University of Toronto

Mathematical Finance

Kaggle Competition

Home Credit Default Risk

Author

Keren Lai

Zhonghen Shen

Hongyi Wu

Tianyu(Shin) Ren

Contents

1. INTRODUCTION

Home Credit is a service dedicated to provided lines of loans to the unbanked population based on customer's previous records. This project is to help Home Credit to make predictions about clients capability of repayment using various statistical and machine learning methods.

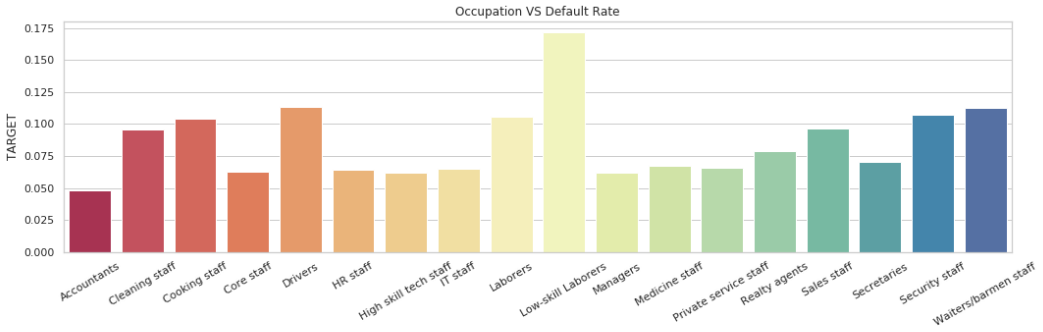
1.1. Data

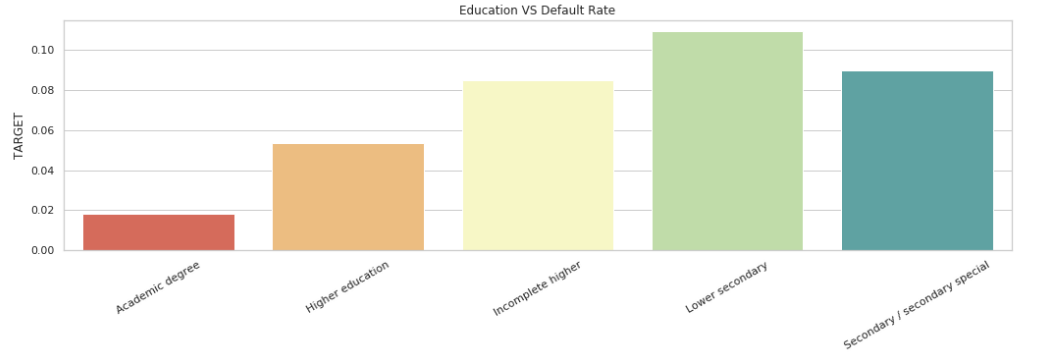
The data we use is provided by Home Credit. There are 7 different sources of data:

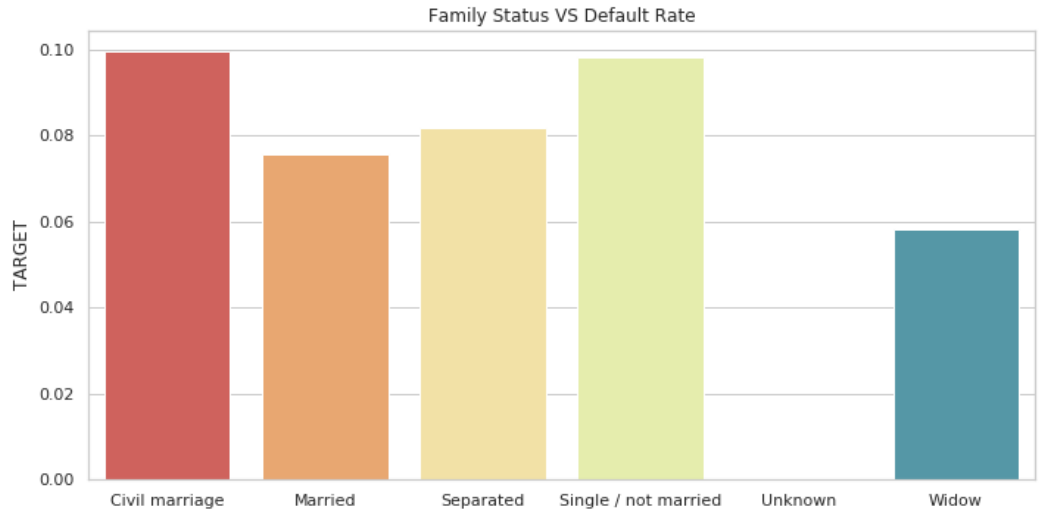
- application_train/application_test: This is the main training and testing data we use to get information about each customers' loan application at Home Credit. We randomly choose 70% data from application_train to build our model, and use the rest 30% to validate the model; we produce Target for application_test indicating 0: the loan was likely to be repaid or 1: the load was likely to default;
- bureau: The data records clients's previous credits from other financial institutions;
- bureau_balance: This is monthly data records clients' previous credits from other financial institutions;
- previous_application: This is data records clients' previous applications for loans at Home Credit;
- POS_CASH_balance: This is monthly data about previous point of sale or cash loans clients have had at Home Credit;
- credit_card_balance: This is monthly data about previous credit cards clients have had at Home Credit;
- installments_payment: This is payment history of clients at Home Credit.

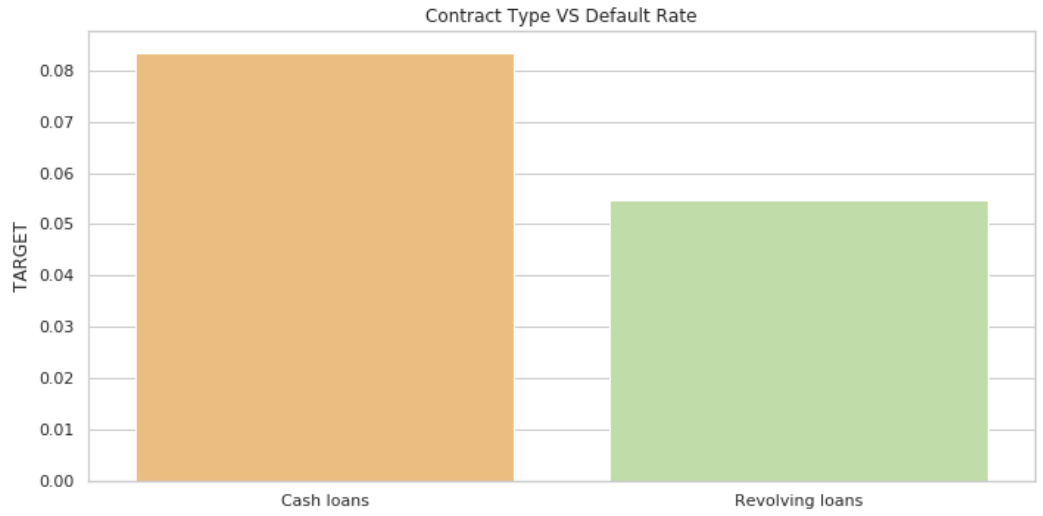
1.2. Exploratory Data Analysis

Before feature engineering, we calculated some statistics and make graphs to see trends, anomalies, patterns, or relationships with the data. To decide which features to use, we make figures showing the relationships between several variables and default rate.


Occupation VS Default Rate
TARGET
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Accountants
Cleaning staff
Cooking staff
Core staff
Drivers
HR staff
High skill tech staff
IT staff
Laborers
Low-skill Laborers
Managers
Medicine staff
Private service staff
Realty agents
Sales staff
Secretaries
Security staff
Waiters/barmen staff


Education VS Default Rate
TARGET
0.10
0.08
0.06
0.04
0.02
0.00
Academic degree
Higher education
Incomplete higher
Lower secondary
Secondary / secondary special


Family Status VS Default Rate
TARGET
0.10
0.08
0.06
0.04
0.02
0.00
Civil marriage
Married
Separated
Single / not married
Unknown
Widow


Contract Type VS Default Rate
TARGET
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Cash loans
Revolving loans

2. DATA TRANSFORMATION

2.1. Final Features Tables

After features engineering, we use 233 features as final features:

FEATURES	NAME_CONTRACT_TYPE	CODE_GENDER
FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN
AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY
AMT_GOODS_PRICE	NAME_TYPE_SUITE	NAME_INCOME_TYPE
NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE
REGION_POPULATION_RELATIVE	DAYS_BIRTH	DAYS_EMPLOYED
DAYS_REGISTRATION	DAYS_ID_PUBLISH	OWN_CAR_AGE
FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE
FLAG_CONT_MOBILE	FLAG_PHONE	FLAG_EMAIL
OCCUPATION_TYPE	CNT_FAM_MEMBERS	REGION_RATING_CLIENT
REGION_RATING_CLIENT_W_CITY	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCESS_START
REG_REGION_NOT_LIVE_REGION	REG_REGION_NOT_WORK_REGION	LIVE_REGION_NOT_WORK_REGION
REG_CITY_NOT_LIVE_CITY	REG_CITY_NOT_WORK_CITY	LIVE_CITY_NOT_WORK_CITY
! ORGANIZATION_TYPE	EXT_SOURCE_1	EXT_SOURCE_2
EXT_SOURCE_3	OBS_30_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE	DEF_60_CNT_SOCIAL_CIRCLE	DAYS_LAST_PHONE_CHANGE
FLAG_DOCUMENT_2	FLAG_DOCUMENT_3	FLAG_DOCUMENT_4
FLAG_DOCUMENT_5	FLAG_DOCUMENT_6	FLAG_DOCUMENT_7
FLAG_DOCUMENT_8	FLAG_DOCUMENT_9	FLAG_DOCUMENT_10
FLAG_DOCUMENT_11	FLAG_DOCUMENT_12	FLAG_DOCUMENT_13
FLAG_DOCUMENT_14	FLAG_DOCUMENT_15	FLAG_DOCUMENT_16
FLAG_DOCUMENT_17	FLAG_DOCUMENT_18	FLAG_DOCUMENT_19
FLAG_DOCUMENT_20	FLAG_DOCUMENT_21	AMT_REQ_CREDIT_BUREAU_HOUR
AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_WEEK	AMT_REQ_CREDIT_BUREAU_MON
AMT_REQ_CREDIT_BUREAU_QRT	AMT_REQ_CREDIT_BUREAU_YEAR	Still_Owed_max
Still_Owed_min	Still_Owed_mean	Overdue_Days_std
Overdue_Days_max	Overdue_Days_min	Overdue_Days_mean

Still_Owed_std	NUM_INSTALMENT_NUMBER_std	Overdue_Days_sum
NUM_INSTALMENT_NUMBER_mean	NUM_INSTALMENT_VERSION_mean	NUM_INSTALMENT_VERSION_min
NUM_INSTALMENT_VERSION_max	NUM_INSTALMENT_VERSION_std	NUM_INSTALMENT_VERSION_sum
NUM_INSTALMENT_NUMBER_sum	NUM_INSTALMENT_NUMBER_min	NUM_INSTALMENT_NUMBER_max
Still_Owed_sum	Excessive_Repay_std	Excessive_Repay_max
Excessive_Repay_min	Excessive_Repay_mean	AMT_DRAWINGS_OTHER_CURRENT_mean
AMT_DRAWINGS_POS_CURRENT_std	AMT_DRAWINGS_ATM_CURRENT_max	AMT_DRAWINGS_ATM_CURRENT_min
AMT_DRAWINGS_ATM_CURRENT_mean	AMT_DRAWINGS_OTHER_CURRENT_min	AMT_DRAWINGS_OTHER_CURRENT_max
AMT_DRAWINGS_OTHER_CURRENT_std	AMT_DRAWINGS_POS_CURRENT_mean	AMT_DRAWINGS_POS_CURRENT_min
AMT_DRAWINGS_POS_CURRENT_max	AMT_DRAWINGS_ATM_CURRENT_std	AMT_CREDIT_LIMIT_ACTUAL_min
AMT_DRAWINGS_ATM_CURRENT_sum	AMT_BALANCE_sum	AMT_BALANCE_std
AMT_BALANCE_max	AMT_CREDIT_LIMIT_ACTUAL_max	AMT_BALANCE_min
AMT_BALANCE_mean	MONTHS_BALANCE_sum	AMT_CREDIT_LIMIT_ACTUAL_std
AMT_CREDIT_LIMIT_ACTUAL_sum	MONTHS_BALANCE_std	MONTHS_BALANCE_max
MONTHS_BALANCE_min	AMT_CREDIT_LIMIT_ACTUAL_mean	AMT_DRAWINGS_CURRENT_mean
AMT_RECIVABLE_sum	AMT_RECIVABLE_std	AMT_RECIVABLE_max
AMT_RECIVABLE_min	AMT_RECIVABLE_mean	AMT_DRAWINGS_CURRENT_min
AMT_DRAWINGS_POS_CURRENT_sum	AMT_DRAWINGS_OTHER_CURRENT_sum	Excessive_Repay_sum
AMT_DRAWINGS_CURRENT_sum	AMT_DRAWINGS_CURRENT_std	AMT_DRAWINGS_CURRENT_max
MONTHS_BALANCE_mean	AMT_ANNUITY_mean	AMT_ANNUITY_min
AMT_ANNUITY_max	AMT_ANNUITY_std	AMT_ANNUITY_sum
AMT_APPLICATION_mean	AMT_APPLICATION_min	AMT_APPLICATION_max
AMT_APPLICATION_std	AMT_APPLICATION_sum	AMT_CREDIT_mean
AMT_CREDIT_min	AMT_CREDIT_max	AMT_CREDIT_std
AMT_CREDIT_sum	RATE_DOWN_PAYMENT_mean	RATE_DOWN_PAYMENT_min
RATE_DOWN_PAYMENT_max	RATE_DOWN_PAYMENT_std	RATE_DOWN_PAYMENT_sum
AMT_DOWN_PAYMENT_mean	AMT_DOWN_PAYMENT_min	AMT_DOWN_PAYMENT_max
AMT_DOWN_PAYMENT_std	AMT_DOWN_PAYMENT_sum	AMT_GOODS_PRICE_mean
AMT_GOODS_PRICE_min	AMT_GOODS_PRICE_max	AMT_GOODS_PRICE_std
AMT_GOODS_PRICE_sum	DAYS_DECISION_mean	DAYS_DECISION_min
DAYS_DECISION_max	DAYS_DECISION_std	DAYS_DECISION_sum
CNT_PAYMENT_mean	CNT_PAYMENT_min	CNT_PAYMENT_max
CNT_PAYMENT_std	CNT_PAYMENT_sum	b_feature_1
b_feature_2	b_feature_3	b_feature_4
b_feature_5	b_feature_6	b_feature_7
b_feature_auto_DAYS_CREDIT_ENDDATE_mean	b_feature_auto_DAYS_CREDIT_ENDDATE_min	b_feature_auto_DAYS_CREDIT_ENDDATE_max
b_feature_auto_DAYS_CREDIT_ENDDATE_std	b_feature_auto_DAYS_CREDIT_ENDDATE_sum	b_feature_auto_AMT_CREDIT_SUM_mean
b_feature_auto_AMT_CREDIT_SUM_min	b_feature_auto_AMT_CREDIT_SUM_max	b_feature_auto_AMT_CREDIT_SUM_std
b_feature_auto_AMT_CREDIT_SUM_sum	b_feature_auto_AMT_CREDIT_SUM_DEBT_mean	b_feature_auto_AMT_CREDIT_SUM_DEBT_min
b_feature_auto_AMT_CREDIT_SUM_DEBT_max	b_feature_auto_AMT_CREDIT_SUM_DEBT_std	b_feature_auto_AMT_CREDIT_SUM_DEBT_sum
b_feature_auto_AMT_CREDIT_MAX_OVERDUE_mean	b_feature_auto_AMT_CREDIT_MAX_OVERDUE_min	b_feature_auto_AMT_CREDIT_MAX_OVERDUE_max
b_feature_auto_AMT_CREDIT_MAX_OVERDUE_std	b_feature_auto_AMT_CREDIT_MAX_OVERDUE_sum	MONTHS_BALANCE_mean_pos

MONTHS_BALANCE_min_pos	MONTHS_BALANCE_max_pos	MONTHS_BALANCE_std_pos
MONTHS_BALANCE_sum_pos	CNT_INSTALMENT_mean_pos	CNT_INSTALMENT_min_pos
CNT_INSTALMENT_max_pos	CNT_INSTALMENT_std_pos	CNT_INSTALMENT_sum_pos
CNT_INSTALMENT_FUTURE_mean_pos	CNT_INSTALMENT_FUTURE_min_pos	CNT_INSTALMENT_FUTURE_max_pos
CNT_INSTALMENT_FUTURE_std_pos	CNT_INSTALMENT_FUTURE_sum_pos	SK_DPD_mean_pos
SK_DPD_min_pos	SK_DPD_max_pos	SK_DPD_std_pos
SK_DPD_sum_pos	SK_DPD_DEF_mean_pos	SK_DPD_DEF_min_pos
SK_DPD_DEF_max_pos	SK_DPD_DEF_std_pos	SK_DPD_DEF_sum_pos
amt_diff	num_previous_loan	insurance_ratio

2.2. Feature Engineering

2.2.1. Manual Generated Features

Based on intuitive understanding, we have created 13 new features as following. Please see Appendix for the code details.

Features Name	Data Source	Methodology to Generate
amt_diff	previous_application	AMT_CREDIT - AMT_APPLICATION
num_previous_loan	previous_application	groupby SK_ID_CURR and count SK_ID_PREV
insurance_ratio	previous_application	percentage that previous loan at Home Credit was insured
Overdue_Days	installments_payments	DAYS_INSTALMENT - DAYS_ENTRY_PAYMENT
Still_Owed	installments_payments	AMT_INSTALMENT - AMT_PAYMENT
Excessive_Repay	credit_card_balance	AMT_PAYMENT_CURRENT - AMT_INST_MIN_REGULARITY
b_feature_1	bureau_data	how many loans will expire in the future
b_feature_2	bureau_data	number of loan this ID has (currently and in the past)
b_feature_3	bureau_data	the ratio of debt to the sum of debt and credit
b_feature_4	bureau_data	whether this ID has prolonged his loan before
b_feature_5	bureau_data	the furthest due date of this ID's loan in the future
b_feature_6	bureau_data	the ratio of overdue to debt
b_feature_7	bureau_data & application_data	the ratio of income to debt

- amt_diff is the difference between ask and given of previous loan at Home Credit.
- num_previous_loan is the total number of previous loan at Home Credit.
- insurance_ratio is the percentage that previous loan at Home Credit was insured.
- Overdue_Days is the delay of repaying installment.
- Still_Owed is the amount still owed after repayment
- Excessive_Repay is the excessive repayment

2.2.2. Features Selection

First, features with too many missing values should be dropped. For the logistic regression, we need to avoid multicollinearity problems and features with high correlation are dropped. Next, we drop columns of building/construction information which add many noises to our model. One method of this is computing feature importance. More details can be found in section 3.2. Model Development (Logistic Regression).

2.3. Documentation of QA

Final Features contains both automated generated features and manual generated features. For the automated generated features, we select useful numerical columns based on our domain knowledge on all datasets except for the 'application_train.csv'. Manual generated features and their methodologies have been illustrated above. Please see Appendix for more details.

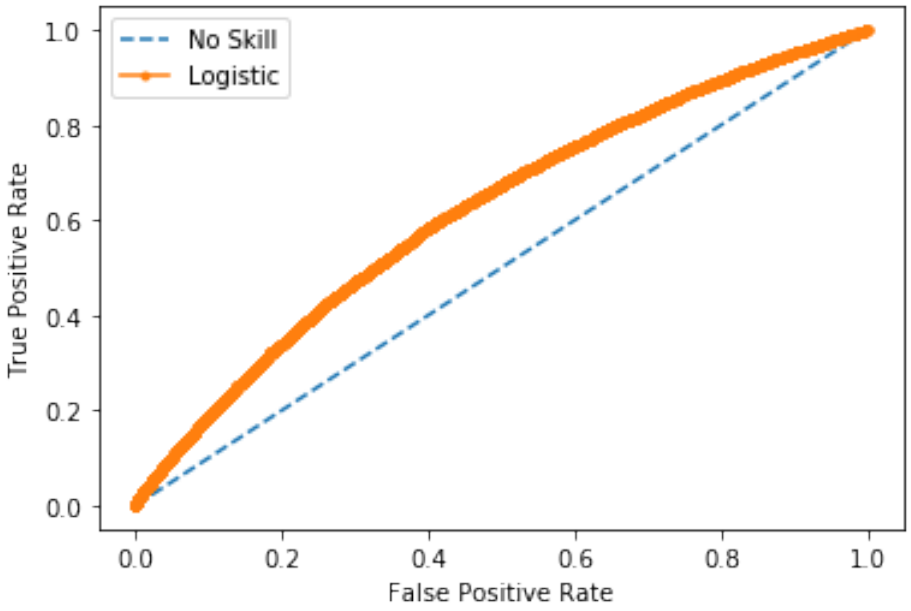
3. MODELS

3.1. Model Measures & Rationale

We use AUC as our model measure because for a binary classification problem, AUC is pretty good measure since it is easy to be interpreted and demonstrated. Also, submissions of this Kaggle competition are evaluated on AUC.

3.2. Model Development (Logistic Regression)

Logistic Regression is used as a benchmark since it is easy to train and interpret. However, we need to fill in NAs, which increases bias drastically to our model. We use mode to fill in categorical variables and median to fill in numerical variables. Before this, we transform binary variables via label encoding while multi-class variables via one-hot encoding. Also, features with more than 0.75 missing values are dropped. The following ROC curve is trained by logistic regression using this raw dataset.



LR Without Feature Engineering, AUC=0.620

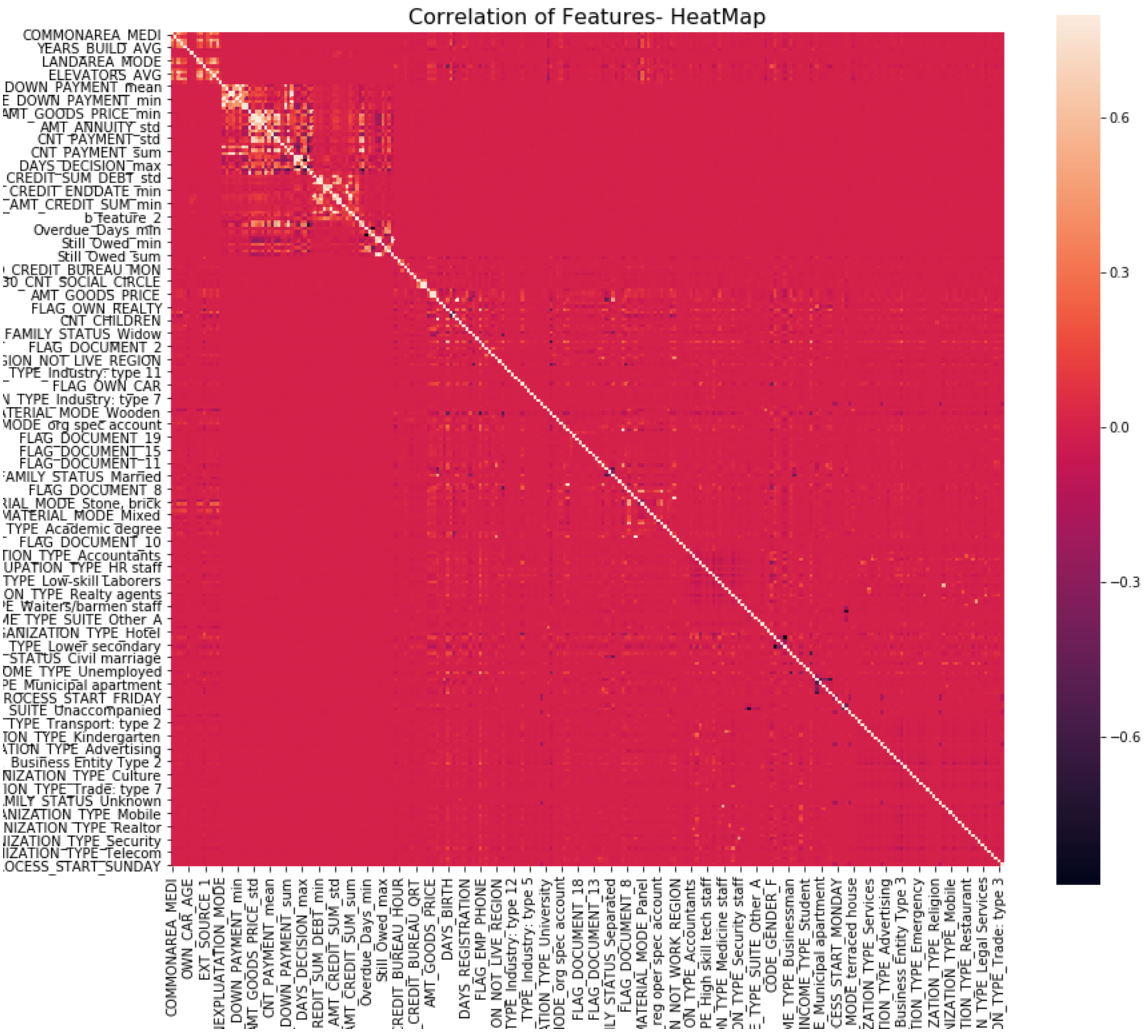
As we can see, the first AUC is not quite satisfying. Our next step is to perform additional data engineering and normalization to increase AUC.

To improve the performance of LR, further data engineering is conducted. It is possible that some features in provided datasets have extreme values, which might distort the information of the data. Therefore, we conduct extreme value detecting and transfer those values into missing values. Following table shows some examples of this processing:

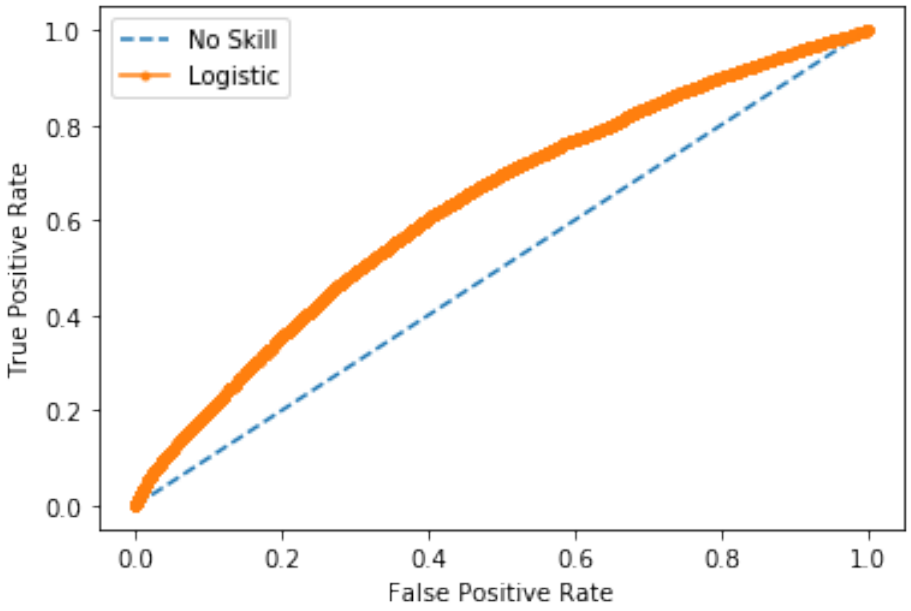
feature name	**value range**	**processing method**
OBS_30_CNT_SOCIAL_CIRCLE	(0, 354)	Turning values more than 100 into missing value
DAYS_CREDIT_ENDDATE	(-42020, 31199)	Turning values less than -2000 into missing value
AMT_CREDIT_SUM	(0, 585,000,000)	Turning values more than 10000000 into missing value

Besides, collinearity is considered. We first calculate the correlation matrix of our feature dataframe and then select the features with high correlation(≥ 0.9). Only one feature in the group with high correlation is kept.

The Heat Map of correlations are displayed below:

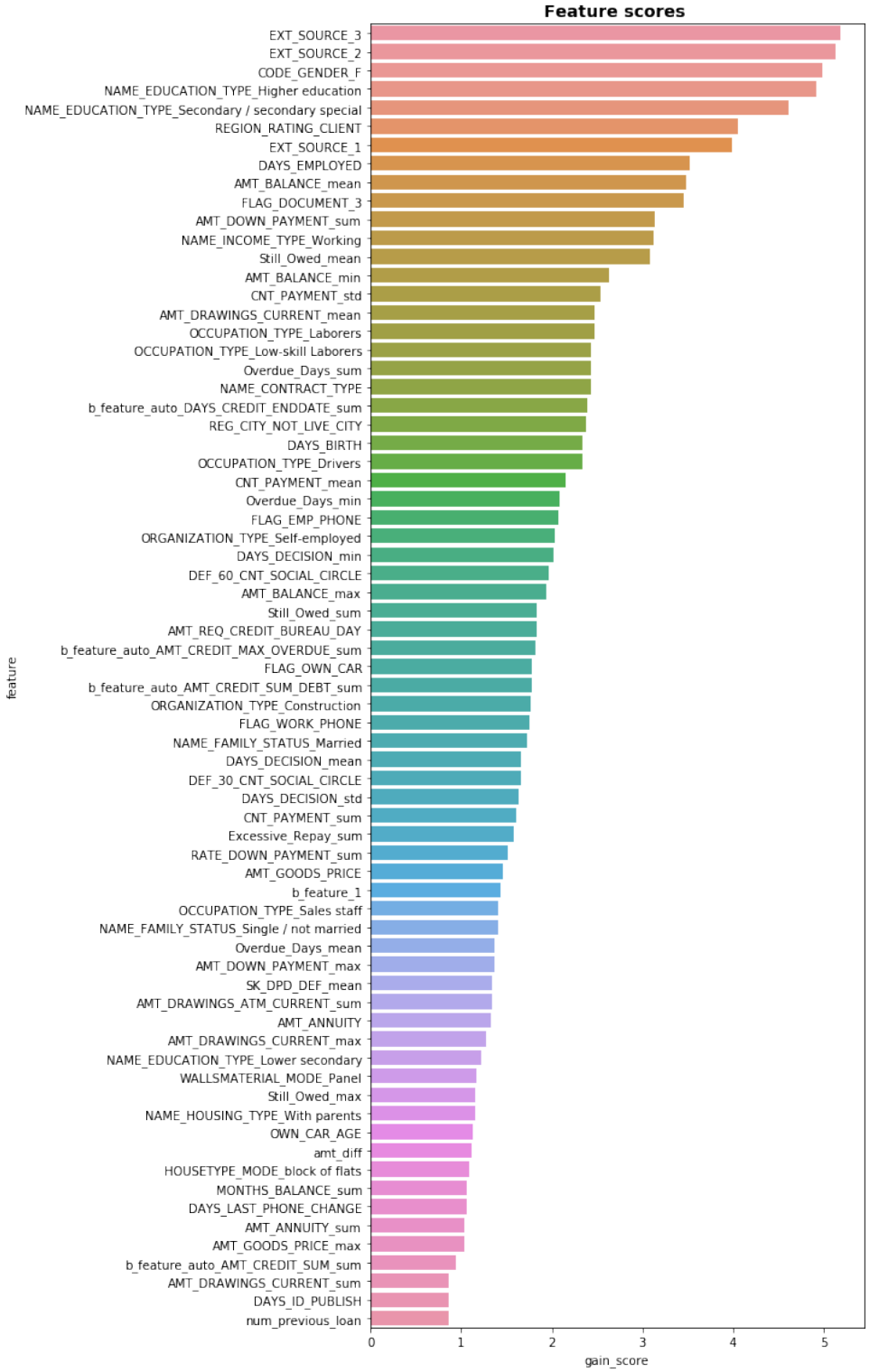


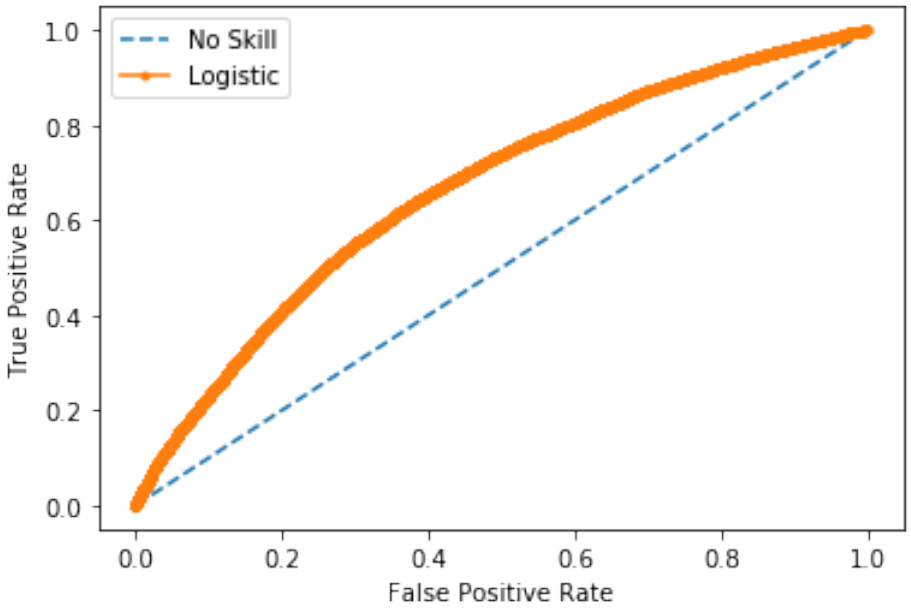
From the ROC diagram below, it is clear that the extreme value and correlation processing has improved the LR performance.



LR after further data processing, AUC=0.632

Finally, feature importance is utilized to select features. This is because some features in the former dataframe may have no contribution to the model performance. LightGBM is used to calculate the feature importance:


Feature scores
EXT_SOURCE_3
EXT_SOURCE_2
CODE_GENDER_F
NAME_EDUCATION_TYPE_Higher education
NAME_EDUCATION_TYPE_Secondary / secondary special
REGION_RATING_CLIENT
EXT_SOURCE_1
DAYS_EMPLOYED
AMT_BALANCE_mean
FLAG_DOCUMENT_3
AMT_DOWN_PAYMENT_sum
NAME_INCOME_TYPE_Working
Still_Owed_mean
AMT_BALANCE_min
CNT_PAYMENT_std
AMT_DRAWINGS_CURRENT_mean
OCCUPATION_TYPE_Laborers
OCCUPATION_TYPE_Low-skill Laborers
Overdue_Days_sum
NAME_CONTRACT_TYPE
b_feature_auto_DAYS_CREDIT_ENDDATE_sum
REG_CITY_NOT_LIVE_CITY
DAYS_BIRTH
OCCUPATION_TYPE_Drivers
CNT_PAYMENT_mean
Overdue_Days_min
FLAG_EMP_PHONE
ORGANIZATION_TYPE_Self-employed
DAYS_DECISION_min
DEF_60_CNT_SOCIAL_CIRCLE
AMT_BALANCE_max
Still_Owed_sum
AMT_REQ_CREDIT_BUREAU_DAY
b_feature_auto_AMT_CREDIT_MAX_OVERDUE_sum
FLAG_OWN_CAR
b_feature_auto_AMT_CREDIT_SUM_DEBT_sum
ORGANIZATION_TYPE_Construction
FLAG_WORK_PHONE
NAME_FAMILY_STATUS_Married
DAYS_DECISION_mean
DEF_30_CNT_SOCIAL_CIRCLE
DAYS_DECISION_std
CNT_PAYMENT_sum
Excessive_Repay_sum
RATE_DOWN_PAYMENT_sum
AMT_GOODS_PRICE
b_feature_1
OCCUPATION_TYPE_Sales staff
NAME_FAMILY_STATUS_Single / not married
Overdue_Days_mean
AMT_DOWN_PAYMENT_max
SK_DPD_DEF_mean
AMT_DRAWINGS_ATM_CURRENT_sum
AMT_ANNUITY
AMT_DRAWINGS_CURRENT_max
NAME_EDUCATION_TYPE_Lower secondary
WALLSMATERIAL_MODE_Panel
Still_Owed_max
NAME_HOUSING_TYPE_With parents
OWN_CAR_AGE
amt_diff
HOUSETYPE_MODE_block of flats
MONTHS_BALANCE_sum
DAYS_LAST_PHONE_CHANGE
AMT_ANNUITY_sum
AMT_GOODS_PRICE_max
b_feature_auto_AMT_CREDIT_SUM_sum
AMT_DRAWINGS_CURRENT_sum
DAYS_ID_PUBLISH
num_previous_loan
feature
0
1
2
3
4
5
gain_score



LR Without Feature Engineering, AUC=0.667

From the ROC diagram above we can know that after feature selection using feature importance, the AUC of logistic regression is further increased to 0.667.

3.3. Model Development (LightGBM)

In terms of the model selection, we choose LightGBM as our classifier. Currently the two most prevailing classification algorithms in Kaggle and other Data Science competitions are XGBoost and LightGBM. The reason that we choose LightGBM is it's 2 - 10 times faster than XGBoost.

3.4. Model Development (LightGBM)

3.5. Model Tuning (LightGBM)

Different hyper-parameter sets have quite large impact on the final result. We first initialized a parameters set, then applied to determine the best set of parameters.

(i) **LightGBM Parameters Introduction**

In LightGBM model, certain parameters plays very important roles and they are the targets of

hyper-parameter tuning. The following is a table demonstrating three important hyper-parameters and their functions.

hyper-parameters	**function**
num_leave	control the complexity of the tree model
min_data_in_leaf	prevent over-fitting in a leaf-wise tree
max_depth	similar to num_leave

Meanwhile, there are some universal hyper-parameters which exist in most machine learning models that worth of being tuned, such as learning-rate, regularization term, etc.

(ii) **Parameter Set**

We tuned our hyper-parameters from the following set:

(iii) **RandomizedSearchCV**

We use RandomizedSearchCV to sample a given number of candidates from a parameter set. Compared with grid search, randomized search is slightly worse in terms of performance, but due to the large amount of our dataset, we sacrificed a little accuracy for lowering the running time. The best parameter set we got is the following (for other parameters, we use the default values):

hyper-parameters	**value**
num_leave	30
reg_alpha	0.01
learning_rate	0.05

3.6. Model Prediction (LightGBM)

We split the whole dataset into training dataset and test dataset by the rule of ”80-20”. After training our model, we evaluated it on the test dataset. The confusion matrix and AUC score we got is the following:

	negative	positive
true	42241	3269
false	1606	14387
AUC	0.78059	

Appendices

```
!pip install kaggle
!mkdir .kaggle
import json
token = {"username":"xyz","key":"xxxxx"}
with open('/content/.kaggle/kaggle.json', 'w') as file:
    json.dump(token, file)
!cp /content/.kaggle/kaggle.json ~/.kaggle/kaggle.json
!kaggle config set -n path -v{/content}
!cp /content/.kaggle/kaggle.json ~/.kaggle/kaggle.json
!kaggle config set -n path -v{/content}
!chmod 600 /root/.kaggle/kaggle.json
!mkdir Kaggle_data
!kaggle competitions download -c home-credit-default-risk -p /content/Kaggle_data
!cd Kaggle_data; unzip \*.zip
!cd Kaggle_data; rm *.zip
!rm -r sample_data/

# code of lightGBM
# data preprocessing
import pandas as pd
import lightgbm as lgb
from sklearn.model_selection import GridSearchCV, train_test_split

train_data = pd.read_csv('Kaggle_data/application_train.csv')
train_data.index = train_data['SK_ID_CURR']
del train_data['SK_ID_CURR']

# Label encoding of binary categorical variables
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
le_count = 0

for col in train_data.columns:
  if train_data[col].dtypes == 'object':
    le.fit(train_data[col].astype(str))
    train_data[col] = le.transform(train_data[col].astype(str))
    le_count += 1

print('%d columns were label encoded.' % le_count)

# one-hot encoding of muti-categorical variables
# train_data = pd.get_dummies(train_data)
print('Training Features shape: ', train_data.shape)

# integrate multiple dataframes by left join
final_app_df = train_data.join(instal_pay_df, sort=False)
final_app_df = final_app_df.join(credit_card_df, sort=False)
final_app_df = final_app_df.join(pre_app_df, sort=False)
final_app_df = final_app_df.join(bureau_df, sort=False)
final_app_df = final_app_df.join(pos_cash_df, sort=False)
final_app_df = final_app_df.join(pre_mannual_features_df, sort=False)

# specify categorical features
f = open('categorical_features.txt', 'r')
x = f.readlines()
f.close()
categorical_features = [f[:-1] for f in x]
categorical_features = categorical_features[:-1]

X_train, X_test, y_train, y_test = train_test_split(train_data_small.loc[:,
train_data_small.columns != 'TARGET'],train_data_small['TARGET'], test_size=0.2)

clf = lgb.LGBMClassifier(n_estimators=10000, objective = 'binary', is_unbalance=True,
metric='auc', learning_rate = 0.05,
```

```
reg_alpha = 0.1, reg_lambda = 0.1,
subsample = 0.8, n_jobs = -1, random_state = 50)

clf.fit(X_train, y_train, eval_metric = 'auc',
    eval_set = [(X_test, y_test), (X_train, y_train)],
    eval_names = ['test', 'train'], categorical_feature = categorical_features,
    early_stopping_rounds = 100, verbose = 200)

# hyper-parameters tuning
from sklearn.metrics import confusion_matrix
y_pred = clf.predict(X_test)
print(confusion_matrix(y_test, y_pred))

from sklearn.model_selection import RandomizedSearchCV
def LgmParamTuning(X_train, y_train):
  # application
  # the last column of train_data is Y
  param_dict = {"num_leaves": [25, 30, 35],
          "reg_alpha": [0, 0.05, 0.1],
          }
          #"min_data_in_leaf": [300, 500, 800]}
  gsearch = GridSearchCV(estimator = lgb.LGBMClassifier(num_iterations=10000,
  objective = 'binary', learning_rate=0.05,
  is_unbalance=True, metric='auc',
  subsample = 0.8, n_jobs = -1, random_state=50, feature_fraction=0.8),
  param_grid = param_dict, scoring='roc_auc')

  gsearch.fit(X_train,y_train)
  return gsearch.best_params_, gsearch.best_score_

# bureau_data analysis and generate new features
def bureau_data_feature_1(df):
  # how many loans will expire in the future
  return sum(df['DAYS_CREDIT_ENDDATE'] > 0) / len(df)
```

```
99
100 def bureau_data_feature_2(df):
101   # how many loans this person has now and past
102   return len(df)
103
104 def bureau_data_feature_3(df):
105   # the ratio of debt and the sum of debt and credit
106   return max(df['AMT_CREDIT_SUM_DEBT'] / (df['AMT_CREDIT_SUM_DEBT'] + df['AMT_CREDIT_SUM']))
107
108 def bureau_data_feature_4(df):
109   # whether this person has prolonged his loan
110   return 1 if np.sum(df['CNT_CREDIT_PROLONG']) > 0 else 0
111
112 def bureau_data_feature_5(df):
113   # the farest due date of a person's loan in the future
114   pos_due_date_ind = df['DAYS_CREDIT_ENDDATE'] > 0
115   tmp = df[pos_due_date_ind]['DAYS_CREDIT_ENDDATE']
116   return 0 if tmp.empty else max(tmp)
117
118 def bureau_data_feature_6(df):
119   # the ratio of overdue and debt
120   return max(df['AMT_CREDIT_SUM_OVERDUE'] / df['AMT_CREDIT_SUM_DEBT'])
121
122 def bureau_data_feature_7_pre(df):
123   # ratio of income to debt
124   # use bureau data and application data
125   # df is a left join table by bureau_df and application_df
126   existing_loan = df['DAYS_CREDIT_ENDDATE'] > 0
127   tmp = df[existing_loan]['AMT_CREDIT_SUM_DEBT']
128
129   debt_sum = np.nan if tmp.empty else sum(tmp)
130   return debt_sum
131
132 def gen_new_feature_from_bureau_data(bureau_data, application_data):
```

```
  grouped_data = bureau_data.groupby('SK_ID_CURR')
  result = pd.DataFrame(index=bureau_data['SK_ID_CURR'].unique())
  # handcrafted features
  result['b_feature_1'] = grouped_data.apply(bureau_data_feature_1)
  result['b_feature_2'] = grouped_data.apply(bureau_data_feature_2)
  result['b_feature_3'] = grouped_data.apply(bureau_data_feature_3)
  result['b_feature_4'] = grouped_data.apply(bureau_data_feature_4)
  result['b_feature_5'] = grouped_data.apply(bureau_data_feature_5)
  result['b_feature_6'] = grouped_data.apply(bureau_data_feature_6)
  debt_sum = grouped_data.apply(bureau_data_feature_7_pre)
  result['b_feature_7'] = application_df.loc[bureau_data['SK_ID_CURR'].unique()]
  ['AMT_INCOME_TOTAL'] / debt_sum
  # auto_generated features
  auto_feature_list = ['DAYS_CREDIT_ENDDATE', 'AMT_CREDIT_SUM', 'AMT_CREDIT_SUM_DEBT',
  'AMT_CREDIT_MAX_OVERDUE']
  stats_dict = {'mean': np.mean, 'min': min, 'max': max, 'std': np.std, 'sum':sum}
  for i in tqdm(range(0, len(auto_feature_list))):
    for k in stats_dict.keys():
      name = 'b_feature_auto_' + auto_feature_list[i] + '_' + k
      result[name] = grouped_data[auto_feature_list[i]].apply(stats_dict[k])
    result
  return result
```

```
# Mannual Feature: the difference between ask and given of previous loan at Home Credit
pre_app_df["amt_diff"] = pre_app_df["AMT_CREDIT"] - pre_app_df["AMT_APPLICATION"]
amt_diff = pre_app_df.groupby("SK_ID_CURR").sum()["amt_diff"].to_frame()
amt_diff.columns = ["amt_diff"]
app_train_df = app_train_df.join(amt_diff, on="SK_ID_CURR")
app_train_df["amt_diff"] = app_train_df["amt_diff"].fillna(value=0)

# Construct new features for installments_payments
feature_select = ['SK_ID_PREV','SK_ID_CURR','NUM_INSTALMENT_VERSION','NUM_INSTALMENT_NUMBER']
instal_pay_df = installments_payments[feature_select]
```

```
instal_pay_df['Overdue_Days'] = installments_payments['DAYS_INSTALMENT']
-installments_payments['DAYS_ENTRY_PAYMENT']
instal_pay_df['Still_Owed'] =  installments_payments['AMT_INSTALMENT']
-installments_payments['AMT_PAYMENT']

# Construct new features for credit_card_balance
feature_select = ['SK_ID_PREV','SK_ID_CURR','MONTHS_BALANCE','AMT_BALANCE',
                  'AMT_CREDIT_LIMIT_ACTUAL','AMT_DRAWINGS_ATM_CURRENT','AMT_DRAWINGS_CURRENT',
                  'AMT_DRAWINGS_OTHER_CURRENT','AMT_DRAWINGS_POS_CURRENT','SK_DPD_DEF','AMT_RECIVABLE',
                  'NAME_CONTRACT_STATUS']
cre_card_df = credit_card_balance[feature_select]
cre_card_df['Excessive_Repay'] = credit_card_balance['AMT_PAYMENT_CURRENT']
-credit_card_balance['AMT_INST_MIN_REGULARITY']

# Mannual Feature: the total number of previous loan at Home Credit
num_previous_loan = pre_app_df.groupby("SK_ID_CURR").count()["SK_ID_PREV"].to_frame()
num_previous_loan.columns = ["num_previous_loan"]
app_train_df = app_train_df.join(num_previous_loan, on="SK_ID_CURR")
app_train_df["num_previous_loan"] = app_train_df["num_previous_loan"].fillna(value=0)

# Mannual Feature: percentage that previous loan at Home Credit was insured
def insurance_ratio(df):
  return sum(df['NFLAG_INSURED_ON_APPROVAL'] == 1) / len(df['NFLAG_INSURED_ON_APPROVAL'])
insurance_ratio = pre_app_df.groupby('SK_ID_CURR').apply(insurance_ratio).to_frame()
insurance_ratio.columns = ["insurance_ratio"]
app_train_df = app_train_df.join(insurance_ratio, on="SK_ID_CURR")
```

Listing 1: Manual Generated Features

```

def del_missing_much(df):
  total = df.isnull().sum().sort_values(ascending = False)
  percent = (df.isnull().sum()/df.shape[0]*100).sort_values(ascending = False)
```

```
  missing_instal_pay_df  = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
  missing_instal_pay_df = missing_instal_pay_df.loc[missing_instal_pay_df['Percent']<75]
  column = missing_instal_pay_df.index
  df_new = df[column]
  df_new = df_new[list(df_new.columns)[::-1]]
  return df[column]

bureau_df = del_missing_much(bureau_df)
pre_app_df = del_missing_much(pre_app_df)
cre_card_df = del_missing_much(cre_card_df)
instal_pay_df = del_missing_much(instal_pay_df)
app_train_df = del_missing_much(app_train_df)
extra_df = del_missing_much(extra_df)

app_train_df['AMT_INCOME_TOTAL'].replace({117000000: np.nan}, inplace = True)
app_train_df['DAYS_EMPLOYED'].replace({365243:np.nan},inplace=True)
app_train_df['REGION_RATING_CLIENT_W_CITY'].replace({-1:np.nan},inplace = True)
app_train_df['OBS_30_CNT_SOCIAL_CIRCLE'].replace({354:np.nan},inplace = True)
app_train_df['OBS_60_CNT_SOCIAL_CIRCLE'].replace({344:np.nan},inplace = True)
app_train_df['AMT_REQ_CREDIT_BUREAU_QRT'].replace({261:np.nan},inplace = True)
idx = bureau['DAYS_CREDIT_ENDDATE'].loc[bureau['DAYS_CREDIT_ENDDATE']<-20000].index
bureau['DAYS_CREDIT_ENDDATE'].loc[idx] = np.nan
idx = bureau['DAYS_ENDDATE_FACT'].loc[bureau['DAYS_ENDDATE_FACT']<-4000].index
bureau['DAYS_ENDDATE_FACT'].loc[idx] = np.nan
idx = bureau['AMT_CREDIT_SUM'].loc[bureau['AMT_CREDIT_SUM']>10000000].index
bureau['AMT_CREDIT_SUM'].loc[idx] = np.nan
idx = bureau['AMT_CREDIT_SUM_DEBT'].loc[bureau['AMT_CREDIT_SUM_DEBT']<0].index
bureau['AMT_CREDIT_SUM'].loc[idx] = 0
idx = bureau['AMT_CREDIT_SUM_DEBT'].loc[bureau['AMT_CREDIT_SUM_DEBT']>50000000].index
bureau['AMT_CREDIT_SUM'].loc[idx] = np.nan
idx = bureau['AMT_ANNUITY'].loc[bureau['AMT_ANNUITY']>10000000].index
bureau['AMT_ANNUITY'].loc[idx] = np.nan
idx = pre_app_df['AMT_DOWN_PAYMENT'].loc[bureau['AMT_DOWN_PAYMENT']<0].index
pre_app_df['AMT_DOWN_PAYMENT'].loc[idx] = 0
```

```
idx = cre_card_df['AMT_DRAWINGS_ATM_CURRENT'].loc[bureau['AMT_DRAWINGS_ATM_CURRENT']<0].index
cre_card_df['AMT_DRAWINGS_ATM_CURRENT'].loc[idx] = np.nan
idx = cre_card_df['AMT_DRAWINGS_CURRENT'].loc[bureau['AMT_DRAWINGS_CURRENT']<0].index
cre_card_df['AMT_DRAWINGS_CURRENT'].loc[idx] = np.nan

# Label encoding of binary categorical variables
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

def Transfer_category(df):

  for col in df:
    if df[col].dtype == 'object':
        # If 2 or fewer unique categories
        if len(list(df[col].unique())) <= 2:
            # Train on the training data
            le.fit(df[col])
            # Transform both training and testing data
            df[col] = le.transform(df[col])
  # one-hot encoding of muti-categorical variables
  df = pd.get_dummies(df)
  try:
    Obj_columns = df.select_dtypes(['object']).columns
  except Exception:
    pass
  df[Obj_columns] = df[Obj_columns].fillna(df[Obj_columns].mode())
  return df

app_train_df = Transfer_category(app_train_df)
instal_pay_df = Transfer_category(instal_pay_df)
cre_card_df = Transfer_category(cre_card_df)

# Threshold for removing correlated variables
threshold = 0.9
```

```

# Absolute value correlation matrix
corr_matrix = final_app_df.iloc[0:10000,:].corr().abs()
corr_matrix.head()

upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
upper.shape
# Select columns with correlations above threshold
to_drop = [column for column in upper.columns if any(upper[column] > threshold)]
final_app_df = final_app_df.drop(columns = to_drop)
```

Listing 2: Data Cleaning

```
from sklearn.feature_selection import RFE
rfe_selector = RFE(estimator=LogisticRegression(), n_features_to_select=150, step=10, verbose=5)
rfe_selector.fit(app_train_df, target)
rfe_support = rfe_selector.get_support()
rfe_feature = app_train_df.loc[:,rfe_support].columns.tolist()
print(str(len(rfe_feature)), 'selected features')
```

Listing 3: Feature Selection

```
final_app_df_original = final_app_df.fillna(final_app_df.mean())
final_app_df_original = final_app_df_original.replace([np.inf, -np.inf], np.nan)
final_app_df_original.dropna(axis =1, inplace =True )
# feature importance
def get_feature_importances(data, shuffle, seed=None):
    # Gather real features
    train_features = [f for f in data if f not in ['TARGET', 'SK_ID_CURR']]
    # Go over fold and keep track of CV score (train and valid) and feature importances

    # Shuffle target if required
```

```
    y = data['TARGET'].copy()
    if shuffle:
        # Here you could as well use a binomial distribution
        y = data['TARGET'].copy().sample(frac=1.0)

    # Fit LightGBM in RF mode, yes it's quicker than sklearn RandomForest
    dtrain = lgb.Dataset(data[train_features], y, free_raw_data=False, silent=True)
    lgb_params = {
        'objective': 'binary',
        'boosting_type': 'rf',
        'subsample': 0.623,
        'colsample_bytree': 0.7,
        'num_leaves': 127,
        'max_depth': 8,
        'seed': seed,
        'bagging_freq': 1,
        'n_jobs': 4
    }

    # Fit the model
    clf = lgb.train(params=lgb_params, train_set=dtrain, num_boost_round=200)

    # Get feature importances
    imp_df = pd.DataFrame()
    imp_df["feature"] = list(train_features)
    imp_df["importance_gain"] = clf.feature_importance(importance_type='gain')
    imp_df["importance_split"] = clf.feature_importance(importance_type='split')
    imp_df['trn_score'] = roc_auc_score(y, clf.predict(data[train_features]))

    return imp_df

df=final_app_df_original.drop('TARGET', 1)
f , ax = plt.subplots(figsize = (14,12))
plt.title('Correlation of Features- HeatMap',y=1,size=16)
```

```
sns.heatmap(df.corr(),square = True,  vmax=0.8)



def logistic_regression(df, y=None):
  if y is None:
    y = df.pop("TARGET")

  X_train, X_test, y_train, y_test = train_test_split(df, y, test_size=0.3, random_state=0)
  ns_probs = [0 for _ in range(len(y_test))]
  lr = LogisticRegression(random_state=0).fit(X_train, y_train)
  lr_probs = lr.predict_proba(X_test)
  lr_probs = lr_probs[:, 1]
  ns_auc = roc_auc_score(y_test, ns_probs)
  lr_auc = roc_auc_score(y_test, lr_probs)
  ns_auc = roc_auc_score(y_test, ns_probs)
  lr_auc = roc_auc_score(y_test, lr_probs)
  print('Logistic: ROC AUC=%.3f' % (lr_auc))
  ns_fpr, ns_tpr, _ = roc_curve(y_test, ns_probs)
  lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_probs)
  plt.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
  plt.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.legend()
  plt.show()
```

Listing 4: Logistic Regression

```
def auto_gen_new_feature(df):
  grouped_data = df.groupby('SK_ID_CURR')
  result = pd.DataFrame(index=app_train_df['SK_ID_CURR'].unique())
  # auto_feature_list = ['AMT_ANNUITY', 'AMT_APPLICATION', 'AMT_CREDIT',
  'RATE_DOWN_PAYMENT', 'AMT_DOWN_PAYMENT', 'AMT_GOODS_PRICE', 'DAYS_DECISION', 'CNT_PAYMENT']
```

```
auto_feature_list = ['MONTHS_BALANCE', 'CNT_INSTALMENT',
'CNT_INSTALMENT_FUTURE', 'SK_DPD', 'SK_DPD_DEF']
stats_dict = {'mean': np.mean, 'min': min, 'max': max, 'std': np.std, 'sum':sum}
for i in range(len(auto_feature_list)):
  for k in tqdm(stats_dict.keys()):
    name = auto_feature_list[i] + '_' + k
    result[name] = grouped_data[auto_feature_list[i]].apply(stats_dict[k])
return result
```

Listing 5: Auto Generated Features