# Highlights on Feature Engineering

Selected engineered features that not in other public kernels

## Loan Coverage:

all\_application\_df['[FE\_APP]CREDIT\_COVERAGE'] =

all\_application\_df['AMT\_CREDIT'] / all\_application\_df['AMT\_GOODS\_PRICE']

## Application Loan Duration:

all\_application\_df['[FE\_APP]LOAN\_DURATION'] =

all\_application\_df['AMT\_CREDIT'] / all\_application\_df['AMT\_ANNUITY']

## Applicant Age at application time:

all\_application\_df['[FE\_APP]AGE'] =

(all\_application\_df['DAYS\_BIRTH'] / 365).astype(int)

## Applicant Age at the end of the loan:

## (The same for previous loan and bureau loan as well)

all\_application\_df['[FE\_APP]AGE\_FINISH'] = (all\_application\_df['[FE\_APP]LOAN\_DURATION']/12 + all\_application\_df['DAYS\_BIRTH'].abs())/365

## Bureau Debt Projection:

bureau\_df['[FE\_B]AVG\_DEBT\_SINCE\_START'] =

bureau\_df['AMT\_CREDIT\_SUM\_DEBT'] / bureau\_df['DAYS\_CREDIT']

bureau\_df['[FE\_B]AVG\_OVERDUE\_SINCE\_START'] =

bureau\_df['AMT\_CREDIT\_MAX\_OVERDUE'] / bureau\_df['DAYS\_CREDIT']

## Using the Bureau overdue history to project application overdue:

bureau\_df['[FE\_B]AVG\_OVERDUE\_SINCE\_START'] =

bureau\_df['AMT\_CREDIT\_MAX\_OVERDUE'] / bureau\_df['DAYS\_CREDIT']

bureau\_df['[FE\_B]APP\_PROJECTED\_DEBT'] =

bureau\_df['[FE\_B]AVG\_DEBT\_SINCE\_START'] \* bureau\_df['[FE\_APP]LOAN\_DURATION']

## Using the installment overdue history to project application overdue:

installments\_df['FE\_DIFF\_DAY'] =   
installments\_df ['DAYS\_ENTRY\_PAYMENT'] - installments\_df['DAYS\_INSTALMENT']

installments\_df['FE\_APP\_PROJECTED\_DIFF\_DAY'] =

installments\_df ['FE\_DIFF\_DAY'] \* installments\_df ['[FE\_APP]LOAN\_DURATION']

## Comparing the details between application loan and the previous loans:

previous\_application\_df['FE\_4'] =

previous\_application\_df['APP\_AMT\_CREDIT'] / (1+previous\_application\_df['AMT\_CREDIT'])

## Credit card credit used ratio:

credit\_card\_balance\_df['FE\_USED\_RATIO'] = credit\_card\_balance\_df['FE\_TOTAL\_AMT\_DRAW'] / credit\_card\_balance\_df['AMT\_CREDIT\_LIMIT\_ACTUAL']

## Credit card bill paid ratio:

credit\_card\_balance\_df['FE\_PAID\_RATIO']= credit\_card\_balance\_df['AMT\_PAYMENT\_TOTAL\_CURRENT']\

/ np.maximum(credit\_card\_balance\_df['AMT\_TOTAL\_RECEIVABLE'], credit\_card\_balance\_df['AMT\_INST\_MIN\_REGULARITY'])

# Highlights on records filtering and division

Basically, a nested for loop.

## For tables other than application.csv, they are filtered into sub-tables:

active\_bureau\_df = bureau\_df[(bureau\_df['CREDIT\_ACTIVE']=='Active')]

closed\_bureau\_df = bureau\_df[(bureau\_df['CREDIT\_ACTIVE']=='Closed')]

active\_bureau\_credit\_card\_df = bureau\_df[(bureau\_df['CREDIT\_TYPE'] == 'Credit card') &(bureau\_df['CREDIT\_ACTIVE']=='Active')]

closed\_bureau\_credit\_card\_df = bureau\_df[(bureau\_df['CREDIT\_TYPE'] == 'Credit card') &(bureau\_df['CREDIT\_ACTIVE']=='Closed')]

sold\_bureau\_df = bureau\_df[(bureau\_df['CREDIT\_ACTIVE']=='Sold')]

## For each sub tables, the they are divided by “time”:

2 different “time” definitions are used:

1. n months before the application, (3, 6, 18, 30, 42, 54, 66 months before)
2. The latest k records, the earliest K records, k is ratio (0.1, 0.2, 0.3, 0.4)

Then the tables are grouped by [‘SK\_ID\_CURR’] and [‘SK\_ID\_CURR’, ‘SK\_ID\_PREV’] and aggregated mainly by mean and max, sometimes sum, min and seldom median. Trend aggregation is also used. It is inspired by the Open Solution.

# Result

The final dataset contains 5000+ features, they are then cutted in 800 features based on 10-fold lightgbm gain importance. CV: 0.80368, Public lb: 0.80386, Private lb: 0.80048

## Acknowledgement

The model is then tuned by @a31314431 and @shivraj by Bayes Optimization and @olivier selection script. CV: 0.80500, Public lb: 0.80533, private lb: 0.80207

Also had an improvement with imputing features from @lingchensun.

@fatihozturk provided me with good starting parameters for lightgbm.