HOME CREDIT DEFAULT RISK CHALLENGE

Project Report

Pranav Teja Palakurthi, 18AE3AI02

Ref: All necessary code files can be found at Google drive link

Data Description

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.

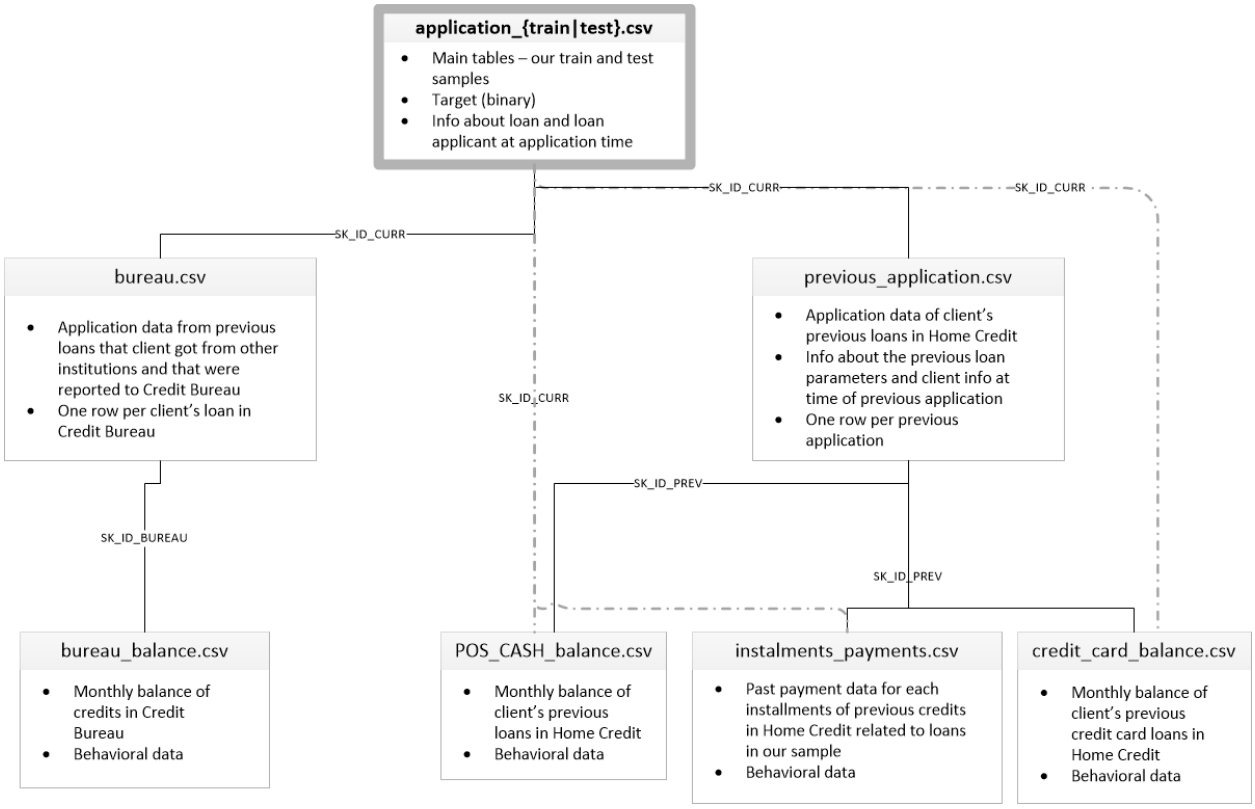
Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. 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.

Evaluation

Submissions are evaluated on area under the ROC curve between the predicted probability and the observed target.

Datasets



Process followed

Out of the many data tables provided here, each table is connected with another with a key id. First of all, we need to start analysis from the bottom tables.

For example, Bureau Balance table is connected to Bureau table with a key as SK_ID_BUREAU and also Bureau table is connected to Application Train/Test tables with a key as SK_ID_CURR.

We will start to analyze Bureau Balance first! Then, we will deduplicate the data according to the SK_ID_BUREAU variable and generate new variables to use on Bureau table. After that, the same processings should apply from Bureau table to Application Train/Test tables by using SK_ID_CURR.

When Bureau and Bureau Balance is done, we need to start analysis other bottom tables again to transfer information up!

Steps

Investigate intersections to key ids of tables!
All tables have different number of observations and variables. Moreover, when we merge two tables, some ids might not connect. That's why, sometimes we will see missing values naturally. The purpose of this step is to raise awareness for

intersections. Also, a classification problem can include unbalanced target variable. If we work on a classification problem, we must look at target variable.

Other Steps
Bureau Balance
Bureau
Pos Cash Balance
Credit Card Balance
Installments Payments
Previous Application
Application Train Test
EDA
Data Pre-processing
Singularization
Generate New Features
Merge all tables with Application Train/Test

Dimensions of data

```
train = pd.read_csv("../input/home-credit-default-risk/application_train.csv")
test = pd.read_csv("../input/home-credit-default-risk/application_test.csv")
bureau = pd.read_csv("../input/home-credit-default-risk/bureau.csv")
bureau_balance = pd.read_csv("../input/home-credit-default-risk/bureau_balance.csv")
pos = pd.read_csv('../input/home-credit-default-risk/POS_CASH_balance.csv')
cc = pd.read_csv('../input/home-credit-default-risk/credit_card_balance.csv')
ins = pd.read_csv('../input/home-credit-default-risk/installments_payments.csv')
prev = pd.read_csv('../input/home-credit-default-risk/previous_application.csv')

print("Dimension")
train.shape, test.shape, bureau.shape, bureau_balance.shape, pos.shape, cc.shape, ins.shape, prev.shape
```

```
Dimension
((307511, 122),
 (48744, 121),
 (1716428, 17),
 (27299925, 3),
 (10001358, 8),
 (3840312, 23),
 (13605401, 8),
 (1670214, 37))
```

Further, the dataset is also class imbalanced as

```
TARGET
    COUNT  RATIO
0  282686   0.92
1   24825   0.08
```

Here 1 refers to the customers who face payment difficulties/ likely to default.

Exploratory Data Analysis - for each table

Tables are connected to each other with SK_ID_CURR, SK_ID_BUREAU and SK_ID_PREV key ids.

```
Number of unique observations in the SK_ID_CURR variable
 TRAIN: 307511    TEST: 48744

Number of unique observations in the SK_ID_BUREAU variable
 BUREAU: 1716428  BUREAU BALANCE: 817395       INTERSECTION: 774354

Number of unique observations in the SK_ID_CURR variable
 TRAIN & BUREAU INTERSECTION: 263491     TEST & BUREAU INTERSECTION: 42320
```

1. Bureau Balance:

Monthly balances of previous credits in Credit Bureau.

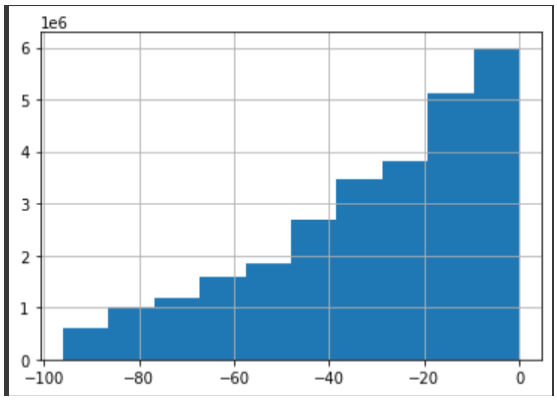
This table has one row for each month of history of every previous credit reported to Credit Bureau – i.e the table has (#loans in sample * # of relative previous credits * # of months where we have some history observable for the previous credits) rows.

Missing Value statistics and Descriptive statistics:

```
SK_ID_BUREAU      0
MONTHS_BALANCE    0
STATUS            0
dtype: int64
```

```
std       23.86
max        0.00
median   -25.00
mean     -30.74
min      -96.00
Name: MONTHS_BALANCE, dtype: float64
8.0  Max year
```

Histogram of Balance Monthly Payments (defaulted)
(Number of Defaulters vs Months defaulted)



2. Bureau table:

a. 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).
b. For every loan in our sample, there are as many rows as number of credits the client had in Credit Bureau before the application date.

EDA for Bureau reveals many missing variables:

Number of Variables include Missing Values: 24

	Feature	Num_Missing	Missing_Ratio	DataTypes
0	MONTHS_BALANCE_CLOSED_DIF	1299036	0.76	float64
1	MONTHS_BALANCE_FIRST_C	1299036	0.76	float64
2	AMT_ANNUITY	1226791	0.71	float32
3	AMT_CREDIT_MAX_OVERDUE	1124488	0.66	float32
4	STATUS_5_MEAN	942074	0.55	float64
5	STATUS_0_MEAN	942074	0.55	float64
6	STATUS_1_MEAN	942074	0.55	float64
7	STATUS_2_MEAN	942074	0.55	float64
8	STATUS_3_MEAN	942074	0.55	float64
9	STATUS_4_MEAN	942074	0.55	float64
10	STATUS_C_MEAN	942074	0.55	float64
11	MONTHS_BALANCE_MAX	942074	0.55	float64
12	STATUS_X_MEAN	942074	0.55	float64
13	STATUS_C0_MEAN_SUM	942074	0.55	float64
14	STATUS_12_MEAN_SUM	942074	0.55	float64
15	STATUS_345_MEAN_SUM	942074	0.55	float64

Process followed for missing values: Missing values are dropped
Now, Bureau database and bureau balance database are merged and their categories are analyzed

The results are as follows:

CREDIT_ACTIVE

	COUNT	RATIO
Closed	1079273	0.63
Active	630607	0.37
Sold	6527	0.00
Bad debt	21	0.00

CREDIT_CURRENCY

	COUNT	RATIO
currency 1	1715020	1.00
currency 2	1224	0.00
currency 3	174	0.00
currency 4	10	0.00

CREDIT_TYPE

	COUNT	RATIO
Consumer credit	1251615	0.73

Credit card	402195	0.23
Car loan	27690	0.02
Mortgage	18391	0.01
Microloan	12413	0.01
Loan for business development	1975	0.00
Another type of loan	1017	0.00
Unknown type of loan	555	0.00
Loan for working capital replenishment	469	0.00
Cash loan (non-earmarked)	56	0.00
Real estate loan	27	0.00
Loan for the purchase of equipment	19	0.00
Loan for purchase of shares (margin lending)	4	0.00
Interbank credit	1	0.00
Mobile operator loan	1	0.00

From this analysis we can make a few inferences:

A. CREDIT_CURRENCY variable is useless for modelling. Almost all of rows are currency 1 category.
B. CREDIT_ACTIVE variable might be useful. There are two rare categories in this column. We can combine these two categories so we assign a new category as Sold_BadDebt. Briefly, CREDIT_ACTIVE variable includes 3 categories as Active, Closed and Sold_BadDebt.
C. CREDIT_TYPE might be useful but there are some rare categories too. We should reduce number of category.

Similar steps in EDA were followed for the remaining tables: For reference, follow the google drive referenced in this document.
These are the remaining tables described here

3. 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 – i.e. the table has (#loans in sample * # of relative previous credit cards * # of months where we have some history observable for the previous credit card) rows.

4.Installments Payments

Repayment history for the previously disbursed credits in Home Credit related to the loans in our sample.
There is a) one row for every payment that was made plus b) one row each for missed payment.

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.

5.Previous Applications

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.

6.Application Train/Test

This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET).

Static data for all applications. One row represents one loan in our data sample.

Code to train classifier model based on the pre-processed data:

```
# 1. PACKAGES
# ----------------------------------------------------------------
# Base
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Model
from lightgbm import LGBMClassifier
# Configuration
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', None)
pd.options.display.float_format = '{:,.2f}'.format

"""# 2. DATA"""
df =
pd.read_feather("../input/homecredit-default-risk-step-by-step-1st-notebook/applications_traintest_feather")
pos =
pd.read_feather("../input/homecredit-default-risk-step-by-step-1st-notebook/poscashbalance_agg_feather")
bb =
pd.read_feather("../input/homecredit-default-risk-step-by-step-1st-notebook/bureau_bureaubalance_agg_feather")
```

```
cc =
pd.read_feather("../input/homecredit-default-risk-step-by-step-1st-noteboo
k/cc_feather")
ins =
pd.read_feather("../input/homecredit-default-risk-step-by-step-1st-noteboo
k/installments_payments_agg_feather")
prev =
pd.read_feather("../input/homecredit-default-risk-step-by-step-1st-noteboo
k/previous_applications_agg_feather")
print(df.shape, pos.shape, bb.shape, cc.shape, ins.shape, prev.shape)
for i in [pos, bb, cc, ins, prev]:
    df = pd.merge(df, i , how = "left", on = "SK_ID_CURR")

print(df.shape)
del pos, bb, ins, cc, prev

"""# 3. TRAIN-TEST SPLIT"""
# Train-Test Split
df.columns = list(map(lambda x: str(x).replace(" ", "_").replace("-",
"_").replace("_/_", "_").upper(), df.columns))
import re
df = df.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
train = df[df.TARGET.isnull() == False]
test = df[df.TARGET.isnull()]
x_train = train.drop(["TARGET", "SK_ID_CURR"], axis = 1)
x_test = test.drop(["TARGET", "SK_ID_CURR"], axis = 1)
y_train = train.TARGET

"""# 4. MODEL"""
# LightGBM parameters found by Bayesian optimization
clf = 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, )
clf.fit(x_train, y_train, eval_set=[(x_train, y_train)],
        eval_metric='auc', verbose=200)

"""# 5. Submission"""
submission = pd.DataFrame({
    "SK_ID_CURR": test.SK_ID_CURR,
    "TARGET": clf.predict_proba(x_test)[:,1]
})
submission.to_csv("submission.csv", index = None)
```

Results:

Training AUC (Area under precision-recall Curve) and final Classifier Model parameters

```
[LightGBM] [Warning] num_threads is set with nthread=4, will be overridden
by n_jobs=-1. Current value: num_threads=-1
[200] training's auc: 0.797198     training's binary_logloss: 0.234742
[400] training's auc: 0.819484     training's binary_logloss: 0.225443
[600] training's auc: 0.83453      training's binary_logloss: 0.219295
[800] training's auc: 0.846823     training's binary_logloss: 0.214246
[1000]      training's auc: 0.857045     training's binary_logloss: 0.2099
[2000]      training's auc: 0.897027     training's binary_logloss:
0.191207
[3000]      training's auc: 0.924346     training's binary_logloss:
0.175962
[4000]      training's auc: 0.944803     training's binary_logloss: 0.16237
[5000]      training's auc: 0.960103     training's binary_logloss:
0.149976
[6000]      training's auc: 0.97113      training's binary_logloss:
0.139124
[7000]      training's auc: 0.979043     training's binary_logloss:
0.129345
[8000]      training's auc: 0.984991     training's binary_logloss:
0.120019
[9000]      training's auc: 0.989492     training's binary_logloss:
0.111315
[10000]     training's auc: 0.99246      training's binary_logloss:
0.103873
```

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

Test Results:

Submissions generated in the last step of the code when submitted to kaggle result in a **final test AUC score of 0.7799**