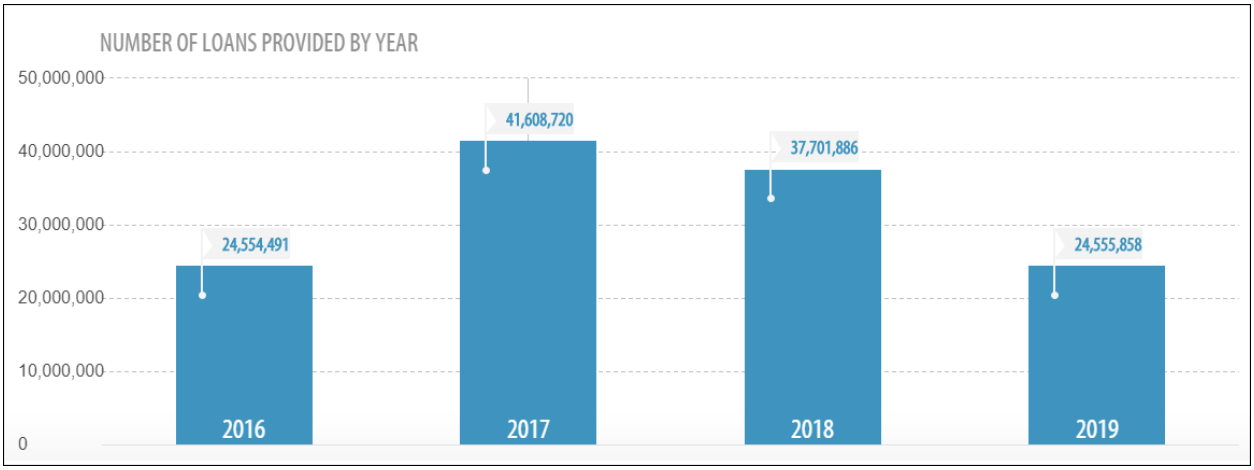
Zheming Lian

Table of Contents

Project Definition & Introduction

Background & Context

Founded in 1997, Home Credit Group is an international consumer finance provider with operations in 9 countries. Their aim is to provide consumer financial products that are easily accessible, affordable and manageable for their customers. They primarily focus on providing financial products to underserved borrowers i.e to people with little or no credit history. Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience; Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.



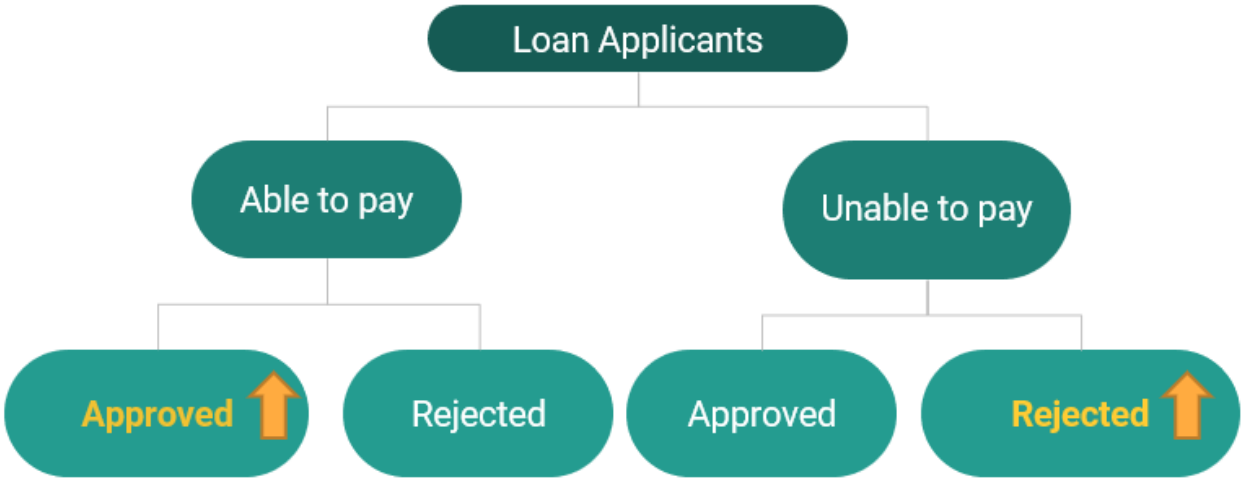
Yearly trend of Number of loans being approved

Problem Statement

The goal of this project is to improve the ability to predict credit default or repayment ability of their customers. This is of utmost importance because there is a declining trend in the number of loans being approved. In order to better the situation, Home Credit wants to enhance its current baseline model to better identify customers who might default which will lead to an increase in positive customer experience.

Home Credit care about the experience customers have during the loan application process. Because rejecting applicants (especially ones who are able to repay) may result in a negative experience that might impact the company's relationships with customers as well as retailers (they have partnered with). Another point of concern (especially for product and risk managers) revolves around customers who truly are a default risk. Extending loans and credit to people who cannot repay damages the

overall Home Credit business and goes against the company's mission of responsible lending and financial inclusivity for underserved ("unbanked") people.



As depicted in the above flow chart, the final objective of this project is to increase the Approvals for customers who can pay and increase the Rejections for customers who might default.

Data Sources Description

For this problem, we were provided with data from two sources. The first was the previous Home Credit loan application data for the customers, and the second was the Credit bureau history data for previous loans and credit card used by the customer. The data is divided into six tables. They are as follows:

1. Application Table - This is the main table. It contains static data for all applications. One row represents one loan in our data sample.
2. Bureau table - This table contains all client's previous credits provided by other financial institutions that were reported to Credit Bureau (for clients who have a loan in our sample). For every loan in our sample, there are as many rows as the number of credits the client had in the Credit Bureau before the application date.
3. Bureau balance table - This table contains the monthly balances of previous credits in the Credit Bureau. This table has one row for each month of the history of every previous credit reported to the Credit Bureau.
4. POS CASH balance table - Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit. This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample.

5. Credit card balance - Monthly balance snapshots of previous credit cards that the applicant has with Home Credit. This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample.
6. Previous application table - All previous applications for Home Credit loans of clients who have loans in our sample. There is one row for each previous application related to loans in our data sample.
7. Installment payments table - Repayment history for the previously disbursed credits in Home Credit related to the loans in our sample. One row is equivalent to one payment of one installment OR one installment corresponding to one payment of one previous Home Credit credit related to loans in our sample.

```
DATA_PATH = '/gdrive/Shared drives/Predictive_Project/data/'
application_test_df = pd.read_csv(DATA_PATH + 'application_test.csv',index_col=0)
application_train_df = pd.read_csv(DATA_PATH + 'application_train.csv',index_col=0)
bureau_df = pd.read_csv(DATA_PATH + 'bureau.csv',index_col=1)
bureau_balance_df = pd.read_csv(DATA_PATH + 'bureau_balance.csv')
installments_payments_df = pd.read_csv(DATA_PATH + 'installments_payments.csv')
POS_CASH_balance_df = pd.read_csv(DATA_PATH + 'POS_CASH_balance.csv')
previous_application_df = pd.read_csv(DATA_PATH + 'previous_application.csv')
sample_submission_df = pd.read_csv(DATA_PATH + 'sample_submission.csv')
credit_balance_df = pd.read_csv(DATA_PATH + 'credit_card_balance.csv',index_col=0)
```

Analysis

Methodology

The first step was to understand the business problem and the data associated with it. Once we got a good understanding of the data and the problem, we proceeded to clean the data. Using domain knowledge and some research we identified the columns in which we can impute missing values with “ZERO”. Next, we went on to create various relevant features that will help achieve our objective. On a combined list of both derived and raw features, we used statistical techniques to reduce the feature set. After the feature engineering & reduction part, we started the core modeling process. Multiple algorithms were implemented on the feature set, along with cyclical addition and removal of features depending on the performance and complexity of features and the model used. Correlated features that are obvious were removed. Since the data was imbalanced, we got good results from LGBM compared to other models we tried. To develop models, we chose MCC as our metric to compare models and for kaggle score we used AUC. Later, the model with the best performance was chosen based on MCC and the same model got the highest AUC among others. As expected, this gave the best reduction in LOSS for the company. But the only limitation with this was that the model was uninterpretable.

Feature Engineering

The raw features present in the data do not prove to be as effective in determining the creditworthiness of the customer. Moving values across months and static demographic variables are good only when we consider a one-dimensional view of the data.

We have created derived and aggregated features for raw fields like balances, loan value & income information, days past due information etc. For eg. a loan value of $1000 and $5000 does not tell much about the risk of two customers, but a variable like loan to income ratio is a good indicator of customer's ability to repay. Besides, we have aggregated the monthly fields to maximum/minimum/average values to capture the risk of the customer as per domain knowledge.

For feature engineering, we first merged all the tables to get a final table.

We performed following aggregations in the tables before merging them:

For Bureau table:

```
# Bureau and bureau_balance numeric features
bureau_agg_dict = {
        'DAYS_CREDIT': ['max'],
        'CREDIT_DAY_OVERDUE': ['max'],
        'DAYS_CREDIT_ENDDATE': ['max'],
        'DAYS_ENDDATE_FACT':['max'],
        'AMT_CREDIT_MAX_OVERDUE': ['max'],
        'CNT_CREDIT_PROLONG' : ['max'],
        'AMT_CREDIT_SUM': ['max', 'mean', 'sum'],
        'AMT_CREDIT_SUM_DEBT': ['max'],
        'AMT_CREDIT_SUM_OVERDUE': ['max'],
        'AMT_CREDIT_SUM_LIMIT': ['max'],
        'AMT_ANNUITY': ['max', 'mean'],
        'CNT_CREDIT_PROLONG': ['sum']
    }
```

For installments table:

```
# installments
installments_payments_df['PAYMENT_MISS_FLAG'] = 0
installments_payments_df.loc[installments_payments_df['AMT_PAYMENT'] > installments_payments_df['AMT_INSTALMENT'] + 10,'PAYMENT_MISS_FLAG'] = 1
installments_agg_dict = {
       'PAYMENT_MISS_FLAG': ['max']
   }
```

For POS CASH table:

```
# POS_CASH_balance
POS_agg_dict = {
        'SK_DPD_DEF': ['max']
    }
```

For previous application

```
# Previous Application
prev_app_agg_dict = {
        'AMT_CREDIT': ['min']
    }
```

After performing aggregations on the data set, we merged the tables to get our final training and test data.

```
# final merged dataset
train = application_train_df.merge(bureau_agg,how='left',on='SK_ID_CURR')\
                            .merge(installments_agg,how='left',on='SK_ID_CURR')\
                            .merge(POS_agg,how='left',on='SK_ID_CURR')\
                            .merge(credit_balance_agg, how='left',on='SK_ID_CURR')

test = application_test_df.merge(bureau_agg,how='left',on='SK_ID_CURR')\
                            .merge(installments_agg,how='left',on='SK_ID_CURR')\
                            .merge(POS_agg,how='left',on='SK_ID_CURR')\
                            .merge(credit_balance_agg, how='left',on='SK_ID_CURR')
```

Treatment of categorical data: We used one-hot encoding for converting our categorical variables into dummy variables.

```
#one-hot encoding
#input: features (no label), test_features

#remove the label before encoding
labels = train['TARGET']
features = train.drop(columns = ['TARGET'])
features = pd.get_dummies(features)
test_features = pd.get_dummies(test)
#test_features.columns
# Align the dataframes by the columns
features, test_features = features.align(test_features, join = 'inner', axis = 1)
```

There were several variables with missing values. The reason for them not being populated is whether the account is closed, or that field does not pertain for the customer on record. For example - If a customer closes the account, his days past due, balance etc. information will not be populated in the system and hence missing values so that is as good as 0. Same is for other variables. For fields like owned house area, where they are missing, if the user has no owned house the associated fields will be blank that is also as good as 0 area, 0 rooms. So, we replaced these fields with zero.

```
def missing_data(data):
    total = data.isnull().sum().sort_values(ascending = False)
    percent = (data.isnull().sum()/data.isnull().count()*100).sort_values(ascending = False)
    return pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
#pd.set_option('display.max_rows', 300)
```

```
treatment_list = list(missing_data(features).Percent[missing_data(features).Percent>0].index)
features[treatment_list] = features[treatment_list].replace({np.nan:'0'})
#do the same thing for test data
test_treatment_list = list(missing_data(test_features).Percent[missing_data(test_features).Percent>0].index)
test_features[test_treatment_list] = test_features[test_treatment_list].replace({np.nan:'0'})
# Ensuring if replacement worked - should have zero missing values now
```

For other important variables which had null values we dropped the data with null values.

Next, was scaling of variables. We used a Min-Max scaler for scaling our variables.

```
### scaling

import numpy as np
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
train_scaled = pd.DataFrame(scaler.fit_transform(features), columns=features.columns,index = features.index)
test_scaled = pd.DataFrame(scaler.transform(test_features), columns=test_features.columns)
train_scaled2 = pd.concat([labels,train_scaled],axis = 1)
```

Feature reduction

Feature reduction is a significant process for a predictive project in the context of high dimensional data. It helps the model to get better performance on a broader scope of data in the real world, rather than just learning the pattern specific to the training data. It also reduces unnecessary computation cost. For this project, we conducted a feature reduction with three different "voters". These voters are either statistical tests or machine learning algorithms. Each of them assigned a significant score to each feature based on some standards. It then votes for features whose importance score is above some specific thresholds. After that, we select those features that are voted by any voters.

Why do we do feature reduction?

Without an appropriate process of feature selection, the high dimensionality of data brings in a collection of phenomena that are commonly considered as the "curse of dimensionality". In general, high dimensional data raises the risk of overfitting, which significantly affects the out-of-sample performance of the model. Moreover, although the sparsity of feature space makes data separable and makes classification easier, beyond some threshold the increase in dimension provides less marginal value than the overhead it adds to the algorithm. Feature reduction, therefore, becomes an important part of this predictive project.

Overview of the feature reduction process

Three round of feature reduction was conducted to rank the importance of features based on different techniques. With each round of selection, feature ranked above a threshold was picked up. In the end, features that were selected by at least two different techniques have remained in the final model.

Details of each feature reduction techniques

a. Pearson-Correlation

Pearson-Correlation is a measure of the linear correlation between two numerical variables. A function was built to rank the importance of each feature based on the Pearson-correlation with a label for each feature. In this round of feature reduction, features ranked in the top 100 were picked up. Even though it might be unfair to numerical features that one-hot encoded features were involved in this process, this technique had a moderate consistency with other techniques when selecting features. (show the number)

b. Chi-square

A Chi-square test measures dependence between stochastic variables, where a higher score reflects higher dependence between variables. With the chi2 function from "**sklearn.feature_selection**" package, a chi-squared stats between feature and label were computed to rank the importance of each feature. Since the chi-squared test only takes non-negative values as input, a min-max scaling was conducted beforehand to scale each variable between 0 and 1. In this round of feature reduction, features ranked in the top 100 were selected.

c. Random Forest

The random forest consists of several decision trees. Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar label end up in the same set. The measure based on which the (locally) optimal condition is chosen is called impurity. With the SelectFromModel function from **"sklearn.features_selection"** package, a random forest model was built to calculate the importance of feature based on its contribution to the decrease in the impurity of data on average. In this round, features with importance above the 1.25 times of the median value of importance were selected.

Below is a sample of features that are selected by the approach we described above:

	Feature	Pearson	Chi-2	Random Forest	Total
1	YEARS_BUILD_MODE	True	True	True	3
2	YEARS_BUILD_MEDI	True	True	True	3
3	YEARS_BUILD_AVG	True	True	True	3
4	YEARS_BEGINEXPLUATATION_MODE	True	True	True	3
5	YEARS_BEGINEXPLUATATION_MEDI	True	True	True	3
6	YEARS_BEGINEXPLUATATION_AVG	True	True	True	3
7	TOTALAREA_MODE	True	True	True	3
8	REG_CITY_NOT_WORK_CITY	True	True	True	3
9	REG_CITY_NOT_LIVE_CITY	True	True	True	3
10	REGION_RATING_CLIENT_W_CITY	True	True	True	3
11	REGION_RATING_CLIENT	True	True	True	3
12	REGION_POPULATION_RELATIVE	True	True	True	3
13	ORGANIZATION_TYPE_Self-employed	True	True	True	3
14	ORGANIZATION_TYPE_Business Entity Type 3	True	True	True	3
15	OCCUPATION_TYPE_Sales staff	True	True	True	3
16	OCCUPATION_TYPE_Laborers	True	True	True	3
17	OCCUPATION_TYPE_Drivers	True	True	True	3
18	NAME_INCOME_TYPE_Working	True	True	True	3
19	NAME_HOUSING_TYPE_With parents	True	True	True	3
20	NAME_HOUSING_TYPE_House / apartment	True	True	True	3

Modeling

To get started, we used basic algorithms to give a baseline of performance, they are:

- Logistic regression:
  We set up logistic regression with regularization.

  Code Snippet:

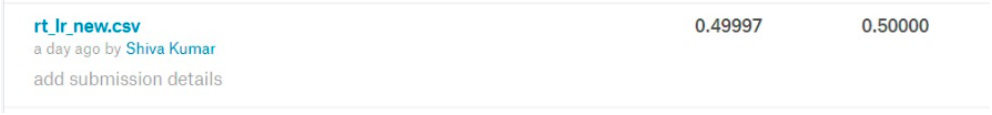
```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.3, random_state=0)

lr= LogisticRegression(max_iter=100,random_state=0,solver='saga',penalty='elasticnet',l1_ratio=0.5)
lr.fit(X_train, y_train)
y_pred_rt = lr.predict_proba(X_val)[:, 1]

y_pred_prob = lr.predict(X_test)

a=pd.DataFrame()
a['TARGET'] =  y_pred_prob
a.index=X_test.index
a.to_csv('rt_lr_new.csv')
```

  Test AUC for public leaderboard is 0.499, whereas the test AUC for private leaderboard is 0.5.



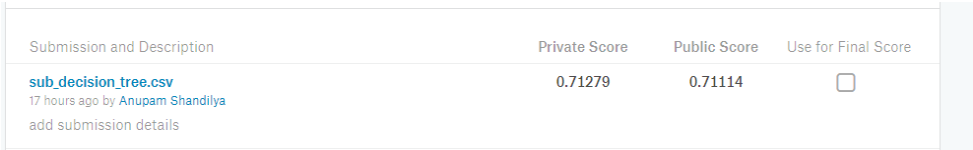
- Random Forest:
  We set up the number of estimators as 50, the criteria of each split as Gini index, and the maximum depth of tree as 5.

  Code Snippet:

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(
    n_estimators=50,
    criterion='gini',
    max_depth=5,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features='auto',
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    min_impurity_split=None,
    bootstrap=True,
    oob_score=False,
    n_jobs=-1,
    random_state=0,
    verbose=0,
    warm_start=False,
    class_weight='balanced'
)
clf.fit(x_train,y_train)
clf.predict(x_test)
y_pred=clf.predict_proba(x_test)

a=pd.DataFrame()M
a['TARGET']=y_pred[:,1]
a.to_csv('sub_decision_tree.csv')
```

  The test AUC score is 0.71.



Submission and Description	Private Score	Public Score	Use for Final Score
sub_decision_tree.csv 17 hours ago by Anupam Shandilya add submission details	0.71279	0.71114	☐

Given that the highest score on the leaderboard is over 0.8, we think basic models are not promising. Our interpretation was that these models are insensitive to the imbalance in class label of data. (0.94 : 0.06) Therefore, we decided to use some advanced models as follows:

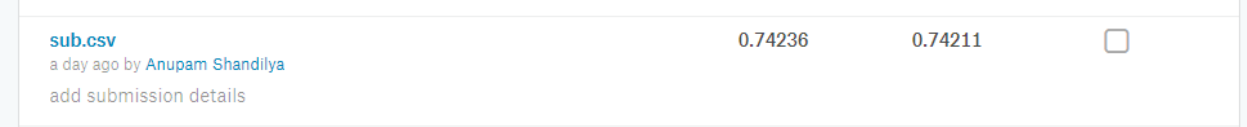
- XGBoost

For XGBoost, we selected several estimators as 1000, and ask algorithm to stop early if the model performance has no improvement in the past 10 rounds.

```
clf = XGBClassifier(n_estimators=1000, objective='binary:logistic', gamma=0.1, subsample=0.5 )
clf.fit(X_train, y_train, eval_set=[(X_val, y_val)], eval_metric='auc', early_stopping_rounds=10)
```

```
y_pred=clf.predict_proba(X_test_final)
y_pred=clf.predict_proba(X_test_final)
a=pd.DataFrame()
a['SK_ID_CURR']=X_test_final.index
a['TARGET']=y_pred[:,1]
a.to_csv('sub.csv')
```

The test AUC is 0.742

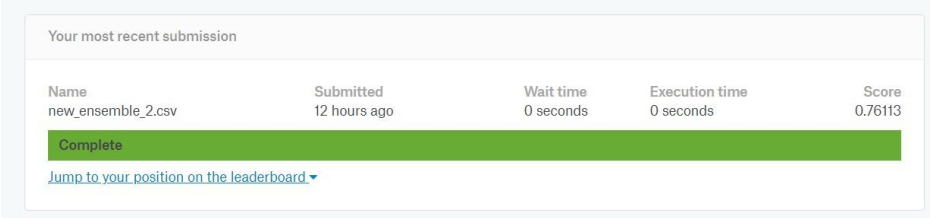


- XGBoost+LGBM:
  A stacked model is built with XGBoost and LGBM. For XGBoost, number of estimators is set as 550 and the threshold of early stopping is set to 10. For LGBM, the threshold of early stopping is set to 50.

  Code Snippet:

```
xgb = XGBClassifier(n_estimators=550, objective='binary:logistic', gamma=0.1, subsample=0.5 )
xgb.fit(X_train, y_train, eval_set=[(X_val, y_val)], eval_metric='auc', early_stopping_rounds=10)
y_predict = xgb.predict_proba(X_train)
b=pd.DataFrame()
b['SK_ID_CURR']=X_train.index
b['TARGET'] =  y_predict[:,1]
X_train = X_train.merge(b,on='SK_ID_CURR',how='left')
y_predict_val = xgb.predict_proba(X_val)
b_2=pd.DataFrame()
b_2['SK_ID_CURR']=X_val.index
b_2['TARGET'] =  y_predict_val[:,1]
X_val = X_val.merge(b_2,on='SK_ID_CURR',how='left')
train_data = lgb.Dataset(data=X_train, label=y_train)
valid_data = lgb.Dataset(data=X_val, label=y_val)
lgbm = lgb.train(params,train_data,2500,valid_sets=valid_data,early_stopping_rounds= 50,verbose_eval= 10)
```

  The test AUC is 0.761



- LGBM
  - Version 1: Used only those features that are interpretable the hyperparameter is set up as follows:

```
params = {'boosting_type': 'gbdt',
          'max_depth' : 5,
          'objective': 'binary',
          'nthread': 5,
          'num_leaves': 32,
          'learning_rate': 0.05,
          'max_bin': 512,
          'subsample_for_bin': 200,
          'subsample': 0.7,
          'subsample_freq': 1,
          'colsample_bytree': 0.8,
          'reg_alpha': 20,
          'reg_lambda': 20,
          'min_split_gain': 0.5,
          'min_child_weight': 1,
          'min_child_samples': 10,
          'scale_pos_weight': 1,
          'num_class' : 1,
          'metric' : 'auc'
          }
```

The test AUC for public leaderboard is 0.760, whereas the test AUC for private leaderboard is 0.757.

- Version 2: Used all the features to get the predictions

  For this model we modified our features. We used the combination of the possible features. In this model we use the mean, minimum and maximum, total and variance of different features. So that we can capture extra information from the interaction of these.

  I. Bureau and bureau balance table:

```
Bureau_bureauBalance_aggregations = {
    'DAYS_CREDIT': ['min', 'max', 'mean', 'var'],
    'DAYS_CREDIT_ENDDATE': ['min', 'max', 'mean'],
    'DAYS_CREDIT_UPDATE': ['mean'],
    'CREDIT_DAY_OVERDUE': ['max', 'mean'],
    'AMT_CREDIT_MAX_OVERDUE': ['mean'],
    'AMT_CREDIT_SUM': ['max', 'mean', 'sum'],
    'AMT_CREDIT_SUM_DEBT': ['max', 'mean', 'sum'],
    'AMT_CREDIT_SUM_OVERDUE': ['mean'],
    'AMT_CREDIT_SUM_LIMIT': ['mean', 'sum'],
    'AMT_ANNUITY': ['max', 'mean'],
    'CNT_CREDIT_PROLONG': ['sum'],
    'MONTHS_BALANCE_MIN': ['min'],
    'MONTHS_BALANCE_MAX': ['max'],
    'MONTHS_BALANCE_SIZE': ['mean', 'sum']
}
```

Ii. Previous application features

```
previous_application_aggregations = {
    'AMT_ANNUITY': ['min', 'max', 'mean'],
    'AMT_APPLICATION': ['min', 'max', 'mean'],
    'AMT_CREDIT': ['min', 'max', 'mean'],
    'APP_CREDIT_PERC': ['min', 'max', 'mean', 'var'],
    'AMT_DOWN_PAYMENT': ['min', 'max', 'mean'],
    'AMT_GOODS_PRICE': ['min', 'max', 'mean'],
    'HOUR_APPR_PROCESS_START': ['min', 'max', 'mean'],
    'RATE_DOWN_PAYMENT': ['min', 'max', 'mean'],
    'DAYS_DECISION': ['min', 'max', 'mean'],
    'CNT_PAYMENT': ['mean', 'sum'],
}
```

Iii. POS CASH

```
# features
pos_cash_aggregations = {
    'MONTHS_BALANCE': ['max', 'mean', 'size'],
    'SK_DPD': ['max', 'mean'],
    'SK_DPD_DEF': ['max', 'mean']
}
```

iv) Installment payment

```
installment_payment_aggregations = {
    'NUM_INSTALMENT_VERSION': ['nunique'],
    'DPD': ['max', 'mean', 'sum'],
    'DBD': ['max', 'mean', 'sum'],
    'PAYMENT_PERC': ['max', 'mean', 'sum', 'var'],
    'PAYMENT_DIFF': ['max', 'mean', 'sum', 'var'],
    'AMT_INSTALMENT': ['max', 'mean', 'sum'],
    'AMT_PAYMENT': ['min', 'max', 'mean', 'sum'],
    'DAYS_ENTRY_PAYMENT': ['max', 'mean', 'sum']
}
```

For this model we used the following features:

```
lgbmclf = LGBMClassifier(
    nthread=4,
    n_estimators=10000,
    learning_rate=0.02,
    num_leaves=34,
    colsample_bytree=0.9497036,
    subsample=0.8715623,
    max_depth=8,
    reg_alpha=0.041545473,
    reg_lambda=0.0735294,
    min_split_gain=0.0222415,
    min_child_weight=39.3259775,
    silent=-1,
    verbose=-1, )

lgbmclf.fit(train_x, train_y, eval_set=[(train_x, train_y), (x_val, y_val)],
    eval_metric= 'auc', verbose= 200, early_stopping_rounds= 200)
```

```
y_pred=clf.predict(X_test_final)
a=pd.DataFrame()
a['SK_ID_CURR']=X_test_final.index
a['TARGET']=y_pred[:,1]
a.to_csv('submission.csv')
```

Submission and Description	Private Score	Public Score	Use for Final Score
submission(1).csv 20 hours ago by Anupam Shandilya add submission details	0.78682	0.78978	☐

The **AUC** obtained after this model is **0.789.**

This model uses some 798 features compared to 67 features in the previous models. Such many features affect the interpretability and explain ability of the model.

```
Train samples: 307511, test samples: 48744
Bureau df shape: (305811, 116)
Previous applications df shape: (338857, 249)
Pos-cash balance df shape: (337252, 18)
Installments payments df shape: (339587, 26)
Credit card balance df shape: (103558, 141)
Starting LightGBM. Train shape: (307507, 798), test shape: (48744, 798)
```

Model Selection

Below we listed out the test performance of each model. In general, LGBM version 2 and stacked model has good performance. If the prediction accuracy is the top priority, Home Credit Group should use LGBM (version 2). On the other hand, there are 700+ features trained in LGBM (version 2) and we can't interpret the interaction of features among each other. If the interpretability is important, then the company should use LGBM (version 1) with features selected by our original feature reduction approach.

Model	Test AUC
Logistic	0.50
Random Forest	0.71
XGBoost	0.74
XGBoost+LGBM (stacked model)	0.76
LGBM version 1*	**0.75**
LGBM version 2	**0.789**

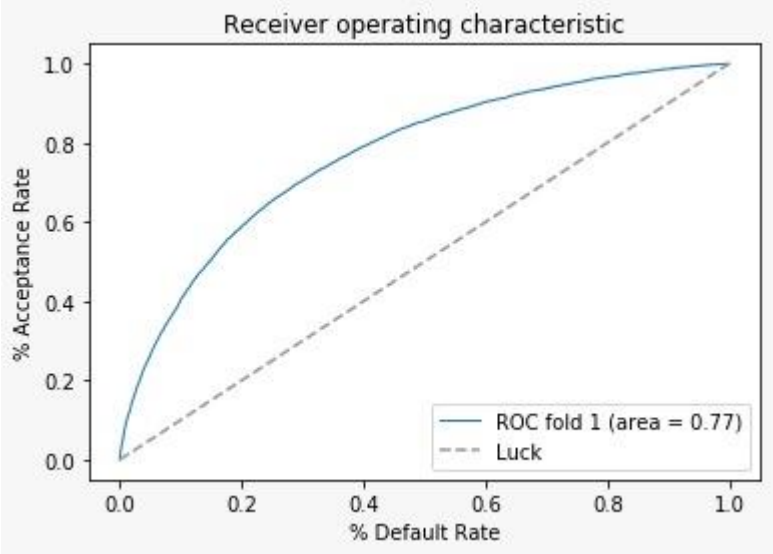
Business Impact of the model

Risk Return Trade Off Curve

The final model operating curve shows the movement of acceptance rates with the default rates. Depending on the business strategy Home Credit is pursuing, the bank can choose an operating region to work on,

For example, if the bank is aggressive on acquisitions and want to increase their portfolio, they will want to compromise on the rate of default of their overall book.

On the other hand, if the bank is conservative with their risk appetite, they will need to trade it off with relatively lower number of acquisitions.



Reduction in Bad Debt Loss

We have calculated the expected credit loss associated with each customer and quantified the loss savings the bank can make on each customer. The expected credit loss (ECL) is calculated as the following:

Expected Credit Loss = Probability of Default * Loan Amount for each customer

The probability of default is output by the model for each customer and the loan amount is an application level field.

Upon summing up the ECL for the entire portfolio book on the test data, we find there is a 9.3% decrease in the portfolio loss, thus saving 180.4 million dollars for the bank (calculation attached in the appendix).

ECL_Calculation.csv

NEXT STEPS:

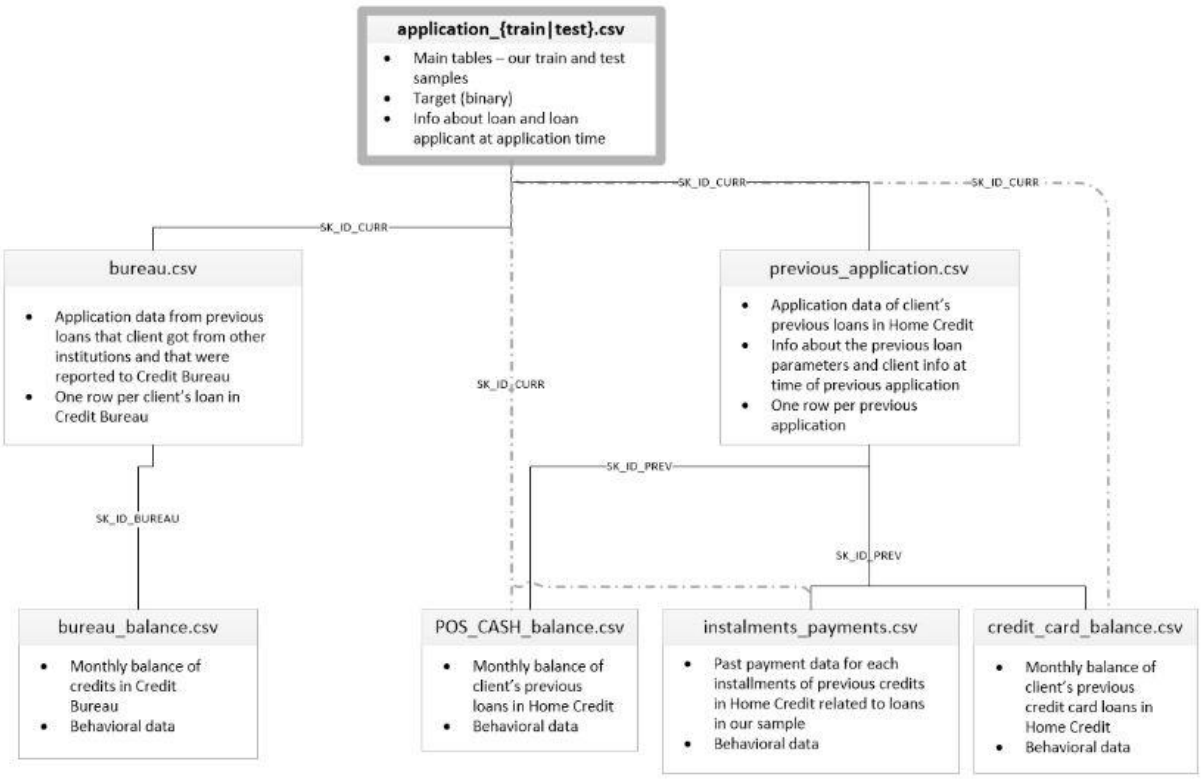
Since, we have a model that can better predict the defaulters in applications. We should probably go ahead and look at other factors such as interest rate, repayment duration etc. Tailored values for these features can be identified that would probably stop a person from defaulting. For Ex: A person may have defaulted just because his repayment period was less. So, if we could do an analysis for such metrics it would further improve their loan approval rate.

APPENDIX:

Code:

The code has been attached here as a Jupyter notebook. Kindly refer it.

Data Model:


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
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_CURR
SK_ID_PREV
SK_ID_BUREAU
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