BA723 Business Analytics Capstone

Credit Risk Modeling using alternative consumer data

Submitted by:

Marian Ilagan (301236806)

Submitted to: David Parent

Due date: August 18, 2023

EXCUTIVE SUMMARY

A recent study conducted by TransUnion has highlighted the significant challenge posed by the reliance on conventional credit models. This challenge directly affects a substantial portion of the Canadian adult population—approximately 9.6 million individuals—who find themselves having limited financial access. This project seeks to address this pressing issue by examining into the idea of incorporating alternative data. The primary objective of this initiative is to leverage alternative data to gain valuable insights into the distinctive characteristics of clients who exhibit a likelihood of defaulting. This exploration has been facilitated through the utilization of various machine learning models such as Decision Tree, Logistic Regression, XGboost, Random Forest and SGD Classifier. The Random Forest model provided the best scores with an ROC-AUC score of 70% and achieving an accuracy rate of 77%. In addition to traditional credit scores, our analysis has presented three important features that play a decisive role in predicting default probabilities. These features include the average amount drawn from credit cards, the duration remaining before the credit ends in the credit bureau (with a period exceeding 2 years), and the individual's highest educational attainment—favoring those with a secondary education. The results show that the use of alternative data can benefit both the company and the customers by offering more credit to customers and empowering financial inclusion.

1 Table of Contents

1 INTRODUCTION

1.1 Background

Consumers take out loans for a variety of reasons such as paying for education, starting up a business, buying a car or a home, or big purchases of other products. Some people, even the wealthy ones, also prefer loans to save their available cash for future emergencies. Loans also play an integral part in a bank's revenue model since most of its income is from the interest generated from loans. However, providing loans is subject to credit risk or the probability that the lender will not be paid on a loan and lose the money to a borrower. One way that lenders use to manage credit risk is by assessing the probability of default of a loan applicant also called default risk. Default risk is defined as the likelihood that a borrower won't be able to make their required debt payments to a lender (Kagan, 2023).

For individual consumers, default risk can be measured based on their credit reports and credit scores. Credit bureaus gather and prepare credit reports from the information being sent by an individual's previous and current lenders which shows their diligence in paying loans. Based on these reports, the credit score will then be calculated, affecting the individual's future loan applications, rental applications, credit card applications, car leasing, and even interest rates. The assumption is that a consumer who has established a record of paying their bills on time is less likely to default in the future; therefore, lenders prefer a good credit score to minimize credit risk.

1.2 Problem Statement

The dependency on credit scores for default risk assessment of most financial institutions creates a setback for credit-disadvantaged customers who accounts for more than 9 million Canadian consumers or 31% of the adult population (TransUnion, 2023). These are the credit-unserved and underserved consumer segments who are struggling to access financial products and services due to no, or not enough, credit history. For instance, new immigrants in the country usually have a slimmer chance of getting approved for a car loan or if not, have higher interest rates since they have lower credit scores. TransUnion suggests that the use of alternative data can provide a bigger picture of the client's repayment capabilities.

An article from FICO.com defines alternative data as information that are usually not part of a credit report or not directly related to a consumer's credit behaviour.

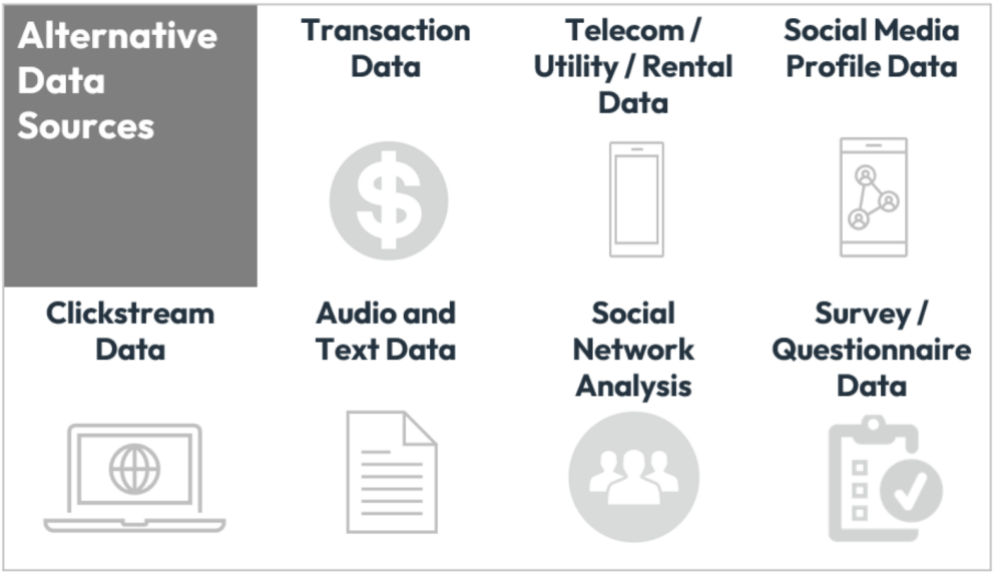


Figure 1. Alternative Data Sources

The alternative data sources mentioned in Figure 1 above are explained below:

- **Transaction data** – these are transactional data from credit and debit cards which could be used to create new key characteristics that are more predictive like Cash to Total Spend ratio for the last X weeks.
- **Clickstream data** – a record of users' online browsing behaviour on a website or application.
- **Telecom/Utility/Rental data** – considered as credit history but categorized as an alternative because these are usually not part of credit reports.
- **Social Media Profile data** – deriving value from metadata like the number of posts and frequency. However, this could be against the regulatory laws.
- **Audio and Text data** – information from application forms, recorded customer service calls or collection calls.
- **Social Network Analysis –** understanding of customer's risk by identifying their social connections.
- **Survey / Questionnaire data** – allows lenders to assess a client's risk through psychometrics.

1.3 Objectives & Measurement

1.3.1 Business Objective

This project aims to incorporate the use of alternative consumer data to predict loan repayment capabilities of the underserved population, ensuring that those who are capable will not be rejected. Understanding the overall financial capacity of underserved consumers will help lenders offer more credit to this segment, therefore generating new credit applications and increasing the customer base.

1.3.2 Analytics Objective

The objectives of this project are as follows:

1. Gain insights from the client's traditional and alternative data to reveal the key characteristics that are important in predicting a credit default.
2. Create different predictive models and identify the best model that will accurately classify a credit-worthy customer from a potential defaulter.
3. Aim to improve the performance of the machine learning model to ensure that our final model will provide the most optimal solution, as the cost of making a mistake in predicting a default can be very high for the company as well as the cost of rejecting a potential customer.

The project will seek to test the hypothesis: *How the use of alternative data in credit risk modelling will help financial institutions determine credit-worthy customers even with little or no credit history in the country?*

Key questions:

1. What key characteristics have the highest importance in determining a default?
2. What variables can accurately classify a defaulter and a non-defaulter?
3. How can we be sure that the model provides the optimal solution (minimizing the cost of error)?

1.3.3 Measurement

The best model will be chosen based on the three metrics, ROC-AUC score, F1-score, and accuracy. Since our dataset is highly imbalanced, ROC-AUC score will be the primary metric since this provides a more comprehensive view of model performance. Secondary metric to be used in the F1-score which shows the balance of correctly predicting defaulted clients and minimizing the false negatives. Finally, accuracy is the last metric since it talks about how accurate our model is regardless of the class.

1.4 Assumptions and Limitations

This project assumes that all variables provided are free and acceptable to use based on the regulatory laws of Canada in the finance industry. Also, we assume that data variables are legally available in Canada.

The dataset used in this study has a high imbalance between class distribution with only 9% of the data categorized as default, a vast number of outliers, skewness, and numerous observations. Because of this, our models have the tendency to be overfit and become complex which can affect interpretability.

2 DATA SOURCE

2.1 Data Set Introduction

The dataset used for this project is prepared by Home Credit for one of the Kaggle competitions. Home Credit is an international consumer finance, which provides a positive and safe borrowing experience to people with little to no credit history using alternative data in assessing client's repayment capabilities. The dataset is composed of the main table, application_train.csv and 6 supporting tables about the clients' previous application, bureau data, previous point of sale (POS) and cash loans, installment payments and previous credit card loans (see Figure 2 below). Each supporting table will be used to create new columns for the main table based on SK_ID_CURR.

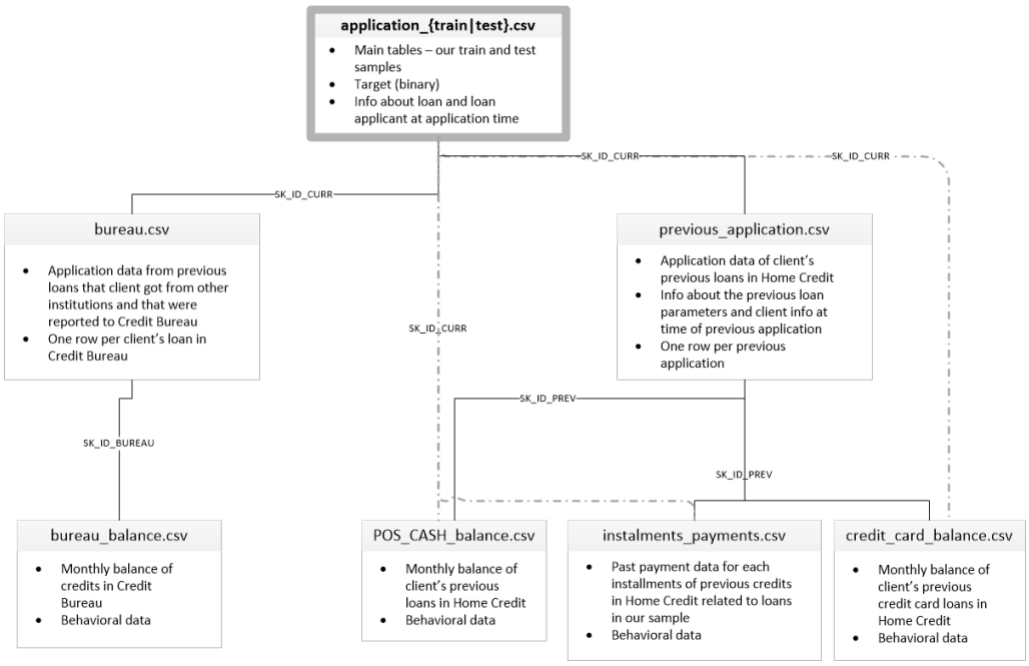


Figure 2 Home Credit - Data model

Application_{train|test}.csv – this is the main table to be used for this project. It contains static data for all applications of each client. Each observation represents one loan in the data sample. Application_train.csv was used since this table contains the target labels 0 for clients who did not default and 1 for defaulted clients.

Previous_application.csv – contains all previous applications of each client in the application_train.csv. One row corresponds to a specific application in the past. A loan can be a Consumer loan, cash loan or revolving loan.

POS_CASH_balance.csv – behavioural data about the monthly balance snapshots of previous POS (point of sale) and cash loans that the client had in 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.

credit_card_balance.csv - Monthly balance snapshots of previous credit cards that the applicant has with Home Credit. This table has one row for each month of the history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in the sample.

installments_payments.csv - 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 payments. 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 the sample.

bureau.csv - 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 the sample, there are as many rows as the number of credits the client had in the Credit Bureau before the application date.

bureau_balance.csv - Monthly balances of previous credits in Credit Bureau. This table has one row for each month of the history of every previous credit reported to the Credit Bureau.

2.2 Exclusions

For this study, only those clients that have previous credit history and returning customers will be considered. Therefore, the observations not in credit bureau tables and previous application tables will be excluded.

2.3 Initial Data Cleansing / Preparation

The seven files were loaded in Python with the following dimensions. The main table, application_df, has 307,5011 observations and 122 features including target variable where 1 means the client has defaulted on the loan and 0 when they did not default.

	application_df	previous_df	bureau_df	bureaubal_df	POS_df	prev_cc_df	installments_df
rows	307511	1670214	1716428	27299925	10001358	3840312	3840312
columns	122	37	17	3	8	23	23

Figure 3 Data dimensions Summary

To prepare the final table, the 6 supporting tables were checked to identify the columns that might be useful and merged into the main table, application_df.

2.3.1 Duplicates

There are no duplicate rows for all the tables.

```
#Checking duplicated rows per table: There are no duplicate rows
duplicated_app = application_df[application_df.duplicated()]
duplicated_prev = previous_df[previous_df.duplicated()]
duplicated_bureau = bureau_df[bureau_df.duplicated()]
duplicated_bureaubal = bureaubal_df[bureaubal_df.duplicated()]
duplicated_pos = pos_df[pos_df.duplicated()]
duplicated_prevcc = prev_cc_df[prev_cc_df.duplicated()]
duplicated_install = installments_df[installments_df.duplicated()]

print("Duplicate Rows in application_df:", len(duplicated_app.index))
print("Duplicate Rows in previous_df:", len(duplicated_prev.index))
print("Duplicate Rows in bureau_df:", len(duplicated_bureau.index))
print("Duplicate Rows in bureaubal_df:", len(duplicated_bureaubal.index))
print("Duplicate Rows in pos_df:", len(duplicated_pos.index))
print("Duplicate Rows in prev_cc_df:", len(duplicated_prevcc.index))
print("Duplicate Rows in installment_df:", len(duplicated_install.index))
```

✓ 1m 2.9s

```
Duplicate Rows in application_df: 0
Duplicate Rows in previous_df: 0
Duplicate Rows in bureau_df: 0
Duplicate Rows in bureaubal_df: 0
Duplicate Rows in pos_df: 0
Duplicate Rows in prev_cc_df: 0
Duplicate Rows in installment_df: 0
```

The below values show the number of unique SK_ID_CURR of application_df that is also in the other tables.

```
Number of overlapping SK_ID_CURR of application_df and previous_df: 291057
Number of overlapping SK_ID_CURR of application_df and bureau_df: 263491
Number of overlapping SK_ID_CURR of application_df and pos_df: 289444
Number of overlapping SK_ID_CURR of application_df and prev_cc_df: 86905
Number of overlapping SK_ID_CURR of application_df and installments_df: 291643
```

2.3.2 Sorting time-based data

Three tables are time-based where the MONTHS_BALANCE and NUM_INSTALMENT_NUMBER columns how recent the update is in the table. Therefore, we sorted these tables based on the mentioned columns.

```
installments_df = installments_df.sort_values(by = ['SK_ID_CURR','SK_ID_PREV','NUM_INSTALMENT_NUMBER'], ascending = True)
prev_cc_df = prev_cc_df.sort_values(by = ['SK_ID_PREV','MONTHS_BALANCE'], ascending = [1,0])
pos_df = pos_df.sort_values(by = ['SK_ID_PREV','MONTHS_BALANCE'], ascending = [1,0])
```

✓ 34.7s Python

2.3.3 Checking of each table

Bureau balance data (bureaubal_df)

This table only contains three columns that shows the previous credit bureau credit and the history of status. We will use this data to check the number of times a certain SK_ID_BUREAU has a delayed payment, on time payment, if it was written off, as well as the most recent status if it is active or closed.

Table 1 Bureau balance table - Data description from Home Credit

Column	Data type	Description
SK_ID_BUREAU	Integer	Recoded ID of previous Credit Bureau credit related to our loan (unique coding for each loan application)
MONTHS_BALANCE	Integer	Month of balance relative to application date (-1 means the freshest balance date)
STATUS	Object	Status of Credit Bureau loan during the month (active, closed, DPD0-30, C means closed, X means status unknown, 0 means no DPD, 1 means maximal did during month between 1-30, 2 means DPD 31-60, 3 DPD 61-90, 4 DPD of 91 - 120, Ö 5 means DPD 120+ or sold or written off])

Bureau data (bureau_df)

Bureau_df contains SK_BUREAU_ID that will be joined with SK_BUREAU_ID of bureaubal_df in order to get the aggregated columns. SK_ID_CURR will also be used to aggregate information from bureau data and join with the main application_df table. One SK_ID_CURR (client) can have multiple records (SK_ID_BUREAU) in bureau table. Some features that will be created are count of active loans, average overdue amount and number of secured / unsecured loans.

Table 2 Bureau data table - Data description from Home Credit

Column	Data type	Description
SK_ID_CURR	Integer	ID of loan in our sample - one loan in our sample can have 0,1,2 or more related previous credits in credit bureau
SK_BUREAU_ID	Integer	Recoded ID of previous Credit Bureau credit related to our loan (unique coding for each loan application)
CREDIT_ACTIVE	Object	Status of the Credit Bureau (CB) reported credits
CREDIT_CURRENCY	Object	Recoded currency of the Credit Bureau credit
DAYS_CREDIT	Integer	How many days before current application did client apply for Credit Bureau credit
CREDIT_DAY_OVERDUE	Integer	Number of days past due on CB credit at the time of application for related loan in our sample
DAYS_CREDIT_ENDDATE	Float	Remaining duration of CB credit (in days) at the time of application in Home Credit
DAYS_ENDDATE_FACT	Float	Days since CB credit ended at the time of application in Home Credit (only for closed credit)
AMT_CREDIT_MAX_OVERDUE	Float	Maximal amount overdue on the Credit Bureau credit so far (at application date of loan in our sample)
CNT_CREDIT_PROLONG	Integer	How many times was the Credit Bureau credit prolonged
AMT_CREDIT_SUM	Float	Current credit amount for the Credit Bureau credit
AMT_CREDIT_SUM_DEBT	Float	Current debt on Credit Bureau credit
AMT_CREDIT_SUM_LIMIT	Float	Current credit limit of credit card reported in Credit Bureau
AMT_CREDIT_SUM_OVERDUE	Float	Current amount overdue on Credit Bureau credit
CREDIT_TYPE	Object	Type of Credit Bureau credit (Car, cash,...)

DAYS_CREDIT_UPDATE	Integer	How many days before loan application did last information about the Credit Bureau credit come
AMT_ANNUITY	Float	Annuity of the Credit Bureau credit

POS_CASH balance (pos_df)

POS_CASH balance table contains payment behavior of a client for each previous application record (SK_ID_PREV). Some columns to be used are average number of delayed payment (SK_DPD) and remaining installments left to pay.

Table 3 POS_CASH balance table - Data description from Home Credit

Column	Data type	Description
SK_ID_PREV	Integer	ID of previous credit in Home Credit related to loan in our sample. (One loan in our sample can have 0,1,2 or more previous loans in Home Credit)
SK_ID_CURR	Integer	ID of loan in our sample
MONTHS_BALANCE	Integer	Month of balance relative to application date (-1 means the information to the freshest monthly snapshot, 0 means the information at application - often it will be the same as -1 as many banks are not updating the information to Credit Bureau regularly)
CNT_INSTALMENT	Float	Term of previous credit (can change over time)
CNT_INSTALMENT_FUTURE	Float	Installments left to pay on the previous credit
NAME_CONTRACT_STATUS	Object	Contract status during the month
SK_DPD	Integer	DPD (days past due) during the month of previous credit
SK_DPD_DEF	Integer	DPD during the month with tolerance (debts with low loan amounts are ignored) of the previous credit

Credit card balance data (prev_cc_df)

In this table, we plan to use the ratio of total amount cash drawings (AMT_DRAWINGS_ATM_CURRENT) over total drawings (AMT_DRAWINGS_CURRENT)

which will tell us the percentage of using the credit card for cash withdrawals, as well as the average number of DPD days (delayed payment).

Table 4 Credit card balance - Data description from Home Credit

Columns	Data Type	Description
SK_ID_PREV	Integer	ID of previous credit in Home credit related to loan in our sample. (One loan in our sample can have 0,1,2 or more previous loans in Home Credit)
SK_ID_CURR	Integer	ID of loan in our sample
MONTHS_BALANCE	Integer	Month of balance relative to application date (-1 means the freshest balance date)
AMT_BALANCE	Float	Balance during the month of previous credit
AMT_CREDIT_LIMIT_ACTUAL	Integer	Credit card limit during the month of the previous credit
AMT_DRAWINGS_ATM_CURRENT	Float	Amount drawing at ATM during the month of the previous credit
AMT_DRAWINGS_CURRENT	Float	Amount drawing during the month of the previous credit
AMT_DRAWINGS_OTHER_CURRENT	Float	Amount of other drawings during the month of the previous credit
AMT_DRAWINGS_POS_CURRENT	Float	Amount drawing or buying goods during the month of the previous credit
AMT_INST_MIN_REGULARITY	Float	Minimal installment for this month of the previous credit
AMT_PAYMENT_CURRENT	Float	How much did the client pay during the month on the previous credit
AMT_PAYMENT_TOTAL_CURRENT	Float	How much did the client pay during the month in total on the previous credit
AMT_RECEIVABLE_PRINCIPAL	Float	Amount receivable for principal on the previous credit
AMT_RECIVABLE	Float	Amount receivable on the previous credit
AMT_TOTAL_RECEIVABLE	Float	Total amount receivable on the previous credit
CNT_DRAWINGS_ATM_CURRENT	Float	Number of drawings at ATM during this month on the previous credit

CNT_DRAWINGS_CURRENT	Integer	Number of drawings during this month on the previous credit
CNT_DRAWINGS_OTHER_CURRENT	Float	Number of other drawings during this month on the previous credit
CNT_DRAWINGS_POS_CURRENT	Float	Number of drawings for goods during this month on the previous credit
CNT_INSTALMENT_MATURE_CUM	Float	Number of paid installments on the previous credit
NAME_CONTRACT_STATUS	Object	Contract status (active signed,...) on the previous credit
SK_DPD	Integer	DPD (Days past due) during the month on the previous credit
SK_DPD_DEF	Integer	DPD (Days past due) during the month with tolerance (debts with low loan amounts are ignored) of the previous credit

Installments data (installments_df)

From this set we will use the difference between DAYS_INSTALMENT and DAYS_ENTRY_PAYMENT to know how many days the client delays the payment as well as the AMT_INSTALMENT minus AMT_PAYMENT to check if the client paid the full amount.

Table 5 Installments data - Data description from Home Credit

Columns	Data Type	Description
SK_ID_PREV	Integer	ID of previous credit in Home credit related to loan in our sample. (One loan in our sample can have 0,1,2 or more previous loans in Home Credit)
SK_ID_CURR	Integer	ID of loan in our sample
NUM_INSTALMENT_VERSION	Float	Version of installment calendar (0 is for credit card) of previous credit. Change of installment version from month to month signifies that some parameter of payment calendar has changed
NUM_INSTALMENT_NUMBER	Integer	On which installment we observe payment
DAYS_INSTALMENT	Float	When the installment of previous credit was supposed to be paid (relative to application date of current loan)

DAYS_ENTRY_PAYMENT	Float	When was the installments of previous credit paid actually (relative to application date of current loan)
AMT_INSTALMENT	Float	What was the prescribed installment amount of previous credit on this installment
AMT_PAYMENT	Float	What the client actually paid on previous credit on this installment

Previous application table (previous_df)

All client details of the each previous applications in Home Credit are contained in this file including the purpose of the loan, type of loan requested, if it was rejected and why.

Table 6 Previous application information - Data description from Home Credit

Column	Data Type	Description
SK_ID_PREV	Integer	ID of previous credit in Home credit related to loan in our sample. (One loan in our sample can have 0,1,2 or more previous loan applications in Home Credit, previous application could, but not necessarily have to lead to credit)
SK_ID_CURR	Integer	ID of loan in our sample
NAME_CONTRACT_TYPE	Object	Contract product type (Cash loan, consumer loan [POS] ,...) of the previous application
AMT_ANNUITY	Float	Annuity of previous application
AMT_APPLICATION	Float	For how much credit did client ask on the previous application
AMT_CREDIT	Float	Final credit amount on the previous application. This differs from AMT_APPLICATION in a way that the AMT_APPLICATION is the amount for which the client initially applied for, but during our approval process he could have received different amount - AMT_CREDIT
AMT_DOWN_PAYMENT	Float	Down payment on the previous application
AMT_GOODS_PRICE	Float	Goods price of good that client asked for (if applicable) on the previous application
WEEKDAY_APPR_PROCESS_START	Object	On which day of the week did the client apply for previous application
HOUR_APPR_PROCESS_START	Integer	Approximately at what day hour did the client apply for the previous application

FLAG_LAST_APPL_PER_CONTRACT	Object	Flag if it was last application for the previous contract. Sometimes by mistake of client or our clerk there could be more applications for one single contract
NFLAG_LAST_APPL_IN_DAY	Integer	Flag if the application was the last application per day of the client. Sometimes clients apply for more applications a day. Rarely it could also be error in our system that one application is in the database twice
NFLAG_MICRO_CASH	Float	Flag Micro finance loan
RATE_DOWN_PAYMENT	Float	Down payment rate normalized on previous credit
RATE_INTEREST_PRIMARY	Float	Interest rate normalized on previous credit
RATE_INTEREST_PRIVILEGED	Float	Interest rate normalized on previous credit
NAME_CASH_LOAN_PURPOSE	Object	Purpose of the cash loan
NAME_CONTRACT_STATUS	Object	Contract status (approved, cancelled, ...) of previous application
DAYS_DECISION	Integer	Relative to current application when was the decision about previous application made
NAME_PAYMENT_TYPE	Object	Payment method that client chose to pay for the previous application
CODE_REJECT_REASON	Object	Why was the previous application rejected
NAME_TYPE_SUITE	Object	Who accompanied client when applying for the previous application
NAME_CLIENT_TYPE	Object	Was the client old or new client when applying for the previous application
NAME_GOODS_CATEGORY	Object	What kind of goods did the client apply for in the previous application
NAME_PORTFOLIO	Object	Was the previous application for CASH, POS, CAR, Ö
NAME_PRODUCT_TYPE	Object	Was the previous application x-sell o walk-in
CHANNEL_TYPE	Object	Through which channel we acquired the client on the previous application
SELLERPLACE_AREA	Integer	Selling area of seller place of the previous application
NAME_SELLER_INDUSTRY	Object	The industry of the seller
CNT_PAYMENT	Float	Term of previous credit at application of the previous application

NAME_YIELD_GROUP	Object	Grouped interest rate into small medium and high of the previous application
PRODUCT_COMBINATION	Object	Detailed product combination of the previous application
DAYS_FIRST_DRAWING	Float	Relative to application date of current application when was the first disbursement of the previous application
DAYS_FIRST_DUE	Float	Relative to application date of current application when was the first due supposed to be of the previous application
DAYS_LAST_DUE_1ST_VERSION	Float	Relative to application date of current application when was the first due of the previous application
DAYS_LAST_DUE	Float	Relative to application date of current application when was the last due date of the previous application
DAYS_TERMINATION	Float	Relative to application date of current application when was the expected termination of the previous application
NFLAG_INSURED_ON_APPROVAL	Float	Did the client requested insurance during the previous application

2.3.4 Removing data from exclusions

As stated in Exclusions section 2.2, new clients and those who have no record in Credit bureau will be excluded from the modeling. The number of remaining observations is now down to 88,109.

2.3.5 Feature Creation

Bureaubal_df data are joined with bureau_df by SK_ID_BUREAU to create the following columns before merging to application_df by aggregating per SK_ID_CURR. Please see data dictionary for the description of each column.

[CB_TOTAL_LOANS, CB_PCT_ACTIVE_LOANS, CB_SEC_LOAN, CB_UNSEC_LOAN, CB_MIN_DAYS_CREDIT, CB_AVG_DAYS_CREDIT_ENDDATE, CB_AVG_DAYS_ENDDATE_FACT, CB_TOTAL_CNT_CREDIT_PROLONG, CB_AMT_CREDIT_SUM_OVERDUE, CB_PCT_DEBT, CB_AMT_CREDIT_SUM_DEBT, CB_MONTHS_0_AVG, CB_MONTHS_1_AVG, CB_MONTHS_2_AVG, CB_MONTHS_3_AVG, CB_MONTHS_4_AVG, CB_MONTHS_5_AVG]

Some columns from prev_cc_df, installments_df and pos_df were also aggregated and joined to previous_df using SK_ID_PREV before combining with application_df per SK_ID_CURR.

[CNT_ACCOUNT_TYPES, CNT_ACTIVE_LOANS, PREV_AVG_DPD_DAYS, PREV_AVG_CNT_DPD, PREV_SUM_AMT_ANNUITY, PREV_SUM_AMT_CREDIT, PREV_DEBT_TO_CREDIT_RATIO, CC_PCT_CASH_DRAWINGS, CC_AVG_AMT_DRAWINGS, (POS_INSTALLMENTS_FUTURE) POS_CNT_INSTALLMENTS_FUTURE, INS_AVG_LATE_PAYMENT, INS_AVG_LESS_PAYMENT]

2.4 Data Dictionary

After merging the columns from credit bureau and previous applications, the final table to be analyzed has 33,032 observations and 151 columns. For the complete list of the columns and corresponding descriptions, please refer to the Appendix Table 14.

3 DATA EXPLORATION

3.1 Data Structure

Final table has 151 columns where 90 are categorized as float data type, 45 are integer and 16 are string. It can be seen that most of the integer variables are actually binary values. For easier visualization of outliers and distribution, these are converted to object data type.

Data Summary

dataframe	Values
Number of rows	33032
Number of columns	151

Data Types

Column Type	Count
float64	90
int64	45
string	16

```
application_df -  {dtype('int64')} - Number of columns are 45 :
['SK_ID_CURR', 'TARGET', 'CNT_CHILDREN', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL', 'FLAG_EMP_PHONE',
'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY',
'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', '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',
'CB_TOTAL_LOANS', 'CB_MIN_DAYS_CREDIT', 'CNT_ACCOUNT_TYPES', 'CNT_ACTIVE_LOANS']
_____________________________________________
application_df -  {dtype('O')} - Number of columns are 16 :
['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE',
'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE', 'WEEKDAY_APPR_PROCESS_START',
'ORGANIZATION_TYPE', 'FONDKAPREMONT_MODE', 'HOUSETYPE_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE']
_____________________________________________
application_df -  {dtype('float64')} - Number of columns are 90 :
['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'REGION_POPULATION_RELATIVE', 'DAYS_REGISTRATION',
'OWN_CAR_AGE', 'CNT_FAM_MEMBERS', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'APARTMENTS_AVG', 'BASEMENTAREA_AVG',
'YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BUILD_AVG', 'COMMONAREA_AVG', 'ELEVATORS_AVG', 'ENTRANCES_AVG', 'FLOORSMAX_AVG',
'FLOORSMIN_AVG', 'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG', 'NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAREA_AVG',
'APARTMENTS_MODE', 'BASEMENTAREA_MODE', 'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE', 'COMMONAREA_MODE',
'ELEVATORS_MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE', 'FLOORSMIN_MODE', 'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE',
'LIVINGAREA_MODE', 'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE', 'APARTMENTS_MEDI', 'BASEMENTAREA_MEDI',
'YEARS_BEGINEXPLUATATION_MEDI', 'YEARS_BUILD_MEDI', 'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_MEDI', 'FLOORSMAX_MEDI',
'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI', 'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MEDI',
'NONLIVINGAREA_MEDI', 'TOTALAREA_MODE', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE',
'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', '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',
'CB_PCT_ACTIVE_LOANS', 'CB_SEC_LOAN', 'CB_UNSEC_LOAN', 'CB_AVG_DAYS_CREDIT_ENDDATE', 'CB_AVG_DAYS_ENDDATE_FACT',
'CB_TOTAL_CNT_CREDIT_PROLONG', 'CB_AMT_CREDIT_SUM_OVERDUE', 'CB_PCT_DEBT', 'CB_AMT_CREDIT_SUM_DEBT', 'CB_MONTHS_0_AVG',
'CB_MONTHS_1_AVG', 'CB_MONTHS_2_AVG', 'CB_MONTHS_3_AVG', 'CB_MONTHS_4_AVG', 'CB_MONTHS_5_AVG', 'PREV_AVG_DPD_DAYS',
'PREV_AVG_CNT_DPD', 'PREV_SUM_AMT_ANNUITY', 'PREV_SUM_AMT_CREDIT', 'PREV_DEBT_TO_CREDIT_RATIO', 'CC_PCT_CASH_DRAWINGS',
'CC_AVG_AMT_DRAWINGS', 'POS_CNT_INSTALLMENTS_FUTURE', 'INS_AVG_LATE_PAYMENT', 'INS_AVG_LESS_PAYMENT']
```

Converting binary variables to categories:

```python
#Converting flags and rating to categorical
cat_convert = application_df_v1.select_dtypes(include='integer').drop(columns=['SK_ID_CURR', 'CNT_CHILDREN', 'DAYS_BIRTH',
'DAYS_EMPLOYED', 'DAYS_ID_PUBLISH','CB_TOTAL_LOANS', 'CB_MIN_DAYS_CREDIT', 'CNT_ACCOUNT_TYPES', 'CNT_ACTIVE_LOANS']).columns
application_df_v1[cat_convert] = application_df_v1[cat_convert].astype(object)

#Verifying changed data types
dtypes = application_df_v1.dtypes.unique()

for i in range(0,len(dtypes)):
    sel_cols = list(application_df_v1.select_dtypes(include=dtypes[i]))
    print('application_df_v1 - ',{dtypes[i]}, '- Number of columns are' , len(sel_cols),':')
    print(sel_cols)
    print("_"*50)
```

Python

3.2 Data Quality

3.2.1 Removing irrelevant columns

Columns HOUR_APPR_PROCESS_START and WEEKDAY_APPR_PROCESS_START are about the day and time the applicant applied in the previous applications. Removing this since this information should not affect or predict the probability of a client's default. The same reason applies for DAYS_REGISTRATION since it only says about the number of days before the application did client change his registration. On the other hand, organization_type was removed because the values has a wide range of variation that is hard to consolidate.

```
#Removing HOUR_APPR_PROCESS_START, WEEKDAY_APPR_PROCESS_START, ORGANIZATION_TYPE

remove = ['HOUR_APPR_PROCESS_START', 'WEEKDAY_APPR_PROCESS_START','ORGANIZATION_TYPE', 'DAYS_REGISTRATION']
application_df_v1 = application_df_v1.drop(columns=remove)
len(application_df_v1.columns)
```

Python

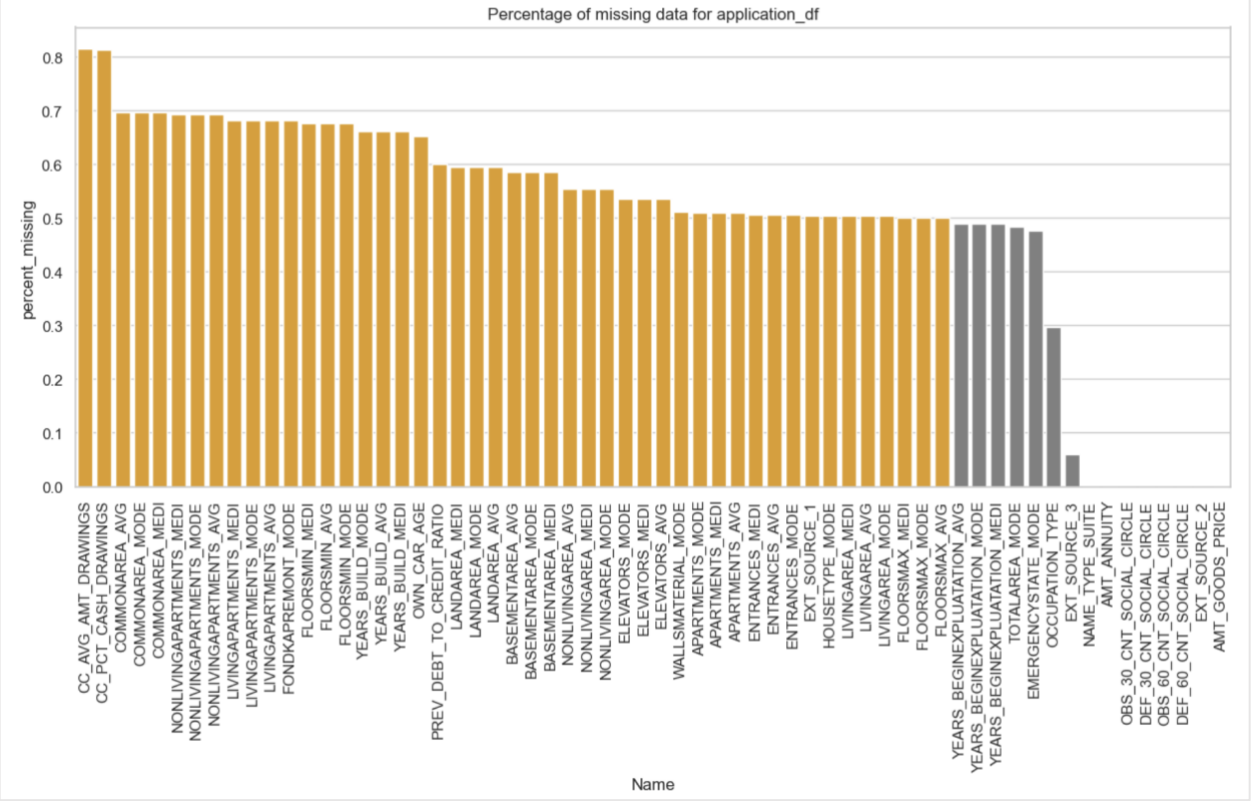
```
147
```

3.2.2 Missing values

There are a total of 62 columns with missing values with 47 of these more than 50% of the data.

Table 7 Visualization of missing values

```
Total number of columns with more than 50% missing values is: 47
```



Note that CC_AVG_AMT_DRAWINGS and CC_PCT_CASH_DRAWINGS are both from the credit card table. Having this missing is expected for those clients that has no cash withdrawals therefore we set this to zero.

EXT_SOURCE_1 is the normalized credit score from an external source and is missing at random. Therefore, this is set to median, and a missing flag was created. Same was done for EXT_SOURCE_2 and EXT_SOURCE_3.

```
application_df_cleaned['M_EXT_SOURCE_1'] = application_df_cleaned['EXT_SOURCE_1'].isnull().astype(int)
application_df_cleaned['M_EXT_SOURCE_2'] = application_df_cleaned['EXT_SOURCE_2'].isnull().astype(int)
application_df_cleaned['M_EXT_SOURCE_3'] = application_df_cleaned['EXT_SOURCE_3'].isnull().astype(int)
```

On the other hand, OWN_CAR_AGE is valid to be missing because this relates to those without car (FLAG_OWN_CAR = N); therefore, this column will be set to zero and missing flag is not needed.

```
#Impute NaN for OWN_CAR_AGE. These are set to 0 since these are for clients without car (OWN_CAR_FLAG = N)
application_df_cleaned['OWN_CAR_AGE'] = application_df_cleaned['OWN_CAR_AGE'].fillna(0)
application_df_cleaned['OWN_CAR_AGE'].isnull().mean()
```

```
0.0
```

For the others, most are related to the normalized building information of where the client lives and seems to provide redundant information like NONLIVINGAPARTMENTS_MEDI, NONLIVINGAPARTMENTS_MODE and NONLIVINGAPARTMENTS_AVG.

Numerical columns with missing data are imputed using median while for object columns we used mode.

```
missing_columns = missing_app[missing_app['percent_missing'] > 0].index

#Impute other numerical values with median and categorical with mode
for i in missing_columns:
    if application_df_cleaned[i].dtypes == 'float64':
        application_df_cleaned[i] = application_df_cleaned[i].fillna(application_df_cleaned[i].median())
    elif application_df_cleaned[i].dtypes == 'O':
        application_df_cleaned[i] = application_df_cleaned[i].fillna(application_df_cleaned[i].mode()[0])
print('Columns with missing values:', application_df_cleaned.isnull().mean().sum())
```

```
Columns with missing values: 0.0
```

3.2.3 Valid Values

In the Appendix Figure 38 , it can be seen that DAYS_* variables (DAYS_BIRTH, DAYS_ID_PUBLISH, DAYS_REGISTRATION, DAYS_EMPLOYED, DAYS_LAST_PHONE_CHANGE) have negative values because these are events that happened before the application date. Each column is checked if it is better to convert to positive value or remain negative.

The DAYS_BIRTH is the client's age at the time of application. This is converted to AGE variable by dividing by 365 days and multiplying by negative 1.

```
#Converting DAYS_BIRTH to AGE by dividing by 365 and multiplying by (-1), round by nearest integer
#Removing DAYS_BIRTH column
application_df_cleaned['AGE'] = round((application_df_cleaned['DAYS_BIRTH']/365)*(-1))
application_df_cleaned = application_df_cleaned.drop(columns=('DAYS_BIRTH'))
```

The DAYS_EMPLOYED variable is the number of days before the application the person started current employment. The only value above zero is a very large number 365243 which means infinity. Further checking showed that this is a scenario for clients with NAME_INCOME_TYPE as Pensioners or Unemployed. These high values were set to zero and converted to positive numbers.

```
#DAYS_EMPLOYED – This column denotes the number of days before the application the person started their current employment.
#The large number is existing for those with NAME_INCOME_TYPE either Pensioner or Unemployed, which is expected.
# Set the value to zero.
application_df_cleaned['DAYS_EMPLOYED'] = application_df_cleaned['DAYS_EMPLOYED'].replace([365243], 0)
```

Python

```
#Changing days_employed to positive
application_df_cleaned['DAYS_EMPLOYED'] = application_df_cleaned['DAYS_EMPLOYED'].apply(lambda x: abs(x))
```

Python

Variables DAYS_ID_PUBLISH, DAYS_LAST_PHONE_CHANGE, CB_MIN_DAYS_CREDIT all have values below zero.

```python
#Max value is 0. Changing DAYS_ID_PUBLISH to positive
application_df_cleaned['DAYS_ID_PUBLISH'] = application_df_cleaned['DAYS_ID_PUBLISH'].apply(lambda x: abs(x))
```

Python

```python
#Max value is 0. Changing DAYS_LAST_PHONE_CHANGE to positive
application_df_cleaned['DAYS_LAST_PHONE_CHANGE'] = application_df_cleaned['DAYS_LAST_PHONE_CHANGE'].apply(lambda x: abs(x))
```

Python

```python
#Max value is 0. Changing CB_MIN_DAYS_CREDIT to positive
application_df_cleaned['CB_MIN_DAYS_CREDIT'] = application_df_cleaned['CB_MIN_DAYS_CREDIT'].apply(lambda x: abs(x))
```

Python

3.2.4 Correlation

Variables that are highly correlated with other variables creates inaccurate predictions and unstable model is used both in the modeling. Therefore, we must check the multicollinearity among the variables that are higher than 60% and remove the ones that either have a lower correlation to the target or have more correlation with other variables.

3.2.4.1 Numerical variables

Correlation for numerical variables can be observed with the use of a correlation heatmap. The results are shown in the Appendix Figure 39. In summary, the table below shows the numeric variables with at least 60% correlation.

Table 8 Numerical variables - Heatmap correlation summary results

Variables	Correlation	Comments / Step(s) performed
Building info for AVG, MEDI and MODE are highly correlated Example: APARTMENTS_AVG, BASEMENT_AVG – 0.6 APARTMENTS_AVG, ELEVATORS_AVG – 0.8 APARTMENTS_AVG, TOTAL_AREA_MODE – 0.9	0.6 to 1	Retained only the relevant *MEDI (Median) variables.
CNT_CHILDREN , CNT_FAM_MEMBERS	0.9	Removed CNT_CHILDREN
AMT_CREDIT, AMT_CREDIT_TO_ANNUITY	0.7	Removed AMT_CREDIT

AMT_CREDIT, AMT_ANNUITY	0.8	Removed AMT_CREDIT
AMT_CREDIT, AMT_GOODS_PRICE	1	Removed both
AMT_ANNUITY, AMT_GOODS_PRICE	0.8	Removed AMT_GOODS_PRICE
AMT_GOODS_PRICE , AMT_CREDIT_TO_ANNUITY	0.6	Removed AMT_GOODS_PRICE
OBS_30_CNT_SOCIAL_CIRCLE, OBS_60_CNT_SOCIAL_CIRCLE	1	Removed OBS_60_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE, DEF_60_CNT_SOCIAL_CIRCLE	0.9	Removed DEF_60_CNT_SOCIAL_CIRCLE
CB_UNSEC_LOAN, CB_TOTAL_LOANS	1	Removed CB_TOTAL_LOANS
CB_MIN_DAYS_CREDIT, CB_AVG_DAYS_ENDDATE_FACT	0.8	Removed CB_AVG_DAYS_ENDDATE_FACT
CB_MONTHS_2_AVG, CB_MONTHS_3_AVG	0.6	Removed CB_MONTHS_3_AVG
CB_MONTHS_3_AVG, CB_MONTHS_4_AVG	0.7	Removed CB_MONTHS_3_AVG
CNT_ACCOUNT_TYPES, CNT_ACTIVE_LOANS	0.9	Removed CNT_ACCOUNT_TYPES
PREV_AVG_DPD_DAYS, PREV_AVG_CNT_DPD	0.8	Removed PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY, PREV_SUM_AMT_CREDIT	0.9	Removed PREV_SUM_AMT_ANNUITY
CB_AMT_CREDIT_SUM_DEBT, DEBT_TO_INCOME_RATIO	0.8	Removed CB_AMT_CREDIT_SUM_DEBT

3.2.4.2 Categorical variables

The chi-square test of independence is a statistical method used to determine the correlation between two categorical variables. The variables above a p-value of 0.5 are considered insignificant and removed from the data. A total of 20 variables are selected which has below 0.5 p-value while 29 variables are removed.

Table 9 Categorical variables - chi-square summary results

Feature	Chi-Square	P-value
TARGET	33020.452	0.000
NAME_INCOME_TYPE	156.665	0.000
REGION_RATING_CLIENT_W_CITY	140.123	0.000
REGION_RATING_CLIENT	134.549	0.000

OCCUPATION_TYPE	145.007	0.000
REG_CITY_NOT_WORK_CITY	93.247	0.000
FLAG_EMP_PHONE	83.366	0.000
REG_CITY_NOT_LIVE_CITY	83.299	0.000
CODE_GENDER	75.845	0.000
NAME_EDUCATION_TYPE	73.946	0.000
FLAG_DOCUMENT_3	49.252	0.000
FLAG_DOCUMENT_6	41.548	0.000
LIVE_CITY_NOT_WORK_CITY	33.039	0.000
FLAG_OWN_CAR	23.810	0.000
NAME_HOUSING_TYPE	35.393	0.000
FLAG_WORK_PHONE	22.200	0.000
FLAG_PHONE	18.323	0.000
NAME_FAMILY_STATUS	14.589	0.006
FONDKAPREMONT_MODE	9.977	0.019
FLAG_DOCUMENT_8	4.700	0.030
NAME_CONTRACT_TYPE	3.759	0.053
FLAG_EMAIL	3.351	0.067
FLAG_DOCUMENT_13	1.843	0.175
FLAG_OWN_REALTY	1.483	0.223
FLAG_DOCUMENT_9	1.478	0.224
WALLSMATERIAL_MODE	7.924	0.244
REG_REGION_NOT_WORK_REGION	1.223	0.269
FLAG_DOCUMENT_20	1.136	0.286
FLAG_DOCUMENT_14	1.132	0.287
FLAG_CONT_MOBILE	1.104	0.293
FLAG_DOCUMENT_16	0.720	0.396
FLAG_DOCUMENT_18	0.292	0.589
REG_REGION_NOT_LIVE_REGION	0.251	0.617
FLAG_DOCUMENT_5	0.172	0.679
NAME_TYPE_SUITE	3.653	0.724
LIVE_REGION_NOT_WORK_REGION	0.109	0.741
EMERGENCYSTATE_MODE	0.106	0.744
HOUSETYPE_MODE	0.537	0.765
FLAG_DOCUMENT_19	0.040	0.842
FLAG_DOCUMENT_17	0.000	1.000
FLAG_DOCUMENT_15	0.000	1.000
FLAG_DOCUMENT_12	0.000	1.000
FLAG_DOCUMENT_2	0.000	1.000

FLAG_DOCUMENT_11	0.000	1.000
FLAG_DOCUMENT_10	0.000	1.000
FLAG_DOCUMENT_7	0.000	1.000
FLAG_DOCUMENT_4	0.000	1.000
FLAG_MOBIL	0.000	1.000
FLAG_DOCUMENT_21	0.000	1.000

Also checked the correlation between each variable using dython library in python which calculates categorical variable correlation (Refer to Appendix Figure 40). From the results we removed one of the correlated variables:

Columns	Correlation	Comments/Step(s) Performed
NAME_INCOME_TYPE, FLAG_DOCUMENT_	0.6	Removed FLAG_DOCUMENT_6
REGION_RATING_CLIENT_W_CITY, REGION_RATING_	1	Removed REGION_RATING_CLIENT
FLAG_EMP_PHONE, FLAG_DOCUMENT_6	0.6	Removed FLAG_DOCUMENT_6
REG_CITY_NOT_WORK_CITY, LIVE_CITY_NOT_WORK_CITY	0.8	Removed LIVE_CITY_NOT_WORK_CITY
FLAG_DOCUMENT_3, FLAG_DOCUMENT_6	0.6	Removed FLAG_DOCUMENT_6
FLAG_DOCUMENT_3, FLAG_DOCUMENT_8	0.6	Removed FLAG_DOCUMENT_8
FONDKAPREMONT_MODE	multiple	Removed, a lot of correlations with other fields, unknown meaning
TOTAL AREA MODE	multiple	Removed, has a lot of correlations with other variables

3.2.4.3 Removing variables based on correlation.

Removing variables based on Chi-square:

[DATA_CLEANING] Remove not significant variables

```
#columns to be removed based on Chi-square
remove_categories = ['NAME_CONTRACT_TYPE', 'FLAG_OWN_REALTY', 'NAME_TYPE_SUITE', 'FLAG_MOBIL', 'FLAG_CONT_MOBILE',
'FLAG_EMAIL', 'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION', 'HOUSETYPE_MODE',
'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_7',
'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']

#remove from application_df_cleaned
application_df_cleaned = application_df_cleaned.drop(columns=remove_categories)
application_df_cleaned.shape
```

✓ 3.7s Python

```
(33032, 113)
```

Removing variables based on heatmap and dython correlation:

```
#columns to be removed based on correlation
remove_col = ['CNT_CHILDREN', 'AMT_CREDIT', 'AMT_GOODS_PRICE', 'REGION_RATING_CLIENT', 'LIVE_CITY_NOT_WORK_CITY',
    'APARTMENTS_AVG', 'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BUILD_AVG', 'COMMONAREA_AVG', 'ELEVATORS_AV
    'FLOORSMAX_AVG', 'FLOORSMIN_AVG', 'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG', 'NONLIVINGAPARTMENTS_AVG',
    'APARTMENTS_MODE', 'BASEMENTAREA_MODE', 'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE', 'COMMONAREA_MODE', 'ELEVATOI
    'FLOORSMAX_MODE', 'FLOORSMIN_MODE', 'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE', 'LIVINGAREA_MODE', 'NONLIVINGAPARTMENTS_M
    'TOTALAREA_MODE', 'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE','CB_TOTAL_LOANS','CB_AVG_DAYS_ENDDATE_FACT','CI
    'CNT_ACCOUNT_TYPES','PREV_AVG_CNT_DPD','PREV_SUM_AMT_ANNUITY','CB_AMT_CREDIT_SUM_DEBT','FLAG_DOCUMENT_6','FLAG_DOCUMENT_
len(remove_col)
```

✓ 1.8s Python

```
47
```

```
#columns for main table application_df_cleaned are now 64 from 110.
application_df_cleaned = application_df_cleaned.drop(remove_col, axis=1)
application_df_cleaned.shape
```

✓ 0.6s Python

```
(33032, 67)
```

After checking the correlation, the remaining features are now 67.

3.2.5 Recode categories.

3.2.5.1 Checking unique values of categories that need to be grouped together

Having too many unique values of a categorical variable can cause complexity in the data and overfitting due to the curse of dimensionality. In order to reduce this risk, we will group categories when possible.

Feature	Unique values and distribution	Step(s) performed
TARGET	0 0.904244 1 0.095756	N/A
CODE_GENDER	F 0.671 M 0.329	N/A
FLAG_OWN_CAR	N 0.652 Y 0.348	N/A
NAME_INCOME_TYPE	Working 0.523 Commercial associate 0.239 Pensioner 0.166 State servant 0.071 Student 0.000061	Grouped by Employed and Unemployed
NAME_EDUCATION_TYPE	Secondary / secondary special 0.725 Higher education 0.231 Incomplete higher 0.033 Lower secondary 0.011 Academic degree 0.0005	Grouped by Secondary/secondary special,
NAME_FAMILY_STATUS	Married 0.687 Single / not married 0.120 Civil marriage 0.090 Separated 0.061 Widow 0.042	Grouped by Married (Married, Civil marriage), Single, Widow_Separated
NAME_HOUSING_TYPE	House / apartment 0.895 With parents 0.041 Municipal apartment 0.039 Rented apartment 0.014 Office apartment 0.008 Co-op apartment 0.003	N/A
FLAG_PHONE	0: 0.732 1: 0.268	N/A
OCCUPATION_TYPE	Laborers 0.483 Sales staff 0.111 Core staff 0.086 Managers 0.075 Drivers 0.064 High skill tech staff 0.035239 Accountants 0.0332 Medicine staff 0.0274 Security staff 0.02 Cooking staff 0.02	Grouped using ISCO-08 occupation levels / major groups: 1_Manager: Managers; 2_Professionals: Accountants, High skill tech staff, IT staff; 3_Techinicians and Associate Professionals: Realty

	Cleaning staff 0.016 Private service staff 0.0097 Low-skill Laborers 0.006 Secretaries 0.005 Waiters/barmen staff 0.004 Realty agents 0.003 HR staff 0.0018 IT staff 0.001	agents, medicine staff, sales staff; 4_Clerical Support Workers: HR Staff, Secretary; 5_Service and Sales Workers: Waiters/barmen staff, Cleaning staff, Core staff, Security staff, private service staff; 8_Plant and Machine Operators, and Assemblers : Drivers; 9_Elementary occupation - laborers, low skill laborer, Cooking staff;
REGION_RATING_CLIENT_W_CITY	2 : 0.755 3 : 0.137 1 : 0.107	N/A
REG_CITY_NOT_LIVE_CITY	0 : 0.925 1 : 0.075	N/A
REG_CITY_NOT_WORK_CITY	0 : 0.768 1 : 0.232	N/A
FLAG_DOCUMENT_3	1 : 0.803 0 : 0.197	N/A

3.2.5.2 Recoding categories

Regrouped NAME_INCOME_TYPE by combining Unemployed, Pensioner, and Student to "Unemployed_Pensioner" category while Maternity leave, Commercial associate, State servant and Working are categorized as "Employed".

```
#Recode for NAME_INCOME_TYPE
application_df_cleaned['NAME_INCOME_TYPE'] = application_df_cleaned['NAME_INCOME_TYPE'].replace(['Unemployed', 'Pensioner','Student'], 'Unemployed_Pensioner')
application_df_cleaned['NAME_INCOME_TYPE'] = application_df_cleaned['NAME_INCOME_TYPE'].replace(['Maternity leave', 'Commercial associate', 'State servant','Working'], '
application_df_cleaned.groupby('NAME_INCOME_TYPE')['TARGET'].count()/len(application_df_cleaned)*100
```

✓ 0.3s Python

```
NAME_INCOME_TYPE
Employed               83.358561
Unemployed_Pensioner   16.641439
Name: TARGET, dtype: float64
```

NAME_EDUCATION_TYPE is grouped based on the highest educational attainment where Academic degree is combined to "Higher_education" and Incomplete higher and Secondary/Secondary special as "Secondary".

```
application_df_cleaned['NAME_EDUCATION_TYPE'] = application_df_cleaned['NAME_EDUCATION_TYPE'].replace(['Academic degree', 'Incomplete higher'],
                                                ['Higher education','Secondary / secondary special'])
application_df_cleaned['NAME_EDUCATION_TYPE'] = application_df_cleaned['NAME_EDUCATION_TYPE'].replace(['Secondary / secondary special'], 'Secondary')
application_df_cleaned.groupby('NAME_EDUCATION_TYPE')['TARGET'].count()/len(application_df_cleaned)*100
```

✓ 0.3s

```
NAME_EDUCATION_TYPE
Higher education    23.104868
Lower secondary      1.089852
Secondary           75.805280
Name: TARGET, dtype: float64
```

NAME_FAMILY_STATUS is recoded by grouping civil marriage and married as "Married" while separated and widow is grouped as "Widow_Separated".

```
application_df_cleaned['NAME_FAMILY_STATUS'] = application_df_cleaned['NAME_FAMILY_STATUS'].replace(['Civil marriage'], 'Married')
application_df_cleaned['NAME_FAMILY_STATUS'] = application_df_cleaned['NAME_FAMILY_STATUS'].replace(['Separated', 'Widow'], 'Widow_Separated')
application_df_cleaned.groupby('NAME_FAMILY_STATUS')['TARGET'].count()/len(application_df_cleaned)*100
```

✓ 0.1s

```
NAME_FAMILY_STATUS
Married                 77.767014
Single / not married    11.970211
Widow_Separated         10.262775
Name: TARGET, dtype: float64
```

OCCUPATION_TYPE was grouped based on the International Standards Classification of Occupations (ISCO).

```
#Combining occupation type based on ISCO-08
application_df_cleaned['OCCUPATION_TYPE'] = application_df_cleaned['OCCUPATION_TYPE'].replace(['Managers'], '1_Manager')
application_df_cleaned['OCCUPATION_TYPE'] = application_df_cleaned['OCCUPATION_TYPE'].replace(['Accountants', 'High skill tech staff', 'IT staff'],
'2_Professionals')
application_df_cleaned['OCCUPATION_TYPE'] = application_df_cleaned['OCCUPATION_TYPE'].replace(['Realty agents','Medicine staff', 'Sales staff'],
'3_Tech_Assoc')
application_df_cleaned['OCCUPATION_TYPE'] = application_df_cleaned['OCCUPATION_TYPE'].replace(['HR staff','Secretaries'], '4_Clerical')
application_df_cleaned['OCCUPATION_TYPE'] = application_df_cleaned['OCCUPATION_TYPE'].replace(['Waiters/barmen staff', 'Cleaning staff',
'Core staff', 'Security staff', 'Private service staff'], '5_Service_Sales')
application_df_cleaned['OCCUPATION_TYPE'] = application_df_cleaned['OCCUPATION_TYPE'].replace(['Drivers'], '8_Operators_Assemblers')
application_df_cleaned['OCCUPATION_TYPE'] = application_df_cleaned['OCCUPATION_TYPE'].replace(['Laborers', 'Low-skill Laborers',
'Cooking staff'], '9_Elementary_Occupation')

application_df_cleaned.groupby('OCCUPATION_TYPE')['TARGET'].count()/len(application_df_cleaned)*100
```

✓ 0.2s

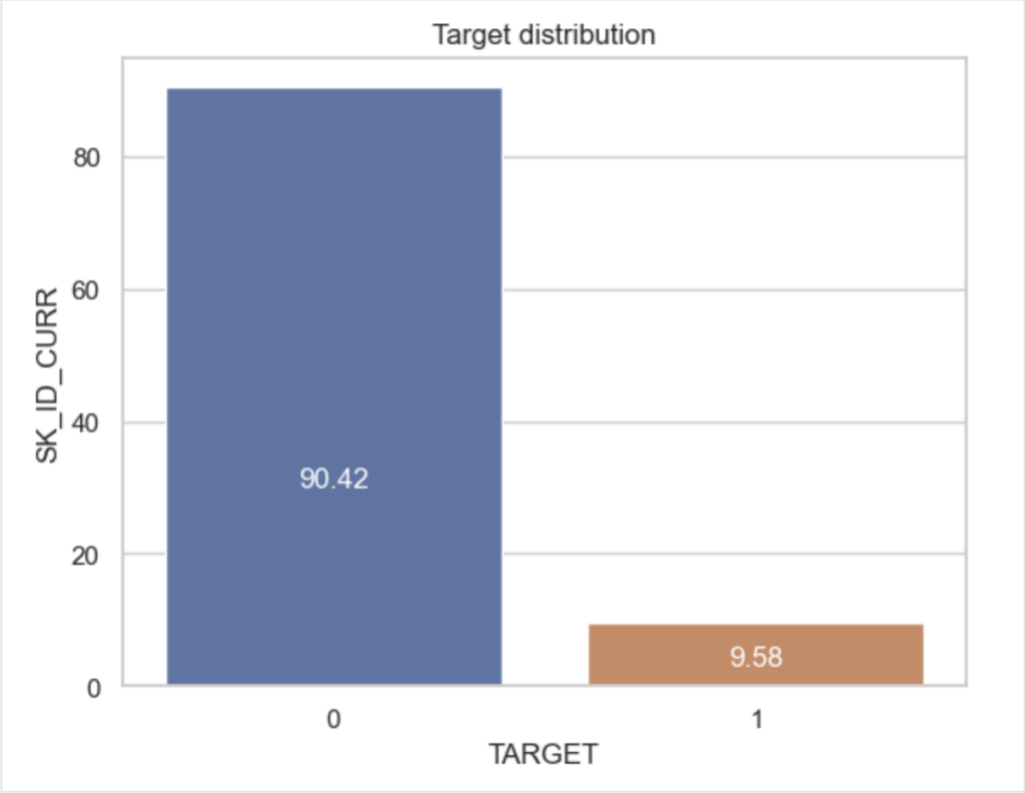
```
OCCUPATION_TYPE
1_Manager                7.453379
2_Professionals          6.969000
3_Tech_Assoc            14.125696
4_Clerical               0.641802
5_Service_Sales         13.502059
8_Operators_Assemblers   6.424074
9_Elementary_Occupation  50.883991
Name: TARGET, dtype: float64
```

3.2.6 Distribution of target variable

This dataset has a binary target variable with 1 meaning the client has defaulted the loan payment and 0 when the client did not default. Based on the distribution, we have a highly imbalanced

dataset with 90.4% of clients who did not default against 9.58% of clients who defaulted. An imbalanced dataset is common to this type of business problem; however, it has potential issues when used directly in modelling such as biased model performance, inaccurate results, and overfitting. This will be countered by using oversampling/undersampling technique before creating the models.

Figure 4 Target distribution - imbalanced data



3.2.7 Univariate Analysis – Distribution and Outliers

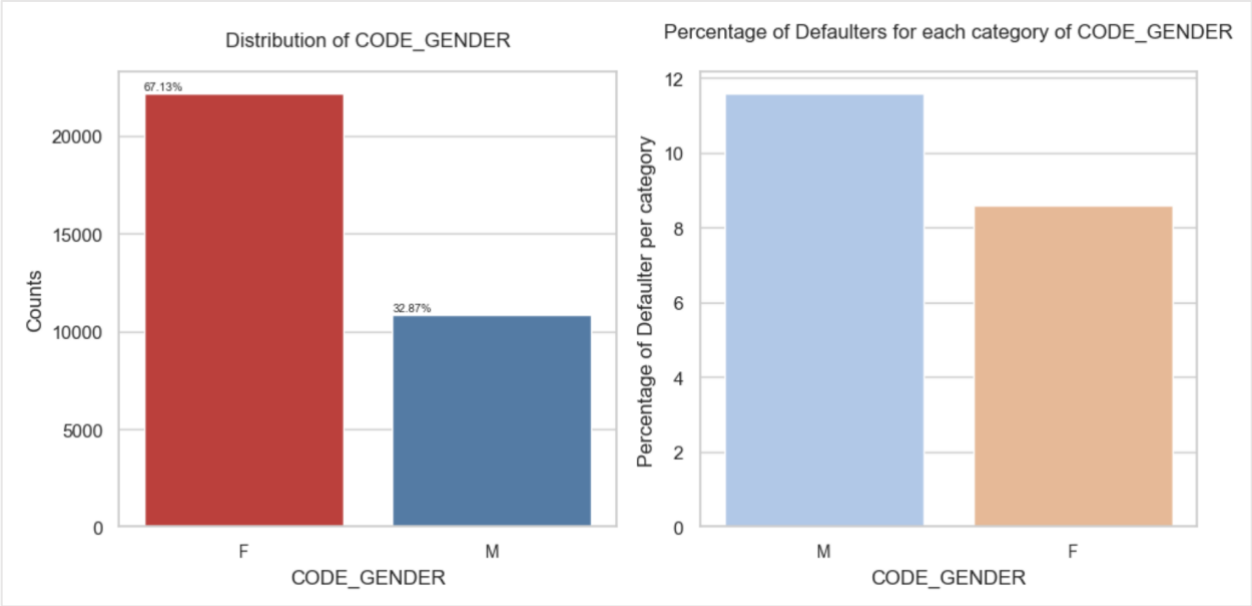
Examining the distribution of data and outliers are important steps in exploratory data analysis because it provides insights on the data patterns and necessary steps for data cleaning and transformation.

3.2.7.1 Categorical variables

CODE_GENDER

This has two categories with a 67:33 ratio of female to male. The male group is seen to have higher percentage of default amounting to 11%.

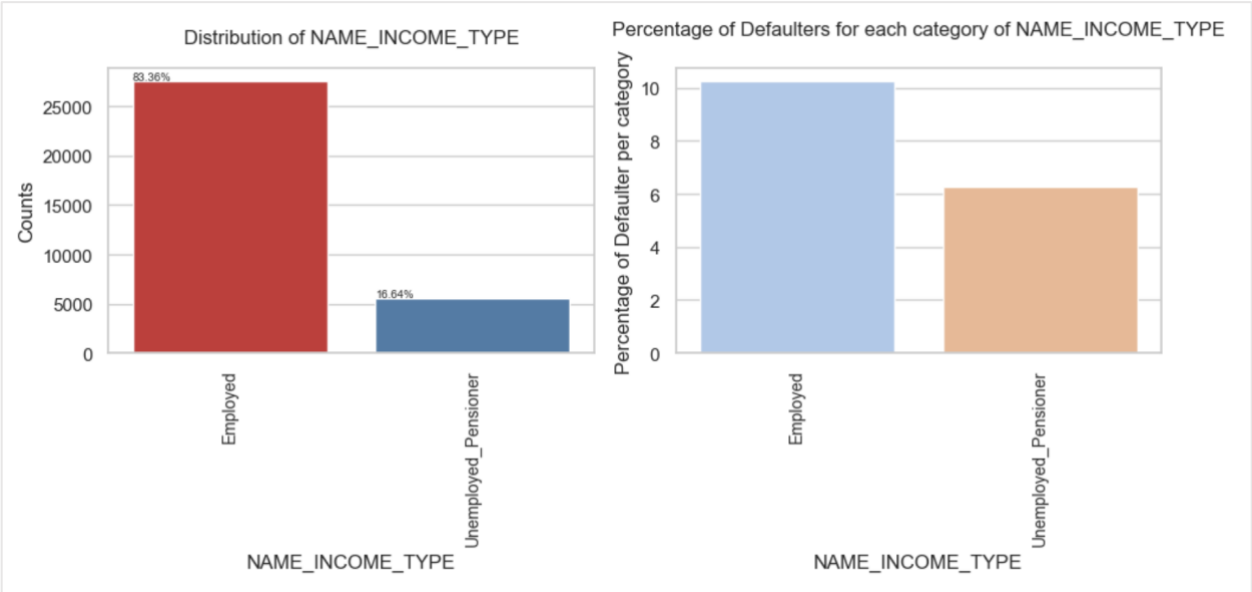
Figure 5 Univariate analysis - CODE_GENDER



NAME_INCOME_TYPE

There's a big gap between the percentage of Employed and Unemployed_Pensioner that has applied for a loan where 83% of the dataset are Employed. It can also noticed that there are 11% and 7% of default for each category.

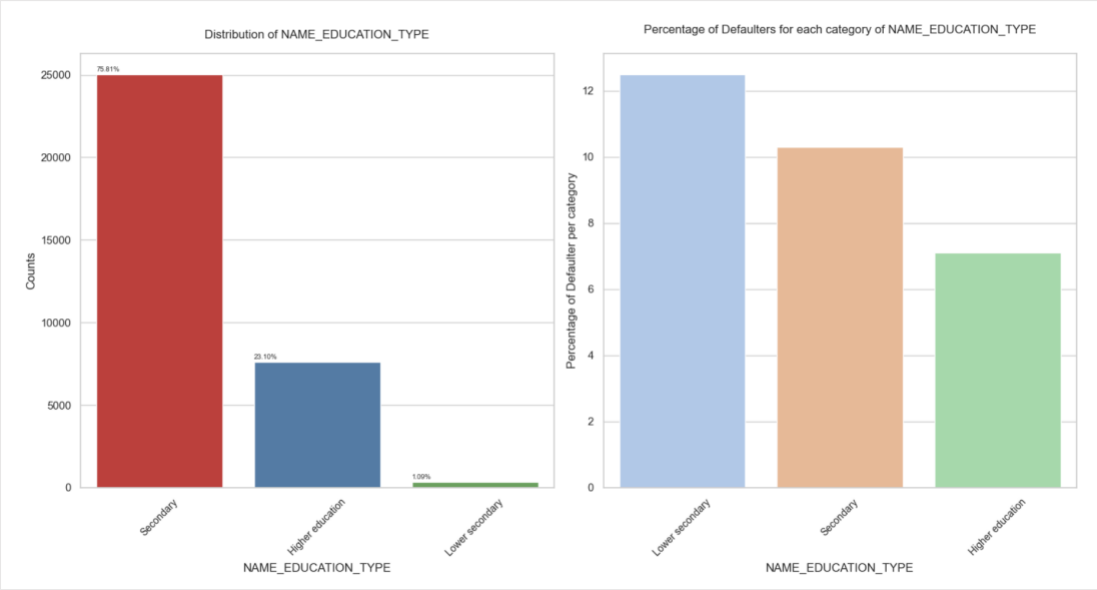
Figure 6 Univariate analysis - NAME_INCOME_TYPE



NAME_EDUCATION_TYPE

Around 71% of the loan applicants have completed Secondary / Secondary special education level, followed by Higher Education with 24%. Highest default rate is seen for lower secondary followed by secondary education level.

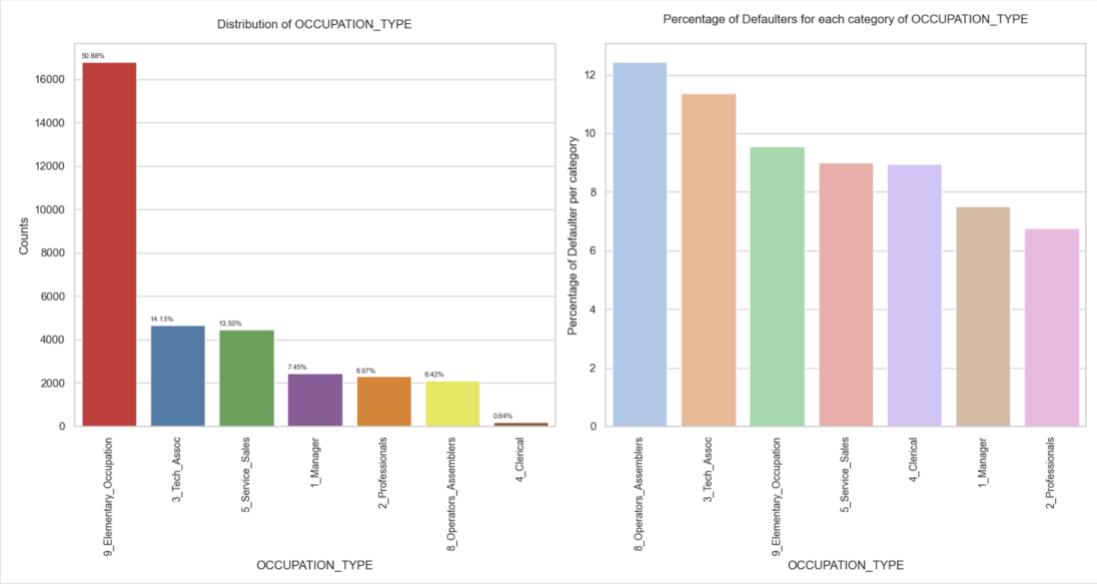
Figure 7 Univariate analysis - NAME_EDUCATION_TYPE



OCCUPATION_TYPE

Most of the applicants have occupation type Laborers (26%), followed by sales staff and core staff. Based on the default percentage, low-skill laborers are highest followed by Drivers, and Waiters/Barmen staff.

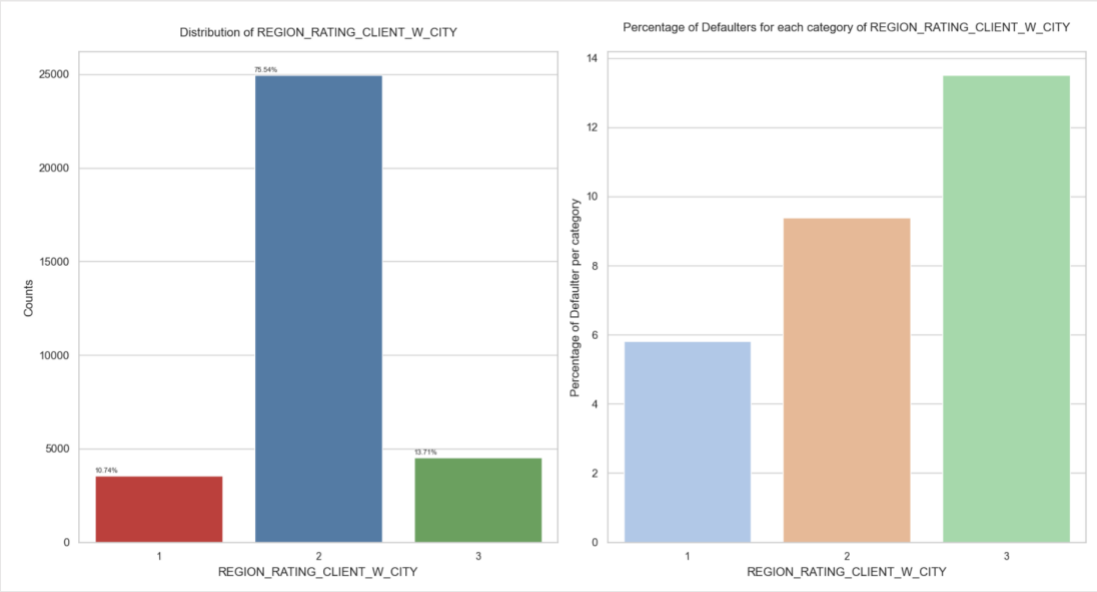
Figure 8 Univariate analysis - Occupation type



REGION_RATING_CLIENT_W_CITY

This is the rating of the region including city where the client lives where the rating 2 has the highest percentage amounting to 75%, followed by rating 3 with 14% and rating 1 with 11%. On the other hand, rating 3 has the highest percentage of default amounting to 13%, followed by rating 2 at 9% and rating 1 with almost 6%.

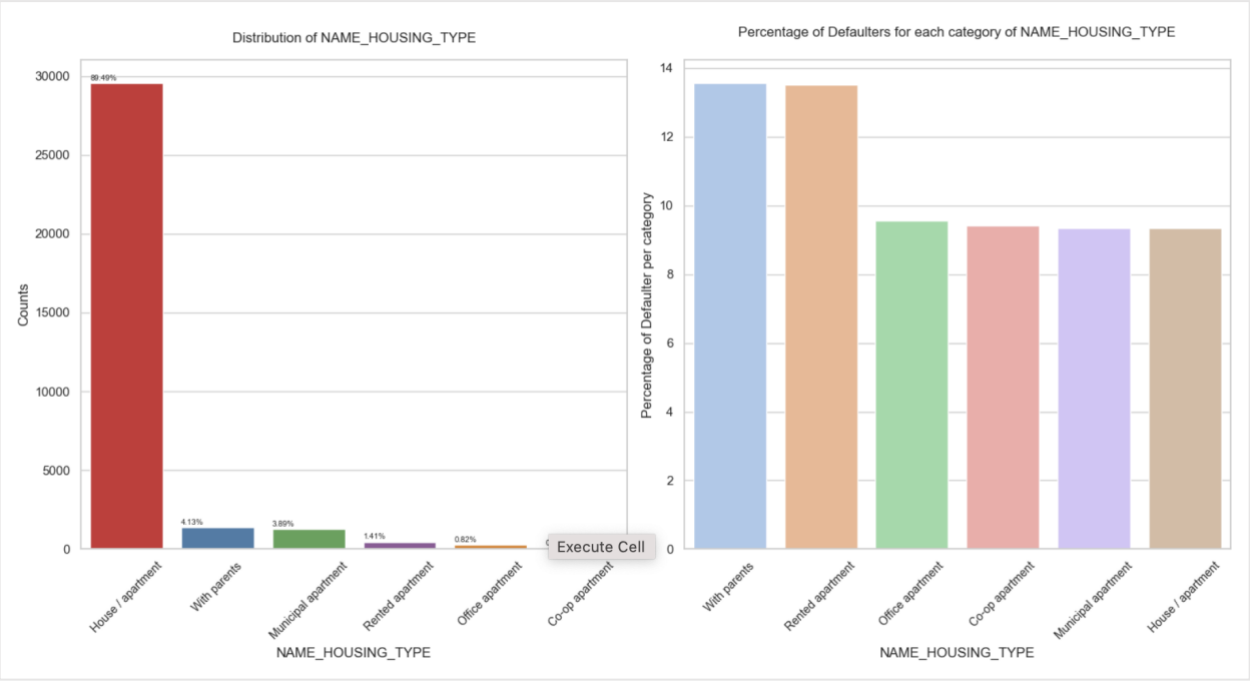
Figure 9 Univariate analysis - REGION_RATING_CLIENT



NAME_HOUSING_TYPE

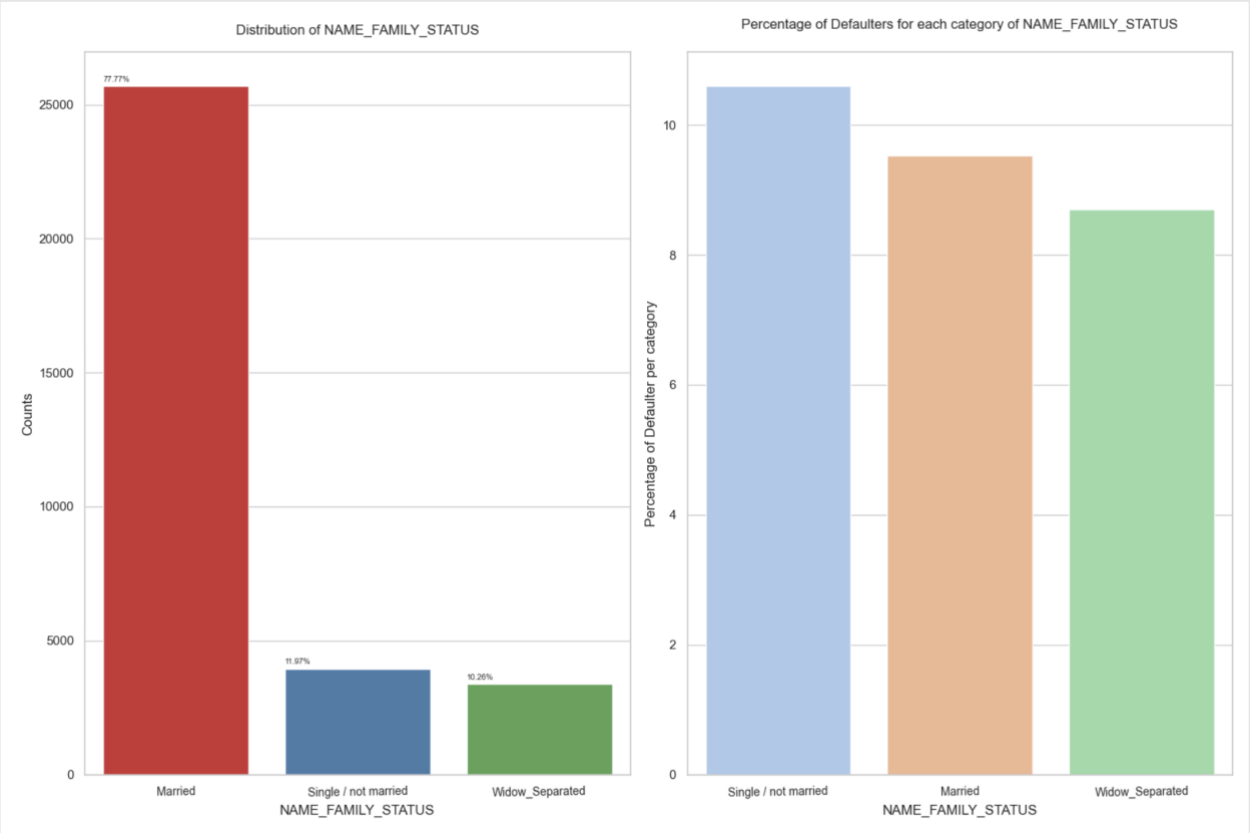
This column describes the housing situation of the client. Majority of the clients, 89%, live in a house / apartment but has the lowest percentage of default which is 9%. The next highest are the clients who live with their parents and living in a municipal apartment which are both have 4%, however, these groups also have the highest percentage of default at 13% each.

Figure 10 Univariate analysis - NAME_HOUSING_TYPE



NAME_FAMILY_STATUS

Almost 78% of the applicants are married and only 12% and 10% are single and widowed/separated. In terms of default, the single group has the highest percentage amounting to 11%.

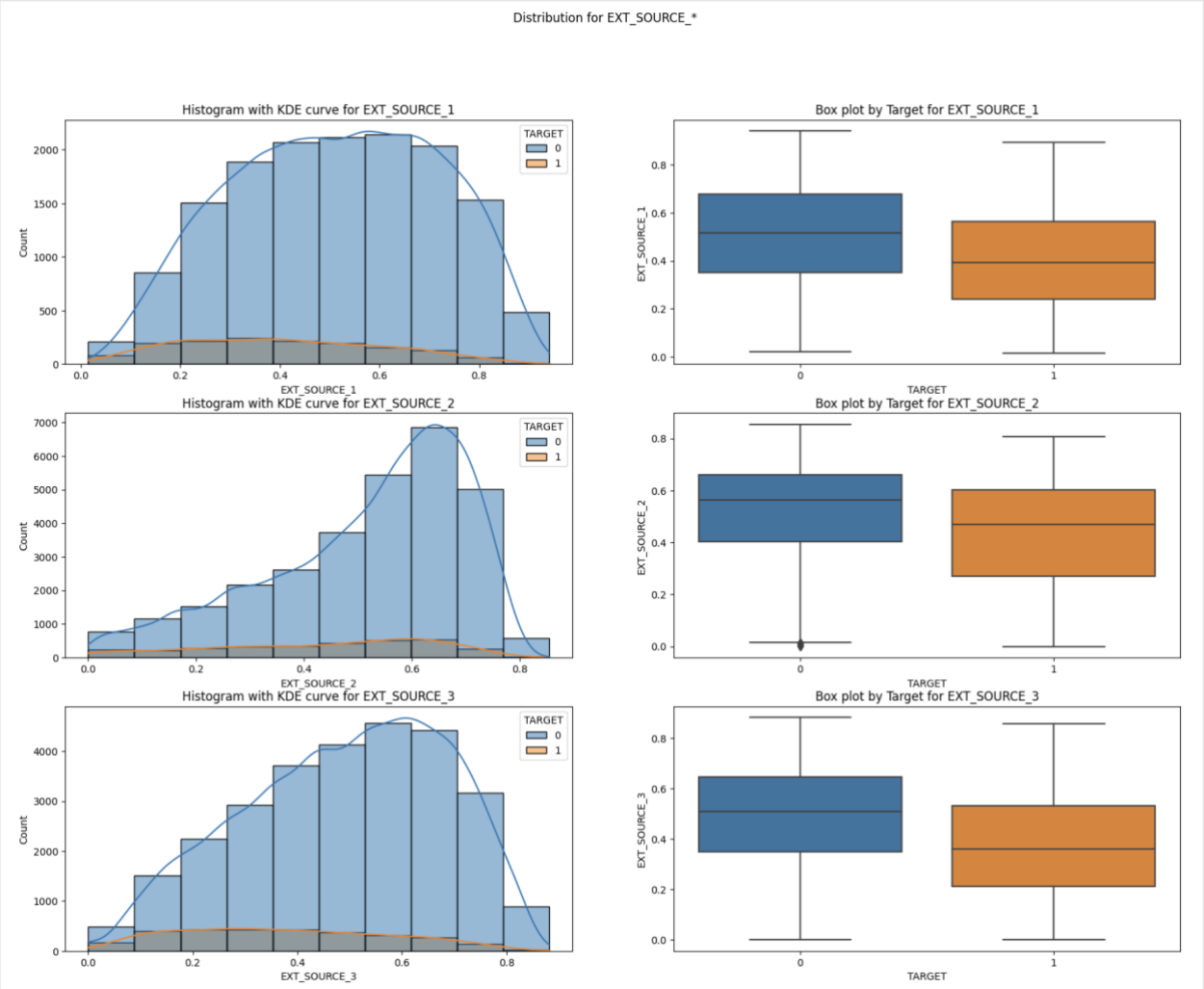


3.2.7.2 Numerical variables

EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3

The variables EXT_SOURCE_1, EXT_SOURCE_2, and EXT_SOURCE_3 pertains to the normalized credit score from external sources. From Figure 4 below, we can see that EXT_SOURCE_1 values are normally distributed and there are no outliers. On the other hand, EXT_SOURCE_2 and EXT_SOURCE_3 are slightly skewed to the left. It can also be noticed that the lower the score, the higher the number of defaulted clients.

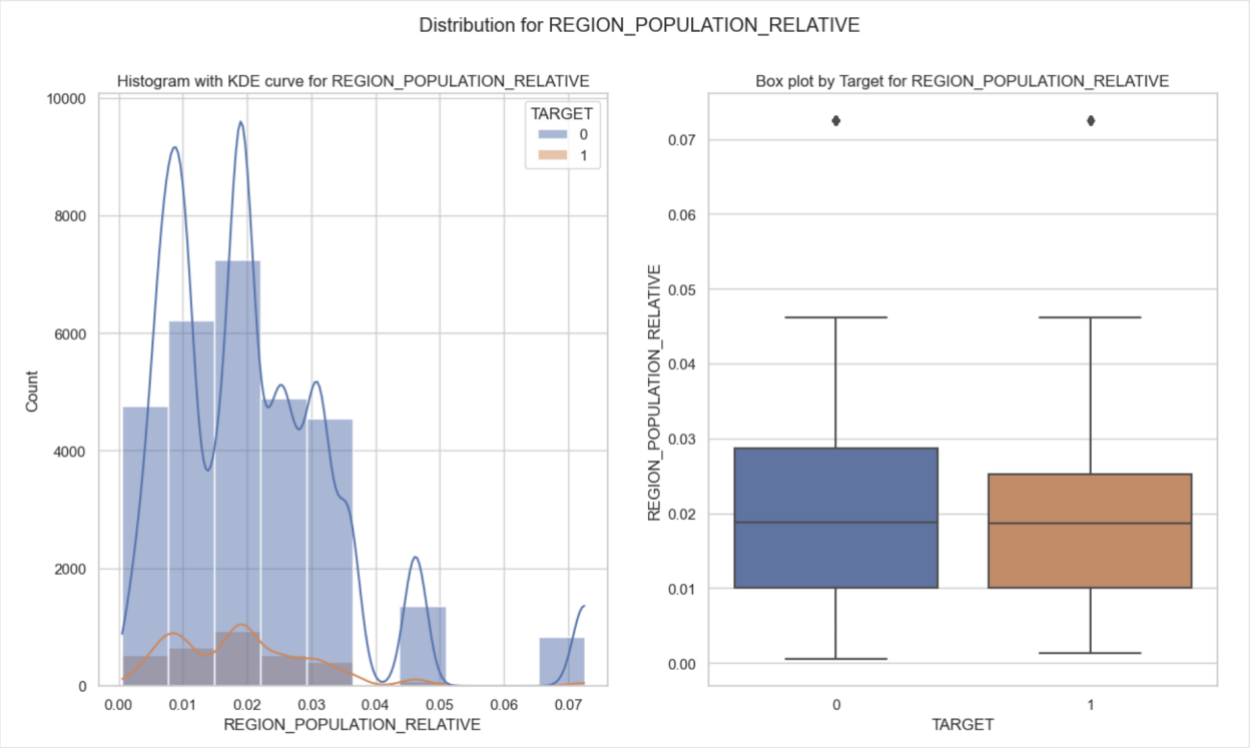
*Figure 11 Univariate analysis - EXT_SOURCE_**



REGION_POPULATION_RELATIVE

This is a normalized population of the region where the client lives where the higher number means the client lives in a more populated region. From Figure 5, it can be noticed that this column is a multimodal distribution where multiple peaks are seen. There are also some outlier both for target 1 and target 0.

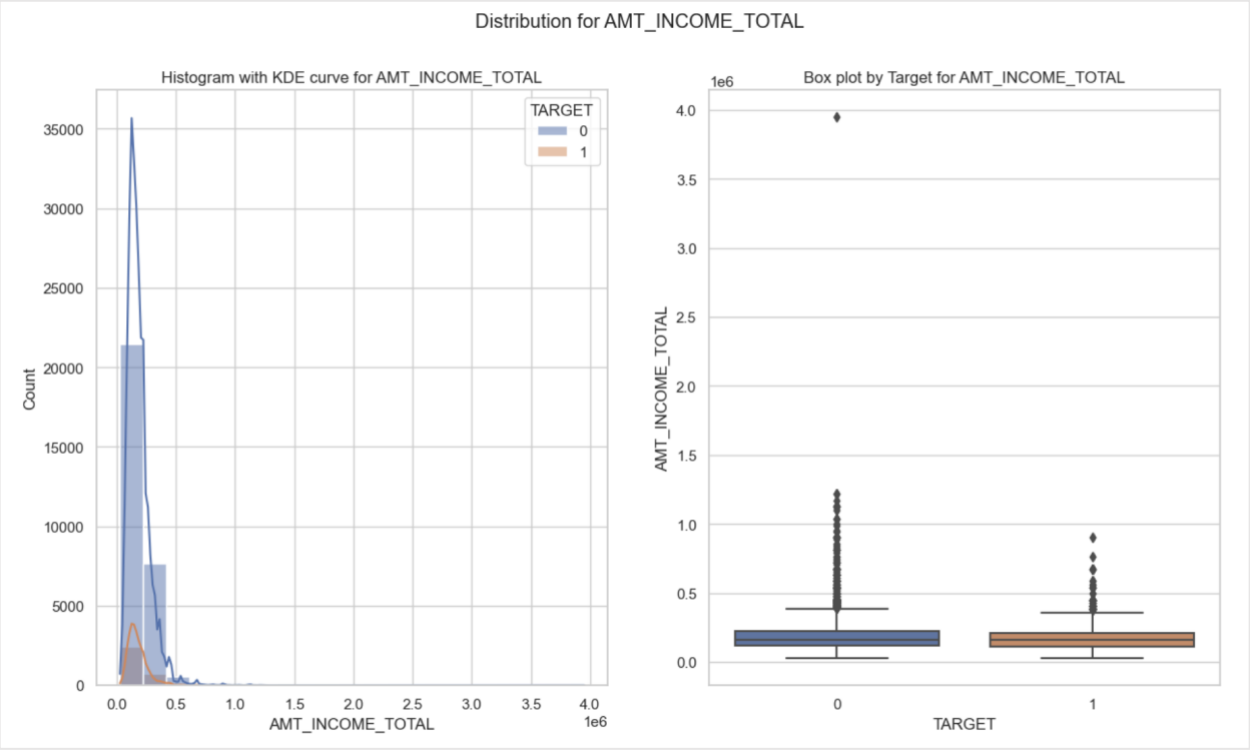
Figure 12 Univariate Analysis - REGION_POPULATION_RELATIVE



AMT_INCOME_TOTAL

This is the monthly income amount of the client. From Figure 5 it can be noticed that it is right skewed where several outliers were seen outside the upper limit.

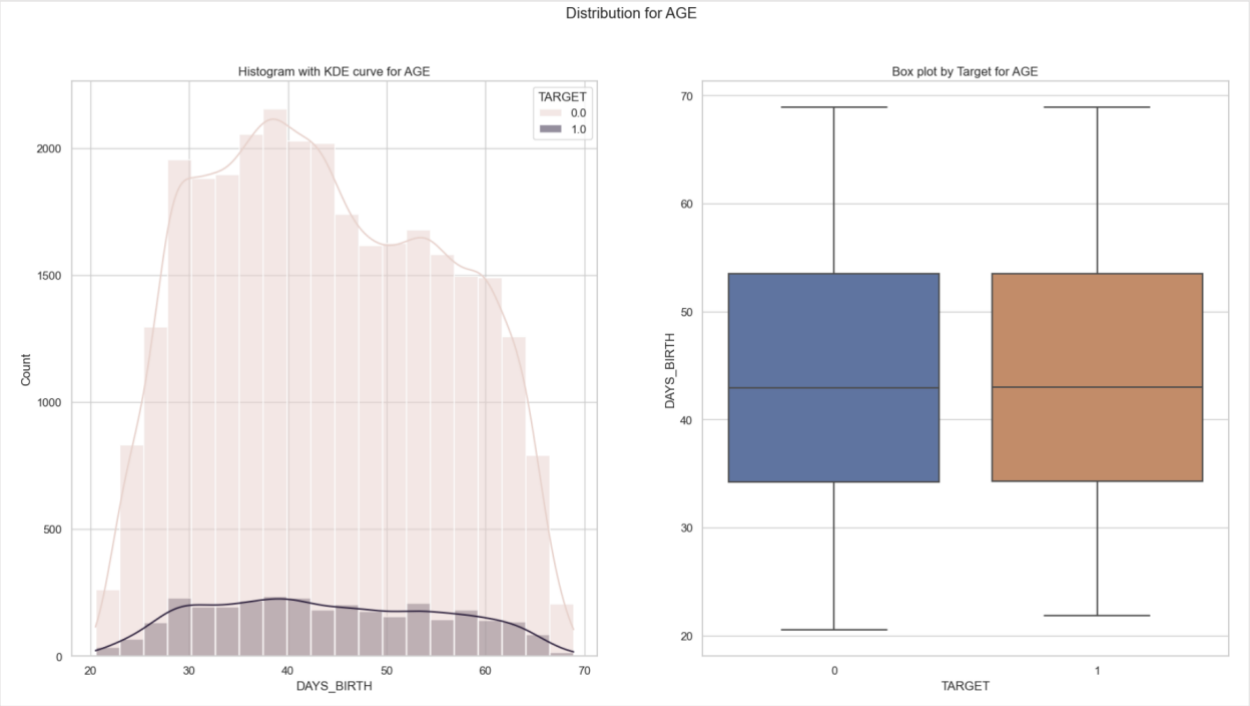
Figure 13 Univariate analysis - AMT_INCOME_TOTAL



DAYS_BIRTH

This feature is converted to AGE and a positive number. It is normally distributed and has no outliers.

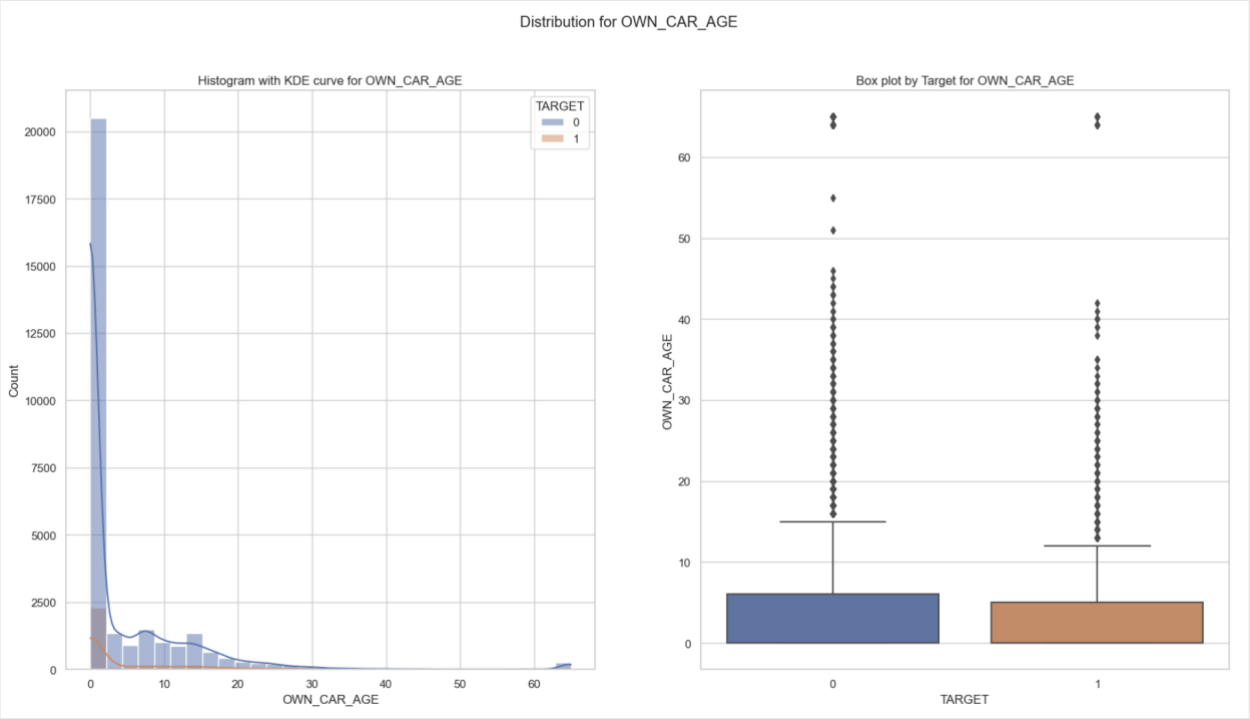
Figure 14 Univariate analysis - DAYS_BIRTH or AGE



OWN_CAR_AGE

This feature is skewed to the right and has multiple outliers outside the upper limit of 15. The maximum age is 65.

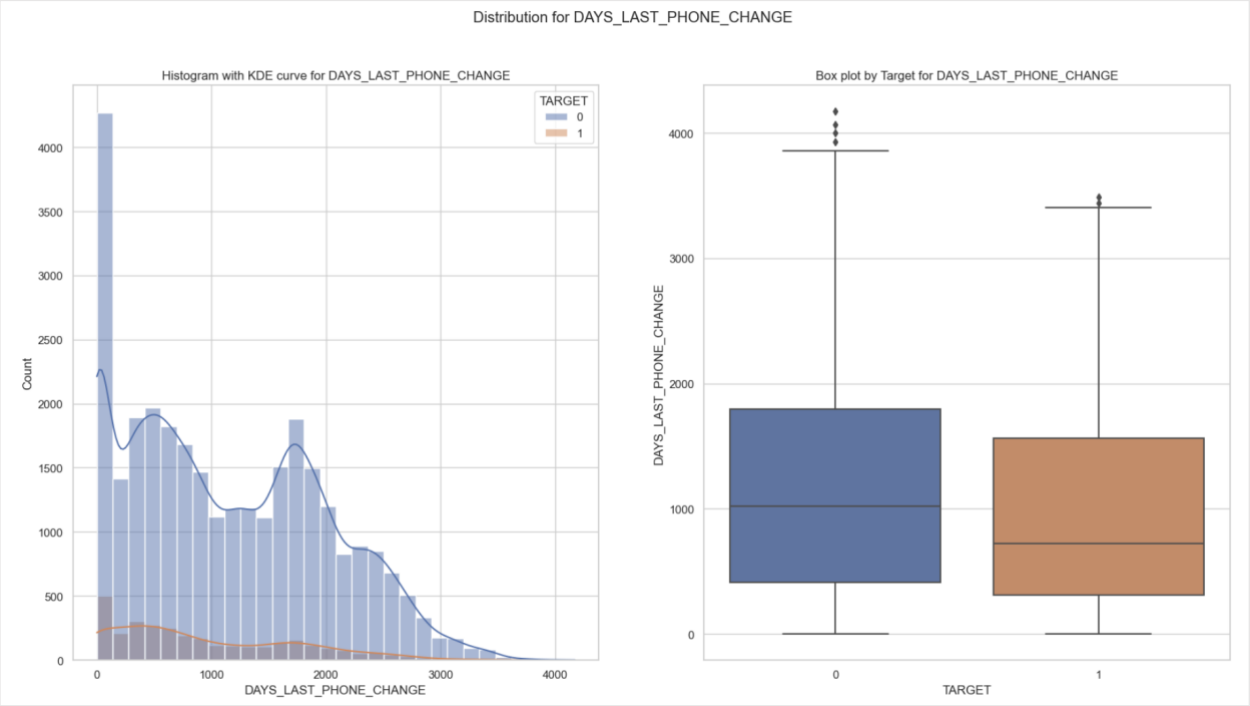
Figure 15 Univariate analysis - OWN_CAR_AGE



DAYS_LAST_PHONE_CHANGE

DAYS_LAST_PHONE_CHANGE shows the number of days before the application did the client change their phone number. This feature is skewed to the right and looks like has multiple modes. There are six outliers as seen in the box plot.

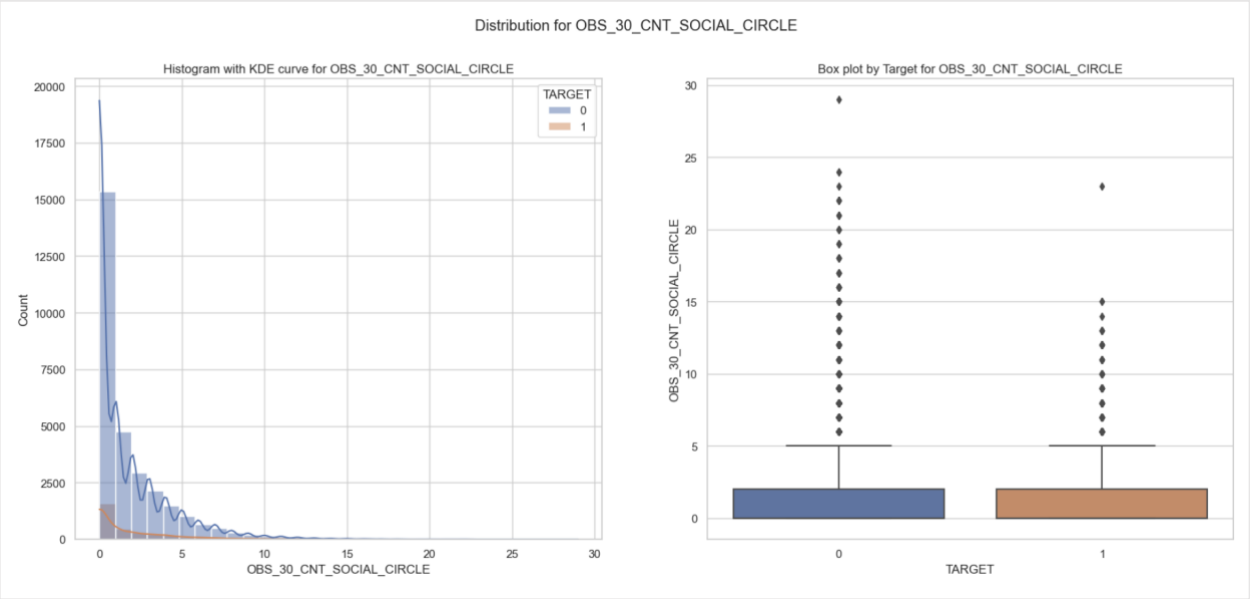
Figure 16 Univariate analysis - DAYS_LAST_PHONE_CHANGE



OBS_30_CNT_SOCIAL_CIRCLE

This feature talks about how many observations of the client's social surroundings have observable 30 DPD (days past due). This is right skewed and multiple outliers exists for count of 5.

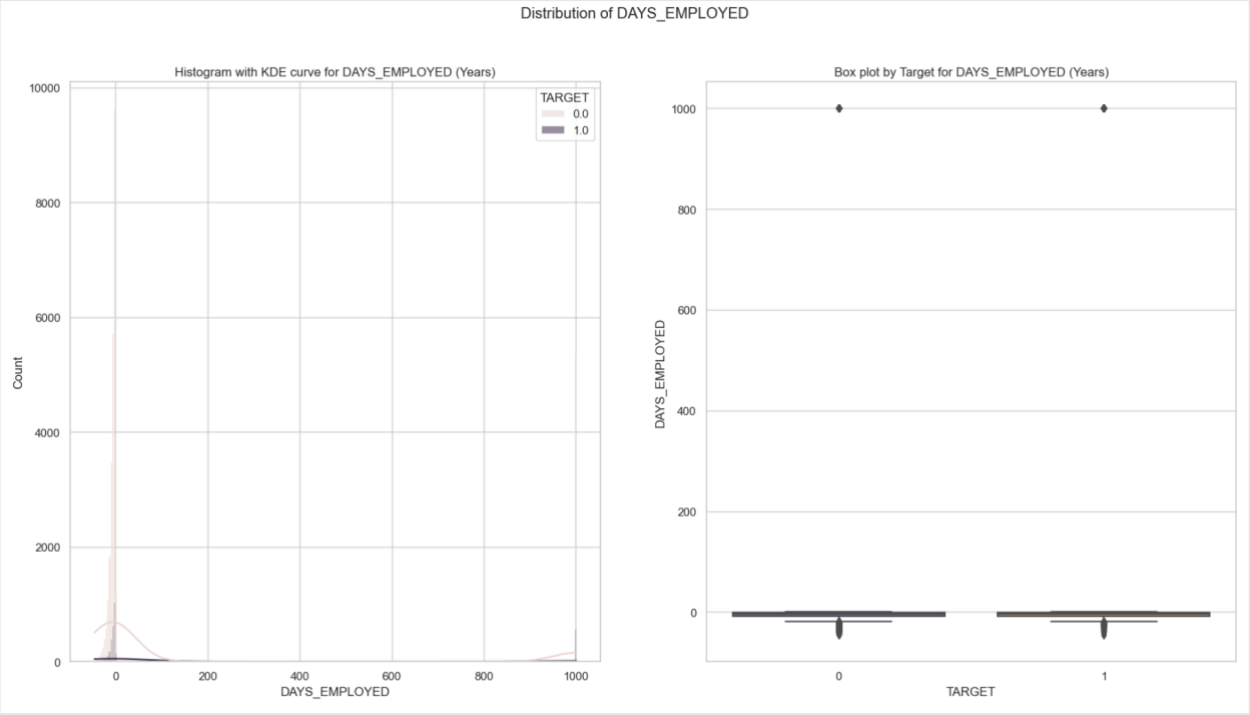
Figure 17 Univariate analysis - OBS_30_CNT_SOCIAL_CIRCLE



DAYS_EMPLOYED (converted to YEARS)

This feature is the number of days the client has been working converted to years. There is a very large number 1000 yrs which needs to be corrected. However, looks like there's no distribution for this column. This will be converted to bins.

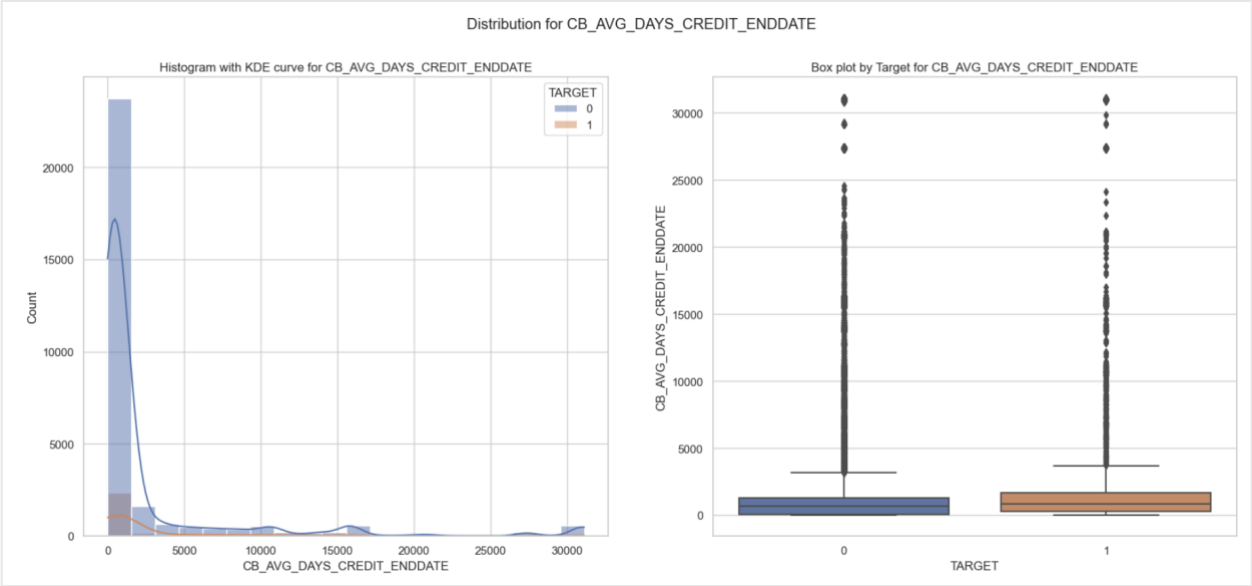
Figure 18 Univariate analysis - DAYS_EMPLOYED



CB_AVG_DAYS_CREDIT_ENDDATE

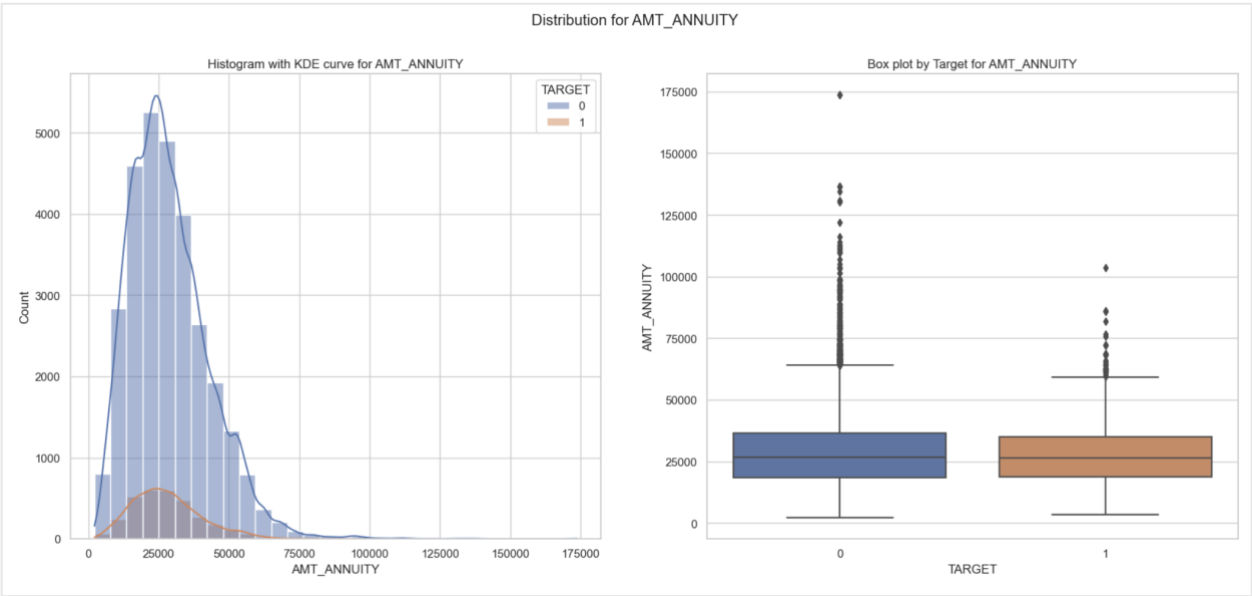
The figure below shows the average remaining duration (in days) of credit bureau credit at the time of the client's application. Most of the data are zero, and has a uniform distribution between 250-1000. There are also multiple outliers seen. This column is better to be converted to bins.

Figure 19 Univariate analysis - CB_AVG_DAYS_CREDIT_ENDDATE



AMT_ANNUITY

This column is skewed to the left and a lot of outliers were seen outside the upper fence.



3.2.8 Create bins

For numerical variables, binning is one way to reduce the dimensionality and deal with outliers especially the variables with too many values and sparse distribution.

CNT_FAM_MEMBERS

```
#creating bins for CNT_FAM_MEMBERS
application_df_cleaned2['CNT_FAM_MEMBERS'] = pd.cut(application_df_cleaned2['CNT_FAM_MEMBERS'],
                                                    bins=[-np.inf, 2, 4, np.inf], labels=["1-2", "3-4", "5+"])
application_df_cleaned2.groupby('CNT_FAM_MEMBERS')['TARGET'].count()
```

✓ 0.1s

```
CNT_FAM_MEMBERS
1-2    23898
3-4     8671
5+       463
```

OWN_CAR_AGE

```
#creating bins for OWN_CAR_AGE
application_df_cleaned2['OWN_CAR_AGE'] = pd.cut(application_df_cleaned2['OWN_CAR_AGE'],
    bins=[-np.inf, 5, 10, 15, 20, 25, 30, np.inf], labels=["5 and under", "6-10", "11-15", "16-20", "21-25","26-30", "31+"])
application_df_cleaned2.groupby('OWN_CAR_AGE')['TARGET'].count()
```

✓ 0.1s Python

```
OWN_CAR_AGE
5 and under    24684
6-10            3205
11-15           2448
16-20           1332
21-25            634
26-30            296
31+              433
```

AMT_REQ_CREDIT_BUREAU_MON

```
#creating bins for AMT_REQ_CREDIT_BUREAU_MON
application_df_cleaned2['AMT_REQ_CREDIT_BUREAU_MON'] = pd.cut(application_df_cleaned2['AMT_REQ_CREDIT_BUREAU_MON'],
                                                    bins=[-np.inf,0, 2, np.inf], labels=["0","1-2", "3+"])
application_df_cleaned2.groupby('AMT_REQ_CREDIT_BUREAU_MON')['TARGET'].count()
```

✓ 0.1s Python

```
AMT_REQ_CREDIT_BUREAU_MON
0      25705
1-2     6902
3+       425
```

AMT_REQ_CREDIT_BUREAU_DAY

```python
#creating bins for AMT_REQ_CREDIT_BUREAU_DAY
application_df_cleaned2['AMT_REQ_CREDIT_BUREAU_DAY'] = pd.cut(application_df_cleaned2['AMT_REQ_CREDIT_BUREAU_DAY'],
                                                              bins=[-np.inf,0, 2, np.inf], labels=["0","1-2", "3+"])
application_df_cleaned2.groupby('AMT_REQ_CREDIT_BUREAU_DAY')['TARGET'].count()
```

✓ 0.1s Python

```
AMT_REQ_CREDIT_BUREAU_DAY
0      32801
1-2      216
3+        15
```

CB_SEC_LOANS

```python
#creating bins for CB_SEC_LOAN
application_df_cleaned2['CB_SEC_LOAN'] = pd.cut(application_df_cleaned2['CB_SEC_LOAN'],
                                                bins=[-np.inf,0, 2, np.inf], labels=["0","1-2", "3+"])
application_df_cleaned2.groupby('CB_SEC_LOAN')['TARGET'].count()
```

✓ 0.7s Python

```
CB_SEC_LOAN
0      29150
1-2     3741
3+       141
```

CB_UNSEC_LOANS

```python
#creating bins for CB_UNSEC_LOAN
application_df_cleaned2['CB_UNSEC_LOAN'] = pd.cut(application_df_cleaned2['CB_UNSEC_LOAN'],
                                                  bins=[-np.inf, 5, 10, 15, 20, 25, 30, np.inf], labels=["0-5", "6-10", "11-15", "16-20", "21-25","26-30", "31+"])
application_df_cleaned2.groupby('CB_UNSEC_LOAN')['TARGET'].count()
```

✓ 0.1s Python

```
CB_UNSEC_LOAN
0-5      19830
6-10      9243
11-15     2776
16-20      862
21-25      219
26-30       64
31+         38
```

CB_AMT_CREDIT_SUM_OVERDUE

```
#creating bins for CB_AMT_CREDIT_SUM_OVERDUE
application_df_cleaned2['CB_AMT_CREDIT_SUM_OVERDUE'] = pd.cut(application_df_cleaned2['CB_AMT_CREDIT_SUM_OVERDUE'],
bins=[-np.inf,0,25000,50000,75000, 100000,np.inf], labels=["no_overdue", "below 25k","26-50k","51-75k","76-100k", "100+"])
application_df_cleaned2.groupby('CB_AMT_CREDIT_SUM_OVERDUE')['TARGET'].count()
```

✓ 0.2s Python

```
CB_AMT_CREDIT_SUM_OVERDUE
no_overdue    32690
below 25k       322
26-50k            9
51-75k            6
76-100k           3
100+              2
```

INS_AVG_LATE_PAYMENT

```
#creating bins for INS_AVG_LATE_PAYMENT
application_df_cleaned2['INS_AVG_LATE_PAYMENT'] = pd.cut(application_df_cleaned2['INS_AVG_LATE_PAYMENT'],
        bins=[-np.inf,0,10,20,30,np.inf], labels=["zero","1-10days","11-20days","21-30days", "31+days"])
application_df_cleaned2.groupby('INS_AVG_LATE_PAYMENT')['TARGET'].count()
#application_df_final2[application_df_final2['INS_AVG_LATE_PAYMENT'] > 30]['INS_AVG_LATE_PAYMENT'].nunique()
```

✓ 0.2s Python

```
INS_AVG_LATE_PAYMENT
zero          9824
1-10days     21297
11-20days     1331
21-30days      384
31+days        196
```

POS_CNT_INSTALMENTS_FUTURE

```
#Creating bins for POS_CNT_INSTALLMENTS_FUTURE
application_df_cleaned2['POS_CNT_INSTALLMENTS_FUTURE'] = pd.cut(application_df_cleaned2['POS_CNT_INSTALLMENTS_FUTURE'],
bins=[-np.inf,0,15,30,45,60,75,np.inf], labels=["none","1-15","16-30","31-45", "46-60", "61-75", "76+"])
application_df_cleaned2.groupby('POS_CNT_INSTALLMENTS_FUTURE')['TARGET'].count()
#application_df_final2.groupby('POS_CNT_INSTALLMENTS_FUTURE')['TARGET'].count()
```

✓ 0.1s Python

```
POS_CNT_INSTALLMENTS_FUTURE
none      14099
1-15      14039
16-30      3018
31-45      1289
46-60       537
61-75        36
76+          14
```

CB_MONTHS_1_AVG

```
#Creating bins for CB_MONTHS_1_AVG
application_df_cleaned2['CB_MONTHS_1_AVG'] = pd.cut(application_df_cleaned2['CB_MONTHS_1_AVG'],
bins=[-np.inf,0,10,20,np.inf], labels=["none","below10x","11-20x","21xandup"])
application_df_cleaned2.groupby('CB_MONTHS_1_AVG')['TARGET'].count()
#application_df_final2[application_df_final2['CB_MONTHS_1_AVG'] > 30]['TARGET'].count()
#application_df_final2['CB_MONTHS_1_AVG'].describe()
```

✓ 0.1s

```
CB_MONTHS_1_AVG
none         20514
below10x     12468
11-20x          44
21xandup         6
```

CB_MONTHS_2_AVG

```
#Creating bins for CB_MONTHS_2_AVG
application_df_cleaned2['CB_MONTHS_2_AVG'] = application_df_cleaned2['CB_MONTHS_2_AVG'].astype(int)
application_df_cleaned2['CB_MONTHS_2_AVG'] = pd.cut(application_df_cleaned2['CB_MONTHS_2_AVG'],
                                                    bins=[-np.inf,0,1,2,np.inf], labels=["none","1x","2x","3xandup"])
application_df_cleaned2.groupby('CB_MONTHS_2_AVG')['TARGET'].count()
```

✓ 0.1s Python

```
CB_MONTHS_2_AVG
none        32577
1x            342
2x             80
3xandup        33
```

CB_MONTHS_4_AVG

```
#Creating bins for CB_MONTHS_4_AVG
application_df_cleaned2['CB_MONTHS_4_AVG'] = application_df_cleaned2['CB_MONTHS_4_AVG'].astype(int)
application_df_cleaned2['CB_MONTHS_4_AVG'] = pd.cut(application_df_cleaned2['CB_MONTHS_4_AVG'],
                                                    bins=[-np.inf,0,1,np.inf], labels=["none","1x","2xandup"])
application_df_cleaned2.groupby('CB_MONTHS_4_AVG')['TARGET'].count()
```

✓ 0.1s Python

```
CB_MONTHS_4_AVG
none        32914
1x             99
2xandup        19
```

CB_AVG_DAYS_CREDIT_ENDDATE

```
#Create bins for CB_AVG_DAYS_CREDIT_ENDDATE
application_df_cleaned2['CB_AVG_DAYS_CREDIT_ENDDATE'] = pd.cut(application_df_cleaned2['CB_AVG_DAYS_CREDIT_ENDDATE'],
bins=[-np.inf,0,90,180,365,730,np.inf], labels=["zero","below3months","3-6months","7-12months","1-2yrs","2yrs+"])
application_df_cleaned2['CB_AVG_DAYS_CREDIT_ENDDATE'].unique()
```

✓ 0.1s

```
['zero', '2yrs+', '1-2yrs', 'below3months', '3-6months', '7-12months']
Categories (6, object): ['zero' < 'below3months' < '3-6months' < '7-12months' < '1-2yrs' < '2yrs+']
```

DEBT_TO_INCOME_RATIO

```
#Set minimum value of DEBT_TO_INCOME_RATIO to 0
application_df_cleaned2.loc[application_df_cleaned2['DEBT_TO_INCOME_RATIO'] < 0, 'DEBT_TO_INCOME_RATIO'] = 0
```

✓ 0.1s Python

```
#Create bins for DEBT_TO_INCOME_RATIO
#application_df_final2['DEBT_TO_INCOME_RATIO'] = application_df_final2['DEBT_TO_INCOME_RATIO'].astype(int)
application_df_cleaned2['DEBT_TO_INCOME_RATIO'] = pd.cut(application_df_cleaned2['DEBT_TO_INCOME_RATIO'],
    bins=[-np.inf,0,20,40,60,80,np.inf], labels=["none","below20pct","21-40pct","41-60pct","61-80pct","81pct_andup"])
application_df_cleaned2.groupby('DEBT_TO_INCOME_RATIO')['TARGET'].count()
```

✓ 0.1s Python

```
DEBT_TO_INCOME_RATIO
none            9332
below20pct     22884
21-40pct         604
41-60pct         144
61-80pct          48
81pct_andup       20
```

YEARS_EMPLOYED

```
#Create bins for DAYS_EMPLOYED - converted to years
application_df_cleaned2['YEARS_EMPLOYED'] = (application_df_cleaned['DAYS_EMPLOYED']/365).astype(int)
application_df_cleaned2['YEARS_EMPLOYED'] = pd.cut(application_df_cleaned2['YEARS_EMPLOYED'],
        bins=[-np.inf,5,10,15,30,np.inf], labels=["below_5","6-10","11-15","16-30", "31+"])
application_df_cleaned2 = application_df_cleaned2.drop(columns=['DAYS_EMPLOYED'])
application_df_cleaned2.groupby('YEARS_EMPLOYED')['TARGET'].count()
```

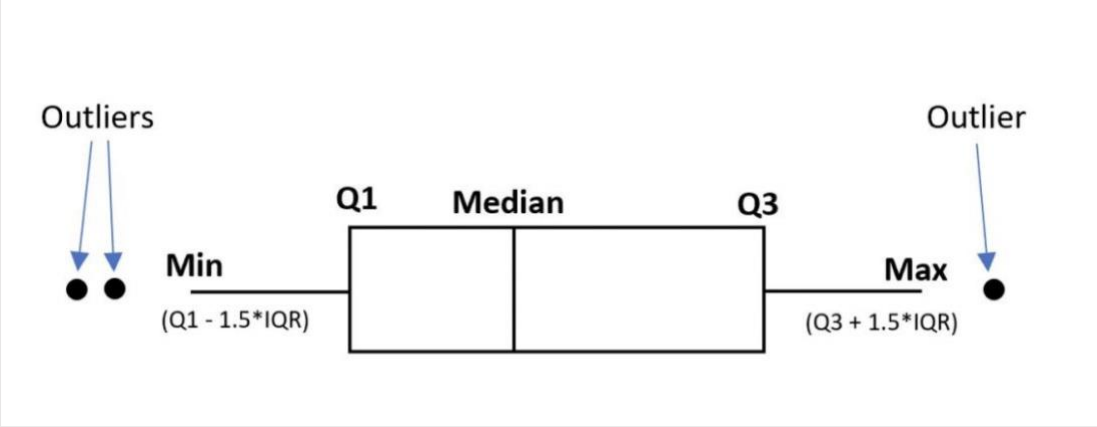
✓ 0.2s Python

```
YEARS_EMPLOYED
below_5    21059
6-10        6958
11-15       2846
16-30       1957
31+          212
```

3.2.9 Treat Outliers – Cap and Floor

Cap and floor are techniques that we used to treat variables with few outliers. Capping means anything above the upper limit is set to maximum value (Q3+1.5*IQR) and flooring means anything below lower limit will be set to the minimum value (Q1-1.5*IQR) as shown in the picture below.

Figure 20 Box and whisker plot



Cap and floor were done for the following columns:

['AMT_INCOME_TOTAL','REGION_POPULATION_RELATIVE','OBS_30_CNT_SOCIAL_CIRCLE',
'DAYS_LAST_PHONE_CHANGE', 'CB_MONTHS_0_AVG', 'PREV_DEBT_TO_CREDIT_RATIO']

3.2.10 Skewness

Same to imbalanced data and outliers, having a skewed distribution of the data poses risks in training a machine learning model. One way to address skewed data is by transforming the data using a log base 2. Checking the skewness of the final table, it can be noticed that there are 16 columns with skewness of more than 2 which needs to be transformed.

Table 10 Skewness

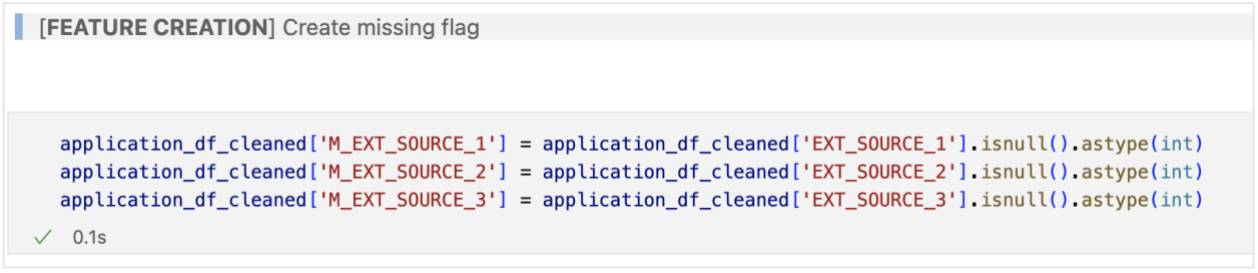
Features	Skewness
YEARS_BUILD_MEDI	-1.49
EXT_SOURCE_2	-0.77
DAYS_ID_PUBLISH	-0.34
EXT_SOURCE_3	-0.29
CB_MIN_DAYS_CREDIT	-0.29
EXT_SOURCE_1	-0.15
M_EXT_SOURCE_1	-0.02
CF_PREV_DEBT_TO_CREDIT_RATIO	0.00
CF_CB_PCT_DEBT	0.02
AGE	0.15
CF_DAYS_LAST_PHONE_CHANGE	0.42
AMT_CREDIT_TO_ANNUITY	0.57

CB_PCT_ACTIVE_LOANS	0.57
CF_CB_MONTHS_0_AVG	0.78
CF_REGION_POPULATION_RELATIVE	0.87
CF_AMT_INCOME_TOTAL	0.87
AMT_REQ_CREDIT_BUREAU_YEAR	0.96
AMT_ANNUITY	1.04
CF_OBS_30_CNT_SOCIAL_CIRCLE	1.09
PREV_SUM_AMT_CREDIT	2.06
CNT_ACTIVE_LOANS	2.19
AMT_REQ_CREDIT_BUREAU_QRT	2.40
CC_PCT_CASH_DRAWINGS	3.24
DEF_30_CNT_SOCIAL_CIRCLE	3.60
M_EXT_SOURCE_3	3.69
APARTMENTS_MEDI	4.04
BASEMENTAREA_MEDI	5.90
AMT_REQ_CREDIT_BUREAU_WEEK	6.45
LANDAREA_MEDI	7.58
INS_AVG_LESS_PAYMENT	9.27
AMT_REQ_CREDIT_BUREAU_HOUR	12.63
PREV_AVG_DPD_DAYS	21.36
CB_MONTHS_5_AVG	23.27
CC_AVG_AMT_DRAWINGS	26.26
M_EXT_SOURCE_2	128.51

4 FEATURE ENGINEERING

We have also created some columns in the final data as follows:

M_EXT_SOURCE_1, M_EXT_SOURCE_2, M_EXT_SOURCE_3
Missing flags were created for each observation that has a missing EXT_SOURCE_1, EXT_SOURCE_2 and EXT_SOURCE_3.



DEBT_TO_INCOME_RATIO

This column computes the ratio of the total debt a customer has over the total income amount.

```
#creating DEBT_TO_INCOME_RATIO column
application_df_cleaned['DEBT_TO_INCOME_RATIO'] = application_df_cleaned['CB_AMT_CREDIT_SUM_DEBT'] / application_df_cleaned['AMT_INCOME_TOTAL']
```

✓ 0.3s

AMT_CREDIT_TO_ANNUITY ratio

This column aims to show the number of times the client needs to pay annuity before it fully pays the credit amount.

```
#creating AMT_CREDIT_TO_ANNUITY Ratio
application_df_cleaned['AMT_CREDIT_TO_ANNUITY'] = application_df_cleaned['AMT_CREDIT'] / application_df_cleaned['AMT_ANNUITY']
```

✓ 0.1s

5 FEATURE SELECTION

Since we have modified some columns during the EDA and preprocessing, we will perform feature selection using two methods. First, we will use the Chi-square test to check if a variable is still statistically significant (p-value below 0.5) and remove it if not. Second, we will use principal component analysis (PCA) to determine the variables that will explain 95% of the variance.

5.1 Chi-square test

Based on the Chi-square results, 26 out of 31 categorical features are selected since the p-values are below 0.5.

Table 11 Feature selection - chi-square

Feature	Chi-Square	P-value
TARGET	33020.4519	0.0000000
CB_AVG_DAYS_CREDIT_ENDDATE	158.424422	0.0000000
REGION_RATING_CLIENT_W_CITY	140.122895	0.0000000
REG_CITY_NOT_WORK_CITY	93.247077	0.0000000
DEBT_TO_INCOME_RATIO	108.277114	0.0000000
NAME_INCOME_TYPE	83.366171	0.0000000
FLAG_EMP_PHONE	83.366171	0.0000000
REG_CITY_NOT_LIVE_CITY	83.298582	0.0000000
CODE_GENDER	75.845072	0.0000000
NAME_EDUCATION_TYPE	72.159737	0.0000000
YEARS_EMPLOYED	70.094005	0.0000000

OCCUPATION_TYPE	71.891229	0.0000000
INS_AVG_LATE_PAYMENT	60.820113	0.0000000
FLAG_DOCUMENT_3	49.252235	0.0000000
CB_AMT_CREDIT_SUM_OVERDUE	52.357208	0.0000000
CB_SEC_LOAN	34.31572	0.0000000
POS_CNT_INSTALLMENTS_FUTURE	38.236928	0.0000010
FLAG_OWN_CAR	23.810156	0.0000011
NAME_HOUSING_TYPE	35.393287	0.0000013
FLAG_WORK_PHONE	22.199719	0.0000025
CB_MONTHS_1_AVG	27.25724	0.0000052
FLAG_PHONE	18.322859	0.0000186
CNT_FAM_MEMBERS	14.52222	0.0007023
CB_TOTAL_CNT_CREDIT_PROLONG	11.742492	0.0028194
OWN_CAR_AGE	16.738191	0.0102952
NAME_FAMILY_STATUS	7.802566	0.0202160
AMT_REQ_CREDIT_BUREAU_MON	5.407696	0.0669474
CB_MONTHS_4_AVG	4.995063	0.0822879
AMT_REQ_CREDIT_BUREAU_DAY	3.866794	0.1446559
CB_UNSEC_LOAN	9.083784	0.1689206
CB_MONTHS_2_AVG	3.613162	0.3063792

There are remaining 61 columns after removing the five variables that are not statistically significant.

```
application_df_final2 = application_df_cleaned2.drop(columns=removed_features)
application_df_final2.shape
```

✓ 0.2s Python

```
(33032, 62)
```

```
application_df_final2 = application_df_final2.drop(columns='SK_ID_CURR')
application_df_final2.shape
```

✓ 0.2s Python

```
(33032, 61)
```

5.2 Principal Component Analysis (PCA)

Principal component analysis (PCA) is a technique used to reduce data dimensionality and select important features for modeling. In this project, we will only use the PCA to determine the variables that can explain 95% of variance in the data.

5.2.1 Performing one-hot-encoding for categorical variables

Before performing PCA, we conducted one-hot-encoding to categorical variables which is a technique to convert categories to numerical format. This resulted to a total of 131 features.

```python
#One hot encoding
final_data = pd.get_dummies(application_df_final2, drop_first=False)

bool_columns=[x for x in final_data.columns if final_data[x].dtype == 'bool']
for i in bool_columns:
    final_data[i] = final_data[i].astype(int)
```

Python

```python
final_data.shape
```

Python

```
(33032, 131)
```

5.2.2 PCA results

The PCA returned 71 unique variables that explain 95% of the variance in the dataset. These columns are now the final features to be used for modeling.

Table 12 Final features for modeling

Columns after PCA	Description
AMT_REQ_CREDIT_BUREAU_HOUR	Number of inquiries 1 hr before the application
AMT_REQ_CREDIT_BUREAU_WEEK	Number of inquiries 1 wk before the application
APARTMENTS_MEDI	Normalized information about the building (median apartment size)
CB_AMT_CREDIT_SUM_OVERDUE_100+	Amount overdue in Credit bureau
CB_AMT_CREDIT_SUM_OVERDUE_26-50k	Amount overdue in Credit bureau
CB_AMT_CREDIT_SUM_OVERDUE_51-75k	Amount overdue in Credit bureau
CB_AMT_CREDIT_SUM_OVERDUE_76-100k	Amount overdue in Credit bureau
CB_AMT_CREDIT_SUM_OVERDUE_no_overdue	Amount overdue in Credit bureau
CB_AVG_DAYS_CREDIT_ENDDATE_1-2yrs	AVG remaining duration of CB credit (in days) at the time of application in Home Credit
CB_AVG_DAYS_CREDIT_ENDDATE_2yrs+	AVG remaining duration of CB credit (in days) at the time of application in Home Credit
CB_AVG_DAYS_CREDIT_ENDDATE_3-6months	AVG remaining duration of CB credit (in days) at the time of application in Home Credit
CB_AVG_DAYS_CREDIT_ENDDATE_7-12months	AVG remaining duration of CB credit (in days) at the time of application in Home Credit
CB_AVG_DAYS_CREDIT_ENDDATE_below3months	AVG remaining duration of CB credit (in days) at the time of application in Home Credit

CB_MONTHS_1_AVG_21xandup	AVG times the client has a DPD within 30days (credit bureau)
CB_MONTHS_1_AVG_none	AVG times the client has a DPD within 30days (credit bureau)
CB_MONTHS_5_AVG	AVG times the client has a DPD more than 120+ days or written off
CB_SEC_LOAN_0	Number of secured loans
CB_SEC_LOAN_3+	Number of secured loans
CB_TOTAL_CNT_CREDIT_PROLONG_0	count of times the Credit Bureau credit prolonged
CB_TOTAL_CNT_CREDIT_PROLONG_3+	count of times the Credit Bureau credit prolonged
CC_AVG_AMT_DRAWINGS	Average amount of credit card drawings
CC_PCT_CASH_DRAWINGS	Percentage of credit card cash drawings
CF_CB_MONTHS_0_AVG	Average number of times the client paid on time (credit bureau)
CF_OBS_30_CNT_SOCIAL_CIRCLE	How many observation of client's social surroundings with observable 30 DPD (days past due) - payment behavior
CNT_ACTIVE_LOANS	Number of active loans
CNT_FAM_MEMBERS_1-2	Count of family members
DAYS_ID_PUBLISH	How many days before the application did client change the identity document with which he applied for the loan
DEBT_TO_INCOME_RATIO_21-40pct	Debt to income ratio
DEBT_TO_INCOME_RATIO_41-60pct	Debt to income ratio
DEBT_TO_INCOME_RATIO_61-80pct	Debt to income ratio
DEBT_TO_INCOME_RATIO_81pct_andup	Debt to income ratio
DEBT_TO_INCOME_RATIO_none	Debt to income ratio
DEF_30_CNT_SOCIAL_CIRCLE	How many observation of client's social surroundings defaulted on 30 DPD (days past due)
EXT_SOURCE_1	Normalized credit score from external source
EXT_SOURCE_2	Normalized credit score from external source
EXT_SOURCE_3	Normalized credit score from external source
FLAG_DOCUMENT_3_0	client did not submit document 3
FLAG_OWN_CAR_N	Client has no car
FLAG_WORK_PHONE_0	client has no work phone
INS_AVG_LATE_PAYMENT_21-30days	average number of days client has late payment (installments)
INS_AVG_LATE_PAYMENT_zero	average number of days client has late payment (installments)
LANDAREA_MEDI	Median size of landarea
NAME_EDUCATION_TYPE_Lower secondary	client's education type
NAME_EDUCATION_TYPE_Secondary	client's education type
NAME_FAMILY_STATUS_Married	client's family status
NAME_FAMILY_STATUS_Widow_Separated	client's family status
NAME_HOUSING_TYPE_Co-op apartment	Housing type

NAME_HOUSING_TYPE_House / apartment	Housing type
NAME_HOUSING_TYPE_Municipal apartment	Housing type
NAME_HOUSING_TYPE_Office apartment	Housing type
NAME_HOUSING_TYPE_Rented apartment	Housing type
NAME_INCOME_TYPE_Employed	Client's income is employment
OCCUPATION_TYPE_1_Manager	Occupation
OCCUPATION_TYPE_2_Professionals	Occupation
OCCUPATION_TYPE_8_Operators_Assemblers	Occupation
OCCUPATION_TYPE_9_Elementary_Occupation	Occupation
OWN_CAR_AGE_11-15	Car age
OWN_CAR_AGE_16-20	Car age
OWN_CAR_AGE_21-25	Car age
OWN_CAR_AGE_26-30	Car age
POS_CNT_INSTALLMENTS_FUTURE_1-15	Remaing count of installments for POS_CASH loans
POS_CNT_INSTALLMENTS_FUTURE_16-30	Remaing count of installments for POS_CASH loans
POS_CNT_INSTALLMENTS_FUTURE_31-45	Remaing count of installments for POS_CASH loans
POS_CNT_INSTALLMENTS_FUTURE_46-60	Remaing count of installments for POS_CASH loans
POS_CNT_INSTALLMENTS_FUTURE_76+	Remaing count of installments for POS_CASH loans
PREV_AVG_DPD_DAYS	Average DPD days in the previous applications
REGION_RATING_CLIENT_W_CITY_3	Rating of client's location is 3
REG_CITY_NOT_LIVE_CITY_0	Registered address is the same as contact address
YEARS_EMPLOYED_11-15	Years employed
YEARS_EMPLOYED_31+	Years employed
YEARS_EMPLOYED_6-10	Years employed

6 MODEL EXPLORATION

6.1 Preparing final table for modeling

The following steps are needed to perform first before training the machine learning models.

6.1.1 Creating features and target data frame

Creating features and target data frames allows the model to distinguish the input (independent) variables from the target (dependent) variable. These separate data structures will be used to help train the models to predict the target variable based on the input features.

```
#Create X and y dataframes
X = df.drop(columns=['TARGET'])
y = df['TARGET']
```
✓ 0.6s

6.1.2 Normalizing dataset

Since the dataset has a wide range of values, we need to do normalization or feature scaling to transform the values into a common scale. Normalization ensures that each feature contributes proportionally to the model's learning process which leads to better performance and more stable training of machine learning models. Several techniques were used for this study, but min-max scaling provided the best outcome. Min-max scaling converts the features to a specified range, in this case [0,1], and preserves the relationship between datapoints.

```
#Normalizing dataset
scaler = MinMaxScaler()
scaled_df = scaler.fit_transform(X)
scaled_df = pd.DataFrame(scaled_df, columns=X.columns)
```
✓ 0.1s

6.1.3 Data partition

The data partition is where we split the dataset into training and validation purposes. The training set will be used to train the model and learn the patterns and relationships within the data while the validation/holdout set is used to optimize the model by changing the hyperparameters and selecting the best model. The validation/holdout set aims to simulate real-world scenarios where the model is run on the new unseen data.

We split the data 50:50 and used stratified sampling (stratify=y) which guarantees that each split contains the same general class ratio of having only 10% default clients. Given that our data is imbalanced, this helps ensure that our models are trained and evaluated on the data that resembles the distribution of the original dataset.

```python
#split train and test
train_X, valid_X, train_y, valid_y = train_test_split(scaled_df, y, test_size=0.5, random_state = 1, stratify=y)
    #stratify ensures that both have the same general class ratio
print(train_X.shape)
print(valid_X.shape)
```

✓ 0.1s Python

```
(16516, 71)
(16516, 71)
```

6.1.4 SMOTE-ENN: Simultaneous oversampling and undersampling

SMOTE-ENN or Synthetic Minority Over-sampling Technique combined with Edited Nearest Neighbors is a sampling method that addresses class imbalance by simultaneously oversampling and undersampling the dataset. It creates a more balanced dataset by generating synthetic samples for the minority class while removing noisy or misclassified instances from both minority and majority classes.

```python
# Apply SMOTE-ENN
smote_enn = SMOTEENN(sampling_strategy= 0.55, random_state=42)
X_resampled_enn, y_resampled_enn = smote_enn.fit_resample(train_X, train_y)
```

✓ 4.6s

✓ #Final train dataset ···

```
After SMOTE-ENN, counts of label '1': 7611
After SMOTE-ENN, counts of label '0': 9242
After SMOTE-ENN, the shape of train_X: (16853, 71)
```

```python
#renaming
train_X = X_resampled_enn
train_y = y_resampled_enn
```

✓ 0.1s

6.2 Modeling

Several models are used in this study such as decision trees, logistic regression, random forest, stochastic gradient descent classifier and XGboost. Hyperparametering tuning was also done for each model to compare which is better in predicting the default. The setting class_weight = ‘balanced’ was used to avoid the bias to the majority class by assigning weights to the classes inversely proportional to their frequencies.

The following metrics are used to evaluate the models:

1. ROC-AUC score is the primary metric we want to use to evaluate the model since is less affected by the imbalanced class distribution and it measures how well the model classifies a

default from non-default. The score ranges from 0-1 where 0.5 means random guessing and 1 means a perfect performance.

- 0.5: Random performance. The model's predictions are no better than random guessing.
- 0.5 - 0.6: Weak performance. The model has a limited ability to discriminate between the classes.
- 0.6 - 0.7: Fair performance. The model is better than random guessing, but improvements are possible.
- 0.7 - 0.8: Good performance. The model has a reasonably strong ability to discriminate between the classes.
- 0.8 - 0.9: Very good performance. The model has a strong ability to discriminate between the classes.
- 0.9 - 1.0: Excellent performance. The model has an almost perfect ability to discriminate between the classes.

2. F1-score is the second metric we're going to use which combines precision and recall into a single value. Since predicting both classes are important in this study, we aim to get the balance of the precision and recall instead of focusing only on the accuracy.
3. Accuracy tells us how accurate our model is in predicting the data regardless of the class.

6.2.1 Decision Tree

A decision tree is one of the most used machine learning algorithms in classification problems because of its ease of use and interpretability. It visualizes predictions with the use of tree-like structure where each internal node represents the feature, each branch represents a decision rule, and each leaf node (terminal node) denotes the outcome of the class label. Aside from providing results that are highly interpretable even to non-technical audiences, some of the advantages of using decision tree is that it can automatically ignore irrelevant features, robust to outliers, and can capture non-linear relationships in the data if any.

6.2.1.1 Model parameters

We defined our classification tree model below with only class_weight = 'balanced' as our parameter.

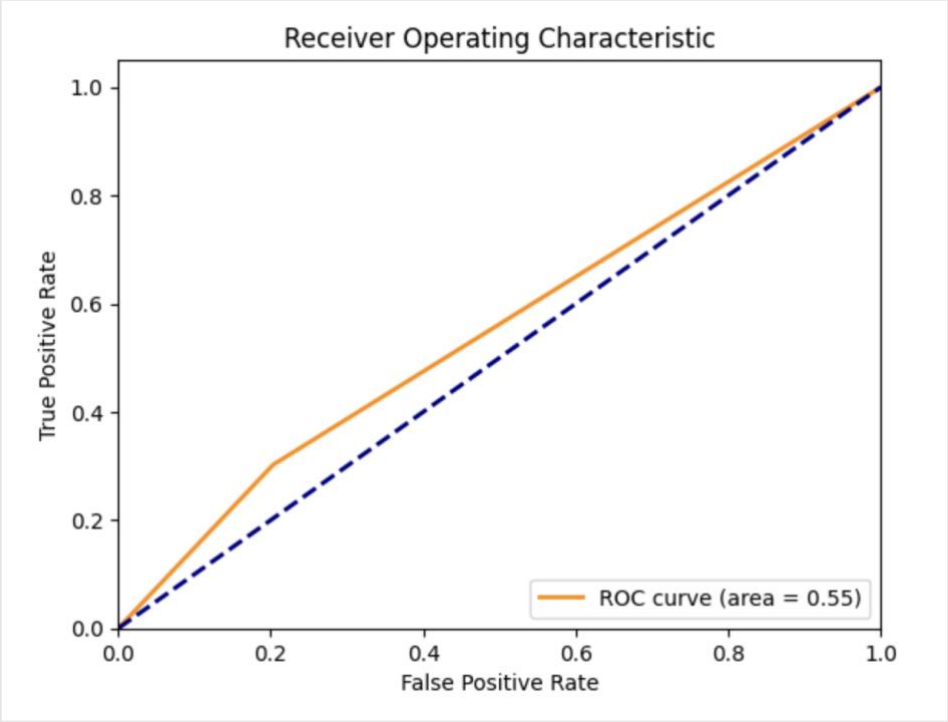
```
classtree = DecisionTreeClassifier(random_state=1, class_weight='balanced')
classtree.fit(train_X, train_y)
```

6.2.1.2 Model Performance

Model	ROC-AUC score	F1_score	Accuracy
Decision_tree	0.55	0.19	0.75

Evaluating the model performance, the ROC-AUC score is only 0.55 which indicates that our model is better than random (dotted line) but has a limited ability to discriminate a default from non-default.

Table 13 ROC-AUC curve for Decision Tree



The F1_score is 0.19 which means the model performance is poor and having difficulty capturing true positive instances. We can see this in the classification matrix below that the precision and recall are both low. The model has a recall of 30% which means that out of all defaults (1582), it can only capture 478. On the other hand, a precision tells us that the model is 14% right in identifying a default. In terms of accuracy, the model returns a correct prediction 75% of the time regardless of the class.

	Prediction	
Actual	not default	default
not default	11903	3031
default	1104	478

Recall = True Positive / (True Positive + False Negative)
= 478 / (478 + 1104)
= 0.3

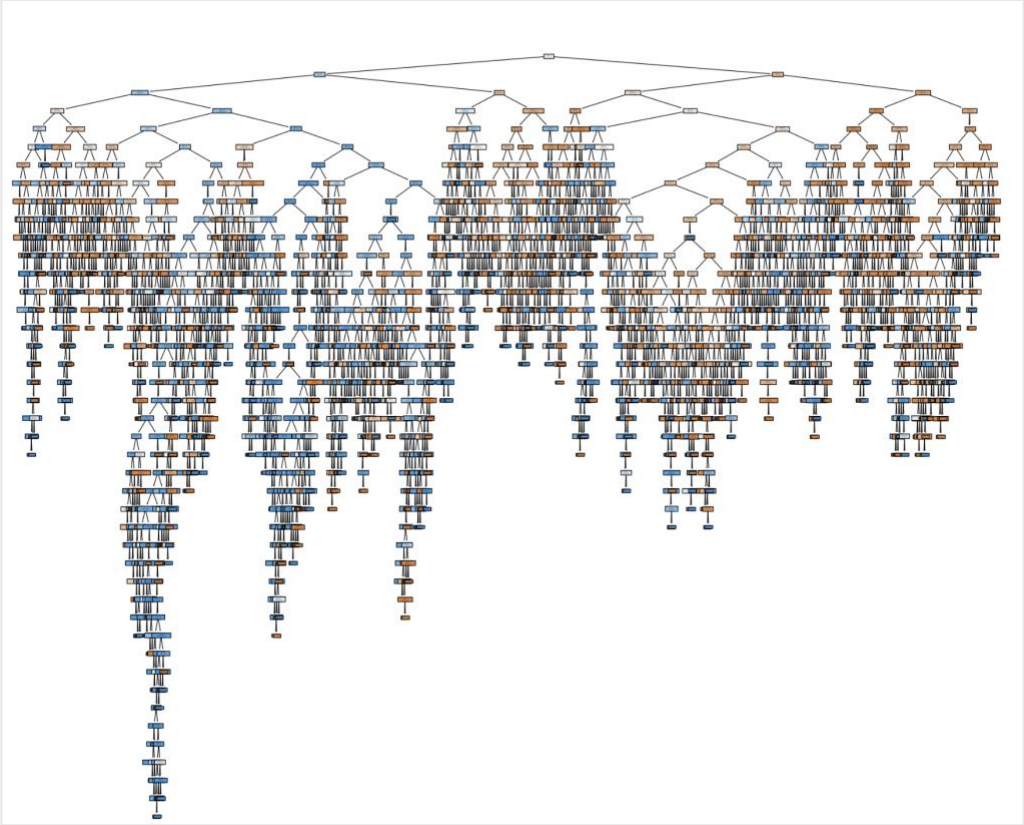
Precision = True Positive / (True Positive + True Negative)
= 478 / (478+3031)
= 0.14

6.2.1.3 Visualizing tree

Unfortunately, since we have a large dataset the tree map cannot be visualized better. The tree has a total of 4254 splits and 2218 terminal nodes. The first split used EXT_SOURCE_3 followed by EXT_SOURCE_2 and NAME_EDUCATION_TYPE_Secondary.

Split Number	Feature Name	Threshold
1	EXT_SOURCE_3	0.564357
2	EXT_SOURCE_2	0.761995
3	NAME_EDUCATION_TYPE_Secondary	0.000790
4	CNT_FAM_MEMBERS_1-2	0.946986
5	CNT_FAM_MEMBERS_1-2	0.003263

Figure 21 Decision tree map



6.2.1.4 Feature Importance

Ranking the important features of this model, we can see that the most important features are the credit scores (EXT_SOURCE_1, EXT_SOURCE_2) from external sources followed by the number of observable default in the client's social connection (CF_OBS_30_CNT_SOCIAL_CIRCLE), the number of days the client changed the identity document (DAYS_ID_PUBLISH) , and the average amount drawing from the credit card (CC_AVG_AMT_DRAWINGS).

Figure 22 Decision Tree – Top 20 Feature Importance

	feature	importance
23	EXT_SOURCE_3	0.136759
35	EXT_SOURCE_2	0.097097
40	CF_OBS_30_CNT_SOCIAL_CIRCLE	0.069621
70	DAYS_ID_PUBLISH	0.051943
56	CC_AVG_AMT_DRAWINGS	0.051314
66	CF_CB_MONTHS_0_AVG	0.045810
20	EXT_SOURCE_1	0.043398
18	APARTMENTS_MEDI	0.034856
62	CNT_ACTIVE_LOANS	0.030753
24	PREV_AVG_DPD_DAYS	0.030673
21	NAME_EDUCATION_TYPE_Secondary	0.030032
27	CB_AVG_DAYS_CREDIT_ENDDATE_2yrs+	0.027250
60	INS_AVG_LATE_PAYMENT_zero	0.026849
12	LANDAREA_MEDI	0.026260
8	NAME_INCOME_TYPE_Employed	0.023967
38	FLAG_OWN_CAR_N	0.020538
30	YEARS_EMPLOYED_6-10	0.020183
15	CNT_FAM_MEMBERS_1-2	0.017681
42	FLAG_WORK_PHONE_0	0.016799
43	CB_SEC_LOAN_0	0.015365

6.2.2 GridSearch Decision Tree

Here we used gridsearch to optimize the parameters of the base model Decision tree.

```
param_grid = {'max_depth': [40, 50,60,70],
              'min_samples_split' : [2,5,10,20,30,40], #based on number of records
              'min_impurity_decrease': [0.05,0.02,0.01, 0.001,0.0001, 0.00001]}
```

✓ 0.6s

```
gridsearch = GridSearchCV(DecisionTreeClassifier(random_state=1, class_weight='balanced'),
                          param_grid, cv=5, n_jobs = -1)
gridsearch.fit(train_X, train_y)
bestregtree = gridsearch.best_estimator_
print(gridsearch.best_params_)
plotDecisionTree(bestregtree, feature_names=train_X.columns)
```

✓ 1m 33.4s

```
{'max_depth': 40, 'min_impurity_decrease': 0.0001, 'min_samples_split': 2}
```

6.2.2.1 Model parameters

The gridsearch results showed the max_depth of 40, min_impurity_decrease of 0.0001 and minimum sample split of 2 which we used to train our model.

```
classtree_gs = DecisionTreeClassifier(random_state=1, class_weight='balanced',max_depth=40, min_impurity_decrease=0.0001, min_samples_split=2)
classtree_gs.fit(train_X, train_y)
```

6.2.2.2 Model performance

Based on the results the F1_score is higher by 1 point (0.2) but has a lower accuracy of 0.74 and ROC-AUC score of 0.53.

Model	ROC-AUC score	F1_score	Accuracy
Decision_Tree_GridSearch	0.53	0.2	0.74

The model has a slightly better recall of 33% and the same precision of 14%. Unfortunately, overall, this is worse than the base model.

	Prediction	
Actual	not default	default
not default	11771	3163
default	1056	526

Recall = True Positive / (True Positive + False Negative)
= 526 / (526 + 1056)
= 0.33

Precision = True Positive / (True Positive + True Negative)
= 526 / (526+3163)
= 0.14

6.2.2.3 Feature Importance

The top 3 feature importance is the same with the first model, EXT_SOURCE_3, EXT_SOURCE_2, CF_OBS_30_CNT_SOCIAL_CIRCLE.

Figure 23 Gridsearch Decision Tree - Top 20 Feature Importance

	feature	importance
23	EXT_SOURCE_3	0.146449
35	EXT_SOURCE_2	0.101237
40	CF_OBS_30_CNT_SOCIAL_CIRCLE	0.075676
56	CC_AVG_AMT_DRAWINGS	0.051743
20	EXT_SOURCE_1	0.042926
70	DAYS_ID_PUBLISH	0.040677
66	CF_CB_MONTHS_0_AVG	0.039294
62	CNT_ACTIVE_LOANS	0.033454
21	NAME_EDUCATION_TYPE_Secondary	0.032491
18	APARTMENTS_MEDI	0.032448
24	PREV_AVG_DPD_DAYS	0.031637
27	CB_AVG_DAYS_CREDIT_ENDDATE_2yrs+	0.030682
60	INS_AVG_LATE_PAYMENT_zero	0.029117
8	NAME_INCOME_TYPE_Employed	0.027444
12	LANDAREA_MEDI	0.023486
38	FLAG_OWN_CAR_N	0.022079
30	YEARS_EMPLOYED_6-10	0.020796
15	CNT_FAM_MEMBERS_1-2	0.017845
43	CB_SEC_LOAN_0	0.016238
42	FLAG_WORK_PHONE_0	0.015244

6.2.3 Logistic Regression (Full)

Logistic regression is used for binary and multiclass classification problems which uses the logistic function or sigmoid function to model the relationship between the input features and the probability of the target variable to be categorized as a certain class. The default probability cutoff

is 0.5, anything higher than 0.5 is categorized as 1 (in this case client has defaulted) and lower than 0.5 is categorized as 0.

6.2.3.1 Model parameters

Maximum iteration of 1500 is used to help the model achieve convergence or a stable and optimal solution. The liblinear solver uses the coordinate descent method to optimize the objective function where it updates one coefficient at a time while keeping the others constant.

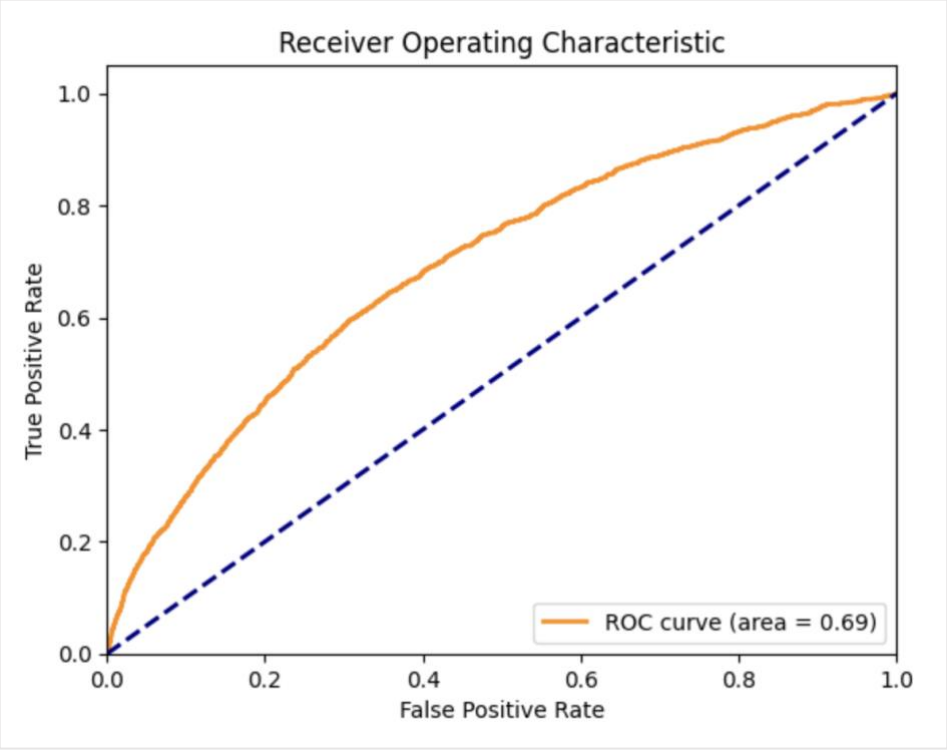
```
logit_reg = LogisticRegression(max_iter=1500, C=1e42, random_state=4, solver='liblinear', class_weight='balanced')
logit_reg.fit(train_X, train_y)
```

6.2.3.2 Model performance

The model has a ROC-AUC score of 0.69 which means that the model is better than random but has fair performance. The F1 score is 0.26 and can accurately predict classes 64% of the time.

Model	ROC-AUC score	F1_score	Accuracy
Logistic_Regression_Full	0.69	0.26	0.64

Figure 24 ROC-AUC score - Logistic regression



Checking the confusion matrix, the model has the ability to capture 64% of the actual defaulted clients (recall) and can correctly predict a default client by 14% (precision).

	Prediction	
Actual	not default	default
not default	9624	5310
default	566	1016

Recall = True Positive / (True Positive + False Negative)
= 1016 / (1016 + 566)
= 0.64

Precision = True Positive / (True Positive + True Negative)
= 1016 / (1016+5310)
= 0.16

6.2.3.3 Feature importance

Since we have scaled our variables before modeling, we need to unscale the values when interpreting the coefficients. The top 10 important features based on odds ratio are showed below where the most important feature is the variable where the total overdue is between 76-100k (CB_AMT_CREDIT_SUM_OVERDUE_76-100k) followed by the number of times an inquiry was made to the credit bureau 1 week before the application.

Figure 25 Logistic regression - Odds ratio

	coef	unscaled_coef	odds_ratio
CB_AMT_CREDIT_SUM_OVERDUE_76-100k	52.129846	52.129846	4.362191e+22
AMT_REQ_CREDIT_BUREAU_WEEK	1.869310	4.832095	1.254736e+02
CB_MONTHS_5_AVG	0.582999	1.739646	5.695327e+00
NAME_INCOME_TYPE_Employed	1.098454	1.098454	2.999525e+00
CB_SEC_LOAN_0	0.898402	0.898402	2.455675e+00
DEF_30_CNT_SOCIAL_CIRCLE	0.356186	0.827037	2.286534e+00
OCCUPATION_TYPE_8_Operators_Assemblers	0.555104	0.555104	1.742122e+00
OWN_CAR_AGE_21-25	0.512366	0.512366	1.669235e+00
FLAG_OWN_CAR_N	0.508808	0.508808	1.663307e+00
NAME_EDUCATION_TYPE_Secondary	0.485060	0.485060	1.624273e+00

Interpretation of odds ratio:

Feature	Interpretation
CB_AMT_CREDIT_SUM_OVERDUE_76-100k	When the credit sum overdue is between 76-100k, the probability of default increases by 4.36e^22 times.
AMT_REQ_CREDIT_BUREAU_WEEK	For every increase in the number of inquiries made to the credit bureau 1 week before the application, the probability of default increases by 125 times.
CB_MONTHS_5_AVG	For every increase in the average number of times the client has a delayed payment more than 120days, the probability of default increases by 5.69 times.
NAME_INCOME_TYPE_Employed	If the client is employed, the probability of default increases by 2.99 times.
CB_SEC_LOAN_0	If the client has no secured loans, the probability of default increases by 2.46 times.
DEF_30_CNT_SOCIAL_CIRCLE	For every increase in the number of defaulted customers within the client's circle, the probability of default increases by 2.29 times.
OCCUPATION_TYPE_8_Operators_Assemblers	If the client belongs to occupation type 8 (Operators and Assemblers), the probability of default increases by 74%.
OWN_CAR_AGE_21-25	If the client's car age is between 21-25, the probability of default increases by 67%.
FLAG_OWN_CAR_N	If the client has no car, the probability of default increases by 66%.
NAME_EDUCATION_TYPE_Secondary	If the highest educational attainment of the client is secondary, the probability of default increases by 62%.

6.2.4 GridSearch Logistic Regression (Full)

The best parameter is searched by setting the hyperparameters below:

```
# Define logistic regression model
model = LogisticRegression(random_state=42, max_iter=2500, class_weight='balanced')

# Define hyperparameter grid
param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],  # Regularization parameter
    'penalty': ['l1', 'l2'],              # Regularization type
    'solver': ['liblinear', 'lbfgs', 'newton-cg', 'sag', 'saga']  # Optimization algorithm
}

# Create GridSearchCV object
grid_search2 = GridSearchCV(estimator=model, param_grid=param_grid, scoring='roc_auc', cv=5)

# Fit the grid search to the training data
grid_search2.fit(train_X, train_y)
```

6.2.4.1 Model parameters

Based on the results above, the model is trained using the parameters below such as maximum iteration of 2500, l1 regularization and solver saga.

```
logit_gs = LogisticRegression(max_iter=2500, C=1e42, penalty='l1', random_state=4, solver='saga', class_weight='balanced')
logit_gs.fit(train_X, train_y)
```

6.2.4.2 Model performance

The model performance shows that it has the same results with the base model logistic regression.

Model	ROC-AUC score	F1_score	Accuracy
Logistic_Regression_Gridsearch	0.69	0.26	0.64

6.2.4.3 Feature importance

The top 10 important features are also the same with the base logistic regression model but have different values for odds ratio.

	coef	unscaled_coef	odds_ratio
CB_AMT_CREDIT_SUM_OVERDUE_76-100k	23.472148	23.472148	1.562515e+10
AMT_REQ_CREDIT_BUREAU_WEEK	1.869023	4.831354	1.253807e+02
CB_MONTHS_5_AVG	0.582193	1.731370	5.648386e+00
NAME_INCOME_TYPE_Employed	1.098469	1.098469	2.999570e+00
CB_SEC_LOAN_0	0.898031	0.898031	2.454764e+00
DEF_30_CNT_SOCIAL_CIRCLE	0.356314	0.827336	2.287217e+00
OCCUPATION_TYPE_8_Operators_Assemblers	0.555056	0.555056	1.742038e+00
OWN_CAR_AGE_21-25	0.512198	0.512198	1.668955e+00
FLAG_OWN_CAR_N	0.508759	0.508759	1.663226e+00
NAME_EDUCATION_TYPE_Secondary	0.484455	0.484455	1.623290e+00

Interpretation of odds ratio:

Feature	Interpretation
CB_AMT_CREDIT_SUM_OVERDUE_76-100k	When the credit sum overdue is between 76-100k, the probability of default increases by 1.56e^10 times.
AMT_REQ_CREDIT_BUREAU_WEEK	For every increase in the number of inquiries made to the credit bureau 1 week before the application, the probability of default increases by 125 times.
CB_MONTHS_5_AVG	For every increase in the average number of times the client has a delayed payment more than 120days, the probability of default increases by 5.65 times.
NAME_INCOME_TYPE_Employed	If the client is employed, the probability of default increases by 2.99 times.
CB_SEC_LOAN_0	If the client has no secured loans, the probability of default increases by 2.45 times.
DEF_30_CNT_SOCIAL_CIRCLE	For every increase in the number of defaulted customers within the client's circle, the probability of default increases by 2.29 times.
OCCUPATION_TYPE_8_Operators_Assemblers	If the client belongs to occupation type 8 (Operators and Assemblers), the probability of default increases by 74%.
OWN_CAR_AGE_21-25	If the client's car age is between 21-25, the probability of default increases by 67%.
FLAG_OWN_CAR_N	If the client has no car, the probability of default increases by 66%.

NAME_EDUCATION_TYPE_Secondary	If the highest educational attainment of the client is secondary, the probability of default increases by 62%.

6.2.5 Random Forest

Random Forest is a type of ensemble learning used in machine learning which aims to improve the accuracy of predictions by combining the results of multiple individual decision trees each trained on a random subset of the data and features. This model is robust on overfitting and can capture complex data relationships.

6.2.5.1 Model parameters

We started modeling using 100 estimators or trees with a max depth of 10.

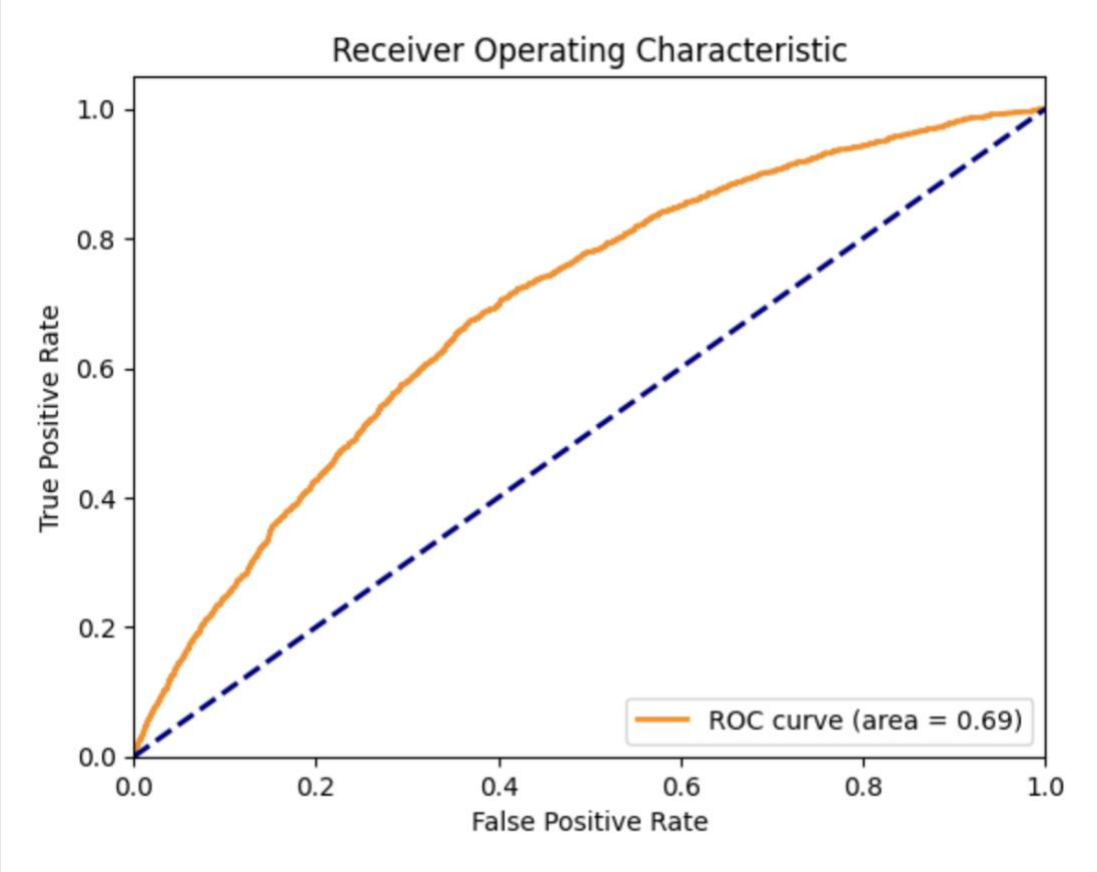
```
rf = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42, class_weight='balanced')
rf.fit(train_X, train_y)
```

6.2.5.2 Model performance

The ROC-AUC score is 0.69 which means that the model is better than random guessing but is fair performance. It has F1_score of 0.26 and can accurately predict a class 77% of the time.

Model	ROC-AUC score	F1_score	Accuracy
Random Forest	0.69	0.26	0.77

Figure 26 ROC-AUC curve - Random Forest



The recall score shows that the random forest model can capture 41% of the default while the precision tells us that when the model says it is a default, the model is right 19% of the time.

	Prediction	
Actual	not default	default
not default	12151	2783
default	941	641

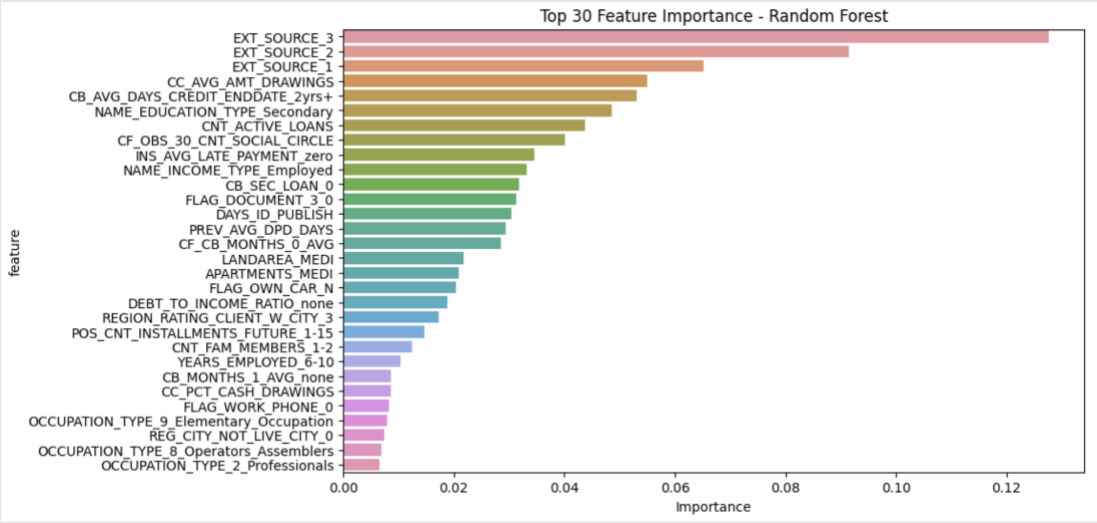
Recall = True Positive / (True Positive + False Negative)
= 641 / (641 + 941)
= 0.41

Precision = True Positive / (True Positive + True Negative)
= 641 / (641+2783)
= 0.19

6.2.5.3 *Feature importance*

The top important features from the random forest model are the external credit score (EXT_SOURCE_3, EXT_SOURCE_2, EXT_SOURCE_1) followed by the average amount of credit card drawings, and remaining end date of credit in bureau more than 2 yrs.

Figure 27 Top 30 important features - Random Forest



6.2.6 Randomized Search Random Forest

Gridsearch was found to be very computationally expensive in this model, so we used randomized search instead to find the best combination of the parameters. The parameters used for the search are included below.

```
#param_grid = {
    'n_estimators': np.arange(50, 300, 500),
    'max_depth': np.arange(5, 20, 1),
    'min_samples_split': np.arange(2, 10, 1),
    'min_samples_leaf': np.arange(1, 10, 1),
    'max_features': ['sqrt', 'log2'],
    'bootstrap': [True, False]
}

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(
    estimator=rf_classifier,
    param_distributions=param_grid,
    n_iter=50,  # Number of parameter settings that are sampled
    scoring='roc_auc',  # Use roc for evaluation
    cv=5,  # Cross-validation folds
    verbose=1,
    n_jobs=-1,  # Use all available CPU cores
    random_state=42
)

# Fit the RandomizedSearchCV object to the data
random_search.fit(train_X, train_y)
```

6.2.6.1 Model parameters

Based on the results of the randomized search, we used 250 trees, minimum number of samples required for split is 5, minimum number of samples in each leaf node is 2, and maximum depth of 19 trees.

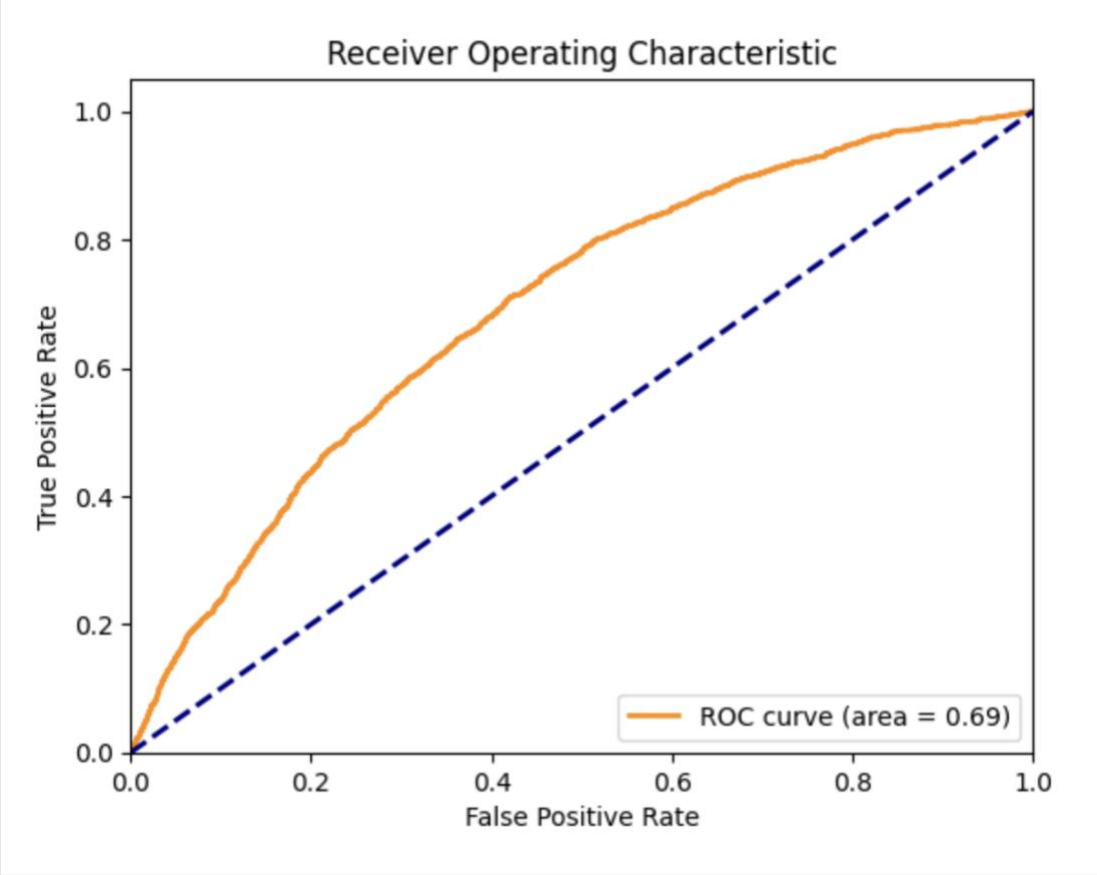
```
rf_tuned = RandomForestClassifier(n_estimators = 250, min_samples_split = 5, min_samples_leaf= 2,
                max_features= 'sqrt', max_depth= 19, bootstrap= False, random_state=42, class_weight='balanced')
rf_tuned.fit(train_X, train_y)
```

6.2.6.2 Model performance

The optimized random forest has the same ROC-AUC score of 0.69. However, as compared to the base random forest model, it has a lower F1_score of 0.21 but higher accuracy of 0.84.

Model	ROC-AUC score	F1_score	Accuracy
Random Forest_Tuned	0.69	0.21	0.84

Figure 28 ROC-AUC curve - Random Forest - Tuned



The recall score shows that the random forest model can capture 22% of the default while the precision tells us that when the model says it is a default, the model is right 20% of the time.

	Prediction	
Actual	not default	default
not default	13565	1369
default	1232	350

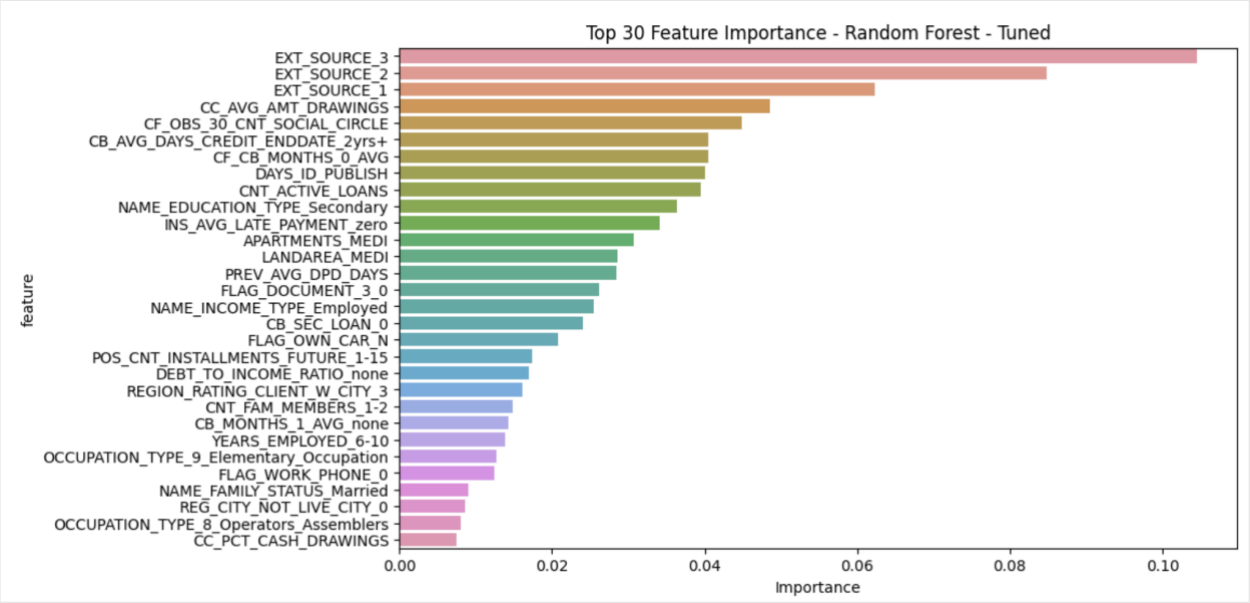
Recall = True Positive / (True Positive + False Negative)
= 350 / (350 + 1232)
= 0.22

Precision = True Positive / (True Positive + True Negative)
= 350 / (350+1369)
= 0.20

6.2.6.3 *Feature importance*

The top important features from the random forest model are the external credit score (EXT_SOURCE_3, EXT_SOURCE_2, EXT_SOURCE_1) followed by the average amount of credit card drawings, and number of observable default in the client's social circle.

Figure 29 Top 30 features - Random Forest - Tuned



6.2.7 SGD Classifier

The Stochastic Gradient Descent Classifier (SGD) is a linear classification algorithm that efficiently handles a large number of samples and features by using stochastic gradient descent as its optimization technique. Instead of computing the gradient of the entire dataset, which can be computationally expensive, SGD updates the model's parameters using small random batches (stochastic updates) of the training data which makes it suitable for large datasets.

6.2.7.1 *Model parameters*

Maximum iteration needed to converge the model is 1000, loss function to be used is log_loss or a logistic regression and tolerance (stopping criterion) is 0.0001.

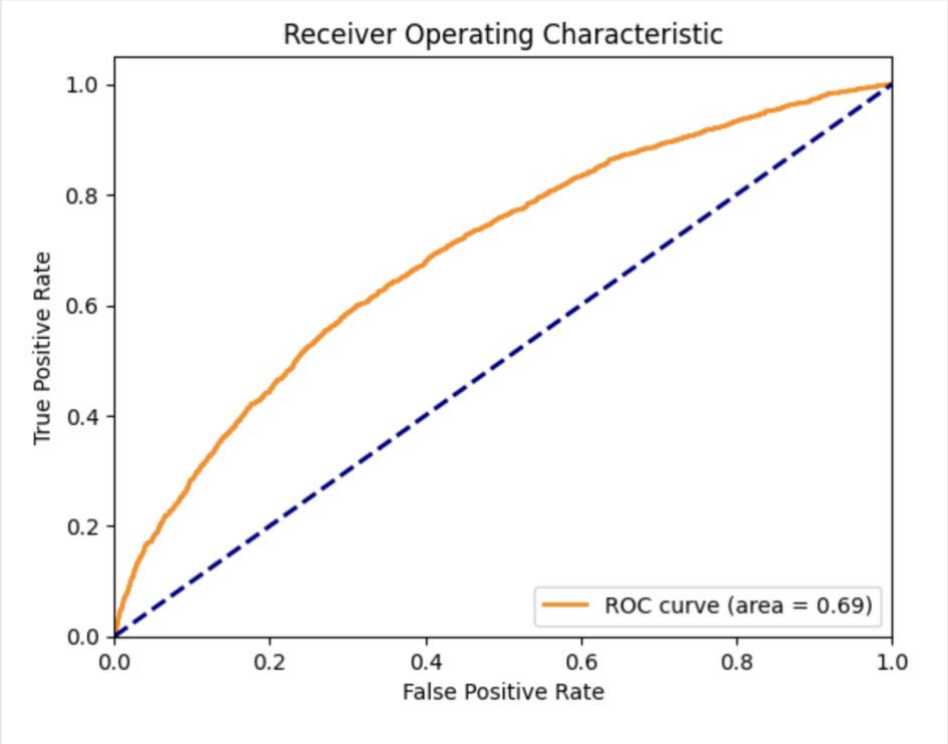
```
sgdc = SGDClassifier(max_iter=1000, tol=0.0001, loss='log_loss', random_state=4)
sgdc.fit(train_X, train_y)
```

6.2.7.2 Model performance

The model evaluation results shows that this model has the same ROC-AUC score of 0.69 and F1 score of 0.26. However, accuracy is lower at 0.68.

Model	ROC-AUC score	F1_score	Accuracy
SGD Classifier	0.69	0.26	0.68

Figure 30 ROC-AUC curve - SGD classifier



The recall score shows that the SGD classifier model can capture 59% of the default while the precision tells us that when the model says it is a default, the model is right 17% of the time.

	Prediction	
Actual	not default	default
not default	10323	4611
default	642	940

Recall = True Positive / (True Positive + False Negative)
= 940 / (940 + 642)
= 0.59

Precision = True Positive / (True Positive + True Negative)
= 940 / (940+4611)
= 0.17

6.2.7.3 Feature importance

Since SGD Classifier is a linear classifier, we can interpret the results using the odds ratio. The top 10 important features based on odds ratio are showed below where the most important feature is the variable where the total overdue is between 76-100k (CB_AMT_CREDIT_SUM_OVERDUE_76-100k) followed by the number of times an inquiry was made to the credit bureau 1 week before the application (AMT_REQ_CREDIT_BUREAU_WEEK) , income type is employed (NAME_INCOME_TYPE_Employed), clients with no secured loans (CB_SEC_LOAN_0) and number of defaulted customers in the client's social circle (DEF_30_CNT_SOCIAL_CIRCLE).

Figure 31 Top 10 Feature Importance - SGD

	coef	unscaled_coef	odds_ratio
CB_AMT_CREDIT_SUM_OVERDUE_76-100k	1.590064	1.590064	4.904062
AMT_REQ_CREDIT_BUREAU_WEEK	0.598804	1.547885	4.701518
NAME_INCOME_TYPE_Employed	1.109093	1.109093	3.031609
CB_SEC_LOAN_0	0.864272	0.864272	2.373278
DEF_30_CNT_SOCIAL_CIRCLE	0.303935	0.705715	2.025294
OCCUPATION_TYPE_8_Operators_Assemblers	0.546727	0.546727	1.727589
NAME_EDUCATION_TYPE_Secondary	0.541917	0.541917	1.719299
OWN_CAR_AGE_21-25	0.501692	0.501692	1.651513
FLAG_OWN_CAR_N	0.437463	0.437463	1.548773
CB_AVG_DAYS_CREDIT_ENDDATE_2yrs+	0.297943	0.297943	1.347085

Interpretation of odds ratio:

Feature	Interpretation
CB_AMT_CREDIT_SUM_OVERDUE_76-100k	When the credit sum overdue is between 76-100k, the probability of default increases by 4.9 times.
AMT_REQ_CREDIT_BUREAU_WEEK	For every increase in the number of inquiries made to the credit bureau 1 week before the application, the probability of default increases by 4.7 times.
NAME_INCOME_TYPE_Employed	If the client is employed, the probability of default increases by 3 times.
CB_SEC_LOAN_0	If the client has no secured loans, the probability of default increases by 2.4 times.

DEF_30_CNT_SOCIAL_CIRCLE	For every increase in the number of defaulted customers within the client's circle, the probability of default increases by 2 times.
OCCUPATION_TYPE_8_Operators_Assemblers	If the client belongs to occupation type 8 (Operators and Assemblers), the probability of default increases by 73%.
NAME_EDUCATION_TYPE_Secondary	If the highest educational attainment of the client is secondary, the probability of default increases by 72%.
OWN_CAR_AGE_21-25	If the client's car age is between 21-25, the probability of default increases by 65%.
FLAG_OWN_CAR_N	If the client has no car, the probability of default increases by 55%.
CB_AVG_DAYS_CREDIT_ENDDATE_2yrs+	If the average remaining days of a client's credit in credit bureau is more than 2years, the probability of default increases by 35%.

6.2.8 XGBoost

Extreme Gradient Boosting or XGBoost is also an ensemble learning method that combines the predictions of multiple weak learners, usually decision trees, to improve a predictive model. It uses gradient boosting as a way to optimize the model by giving more weights / attention to the instances with higher errors in the previous iterations.

6.2.8.1 Model parameters

We only used 100 estimators or trees as the parameter for the base model.

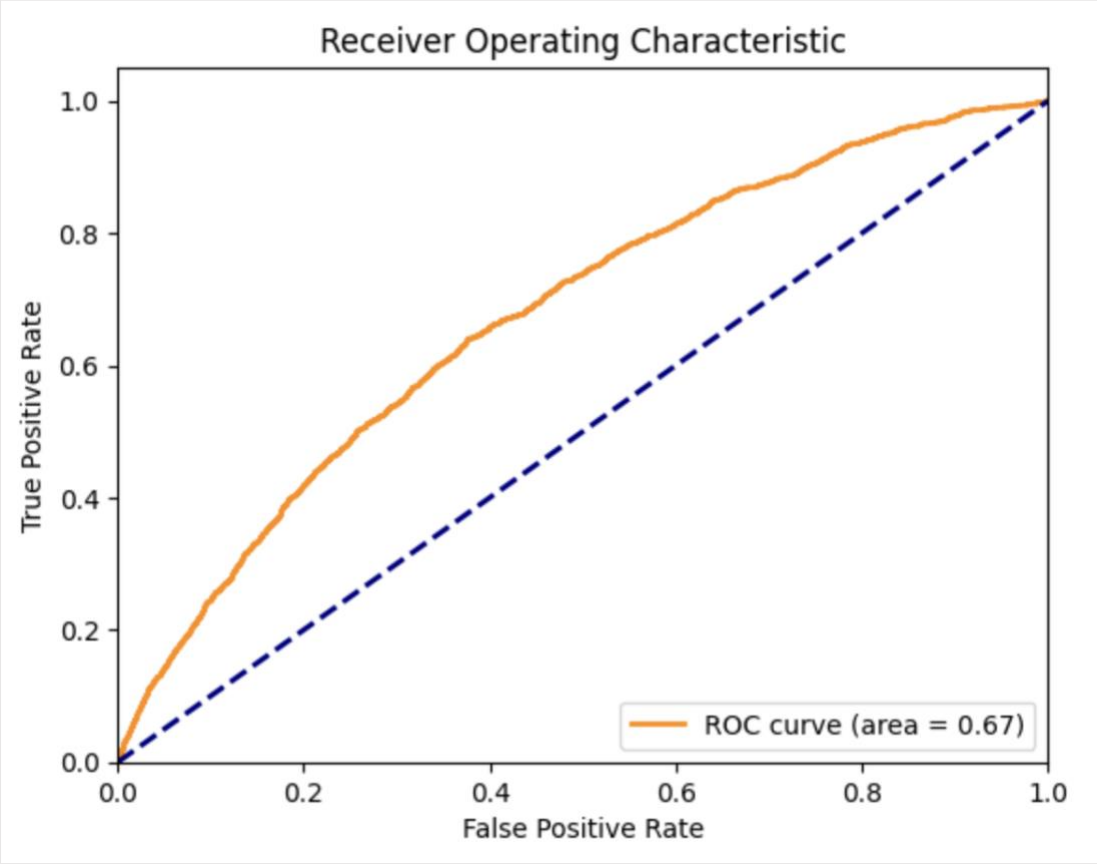
```
xgb = XGBClassifier(random_state = 1, n_estimators=100)
xgb.fit(train_X, train_y)
```

6.2.8.2 Model performance

The model has a relatively lower ROC-AUC score of 0.67, F1-score of 0.2 but has a higher accuracy score of 86%.

Model	ROC-AUC score	F1_score	Accuracy
XGBoost	0.67	0.20	0.86

Figure 32 ROC-AUC curve - XGBoost



The recall score shows that the XGboost model can capture only 18% of the default while the precision tells us that when the model says it is a default, the model is right 22% of the time.

	Prediction	
Actual	not default	default
not default	13927	1007
default	1302	280

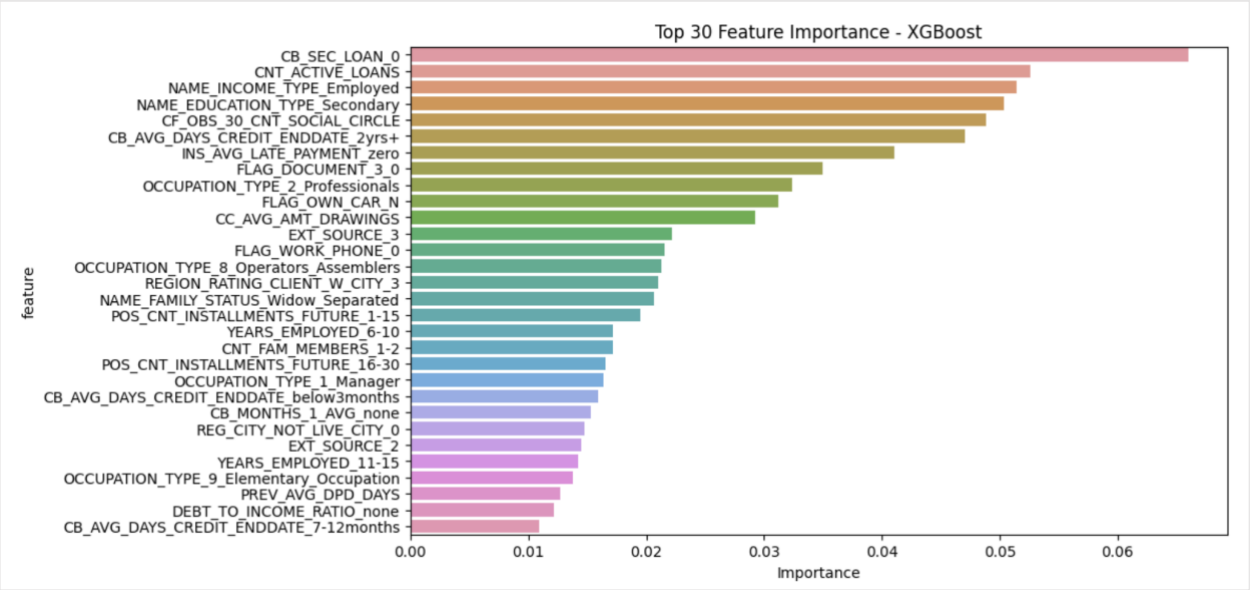
Recall = True Positive / (True Positive + False Negative)
= 280 / (280 + 1302)
= 0.18

Precision = True Positive / (True Positive + True Negative)
= 280 / (280+1007)
= 0.22

6.2.8.3 *Feature importance*

Based on the importance score, the feature with the highest feature is the number of secured loans a client has (CB_SEC_LOAN_0) followed by the number of active loans (CNT_ACTIVE_LOANS), NAME_INCOME_TYPE_Employed, NAME_EDUCATION_TYPE_Secondary and CFS_OBS_30_CNT_SOCIAL_CIRCLE.

Figure 33 Top 30 Important Features - XGboost



6.2.9 XGBoost

6.2.9.1 *Model parameters*

Tuning the XGboost by using learning rate of 0.02, 600 estimators, and binary: logistic as the objective function.

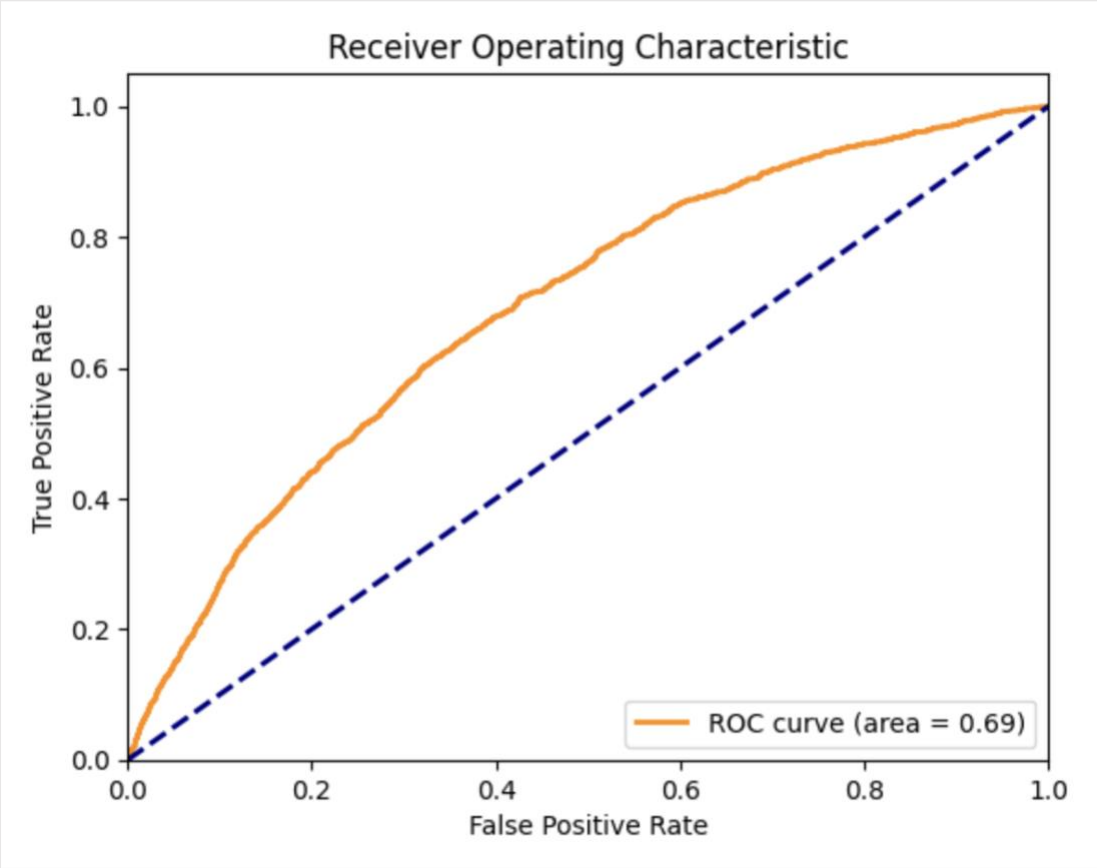
```
xgb_tuned = XGBClassifier(random_state = 1, scale_pos_weight = 1, learning_rate=0.02, n_estimators=600,
                          objective='binary:logistic', nthread=1)
xgb_tuned.fit(train_X, train_y)
```

6.2.9.2 *Model performance*

This model has the same F1-score of 0.2 and accuracy of 0.86 as the base XGBoost model. However, this has a slightly higher ROC-AUC score of 0.69.

Model	ROC-AUC score	F1_score	Accuracy
XGBoost_tuned	0.69	0.20	0.86

Figure 34 ROC-AUC curve - XGBoost tuned



The recall score shows that the XGboost tuned model can capture only 17% of the default while the precision tells us that when the model says it is a default, the model is right 23% of the time.

	Prediction	
Actual	not default	default
not default	13997	937
default	1307	275

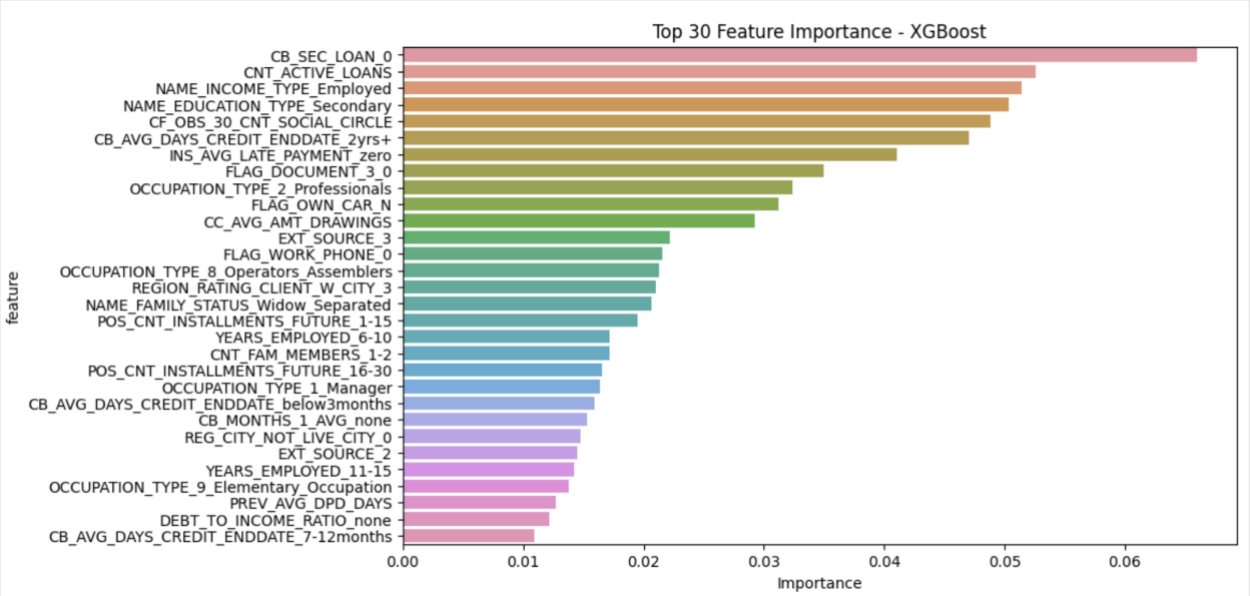
Recall = True Positive / (True Positive + False Negative)
= 275 / (275 + 1307)
= 0.17

Precision = True Positive / (True Positive + True Negative)
= 275 / (275 +937)
= 0.23

6.2.9.3 Feature importance

Comparing the feature importance of the base model and tuned model for XGboost, the XGboost has the same feature importance and importance scores with the base model.

Figure 35 Top 30 Important Features - XGBoost tuned



7 PERFORMANCE EVALUATION

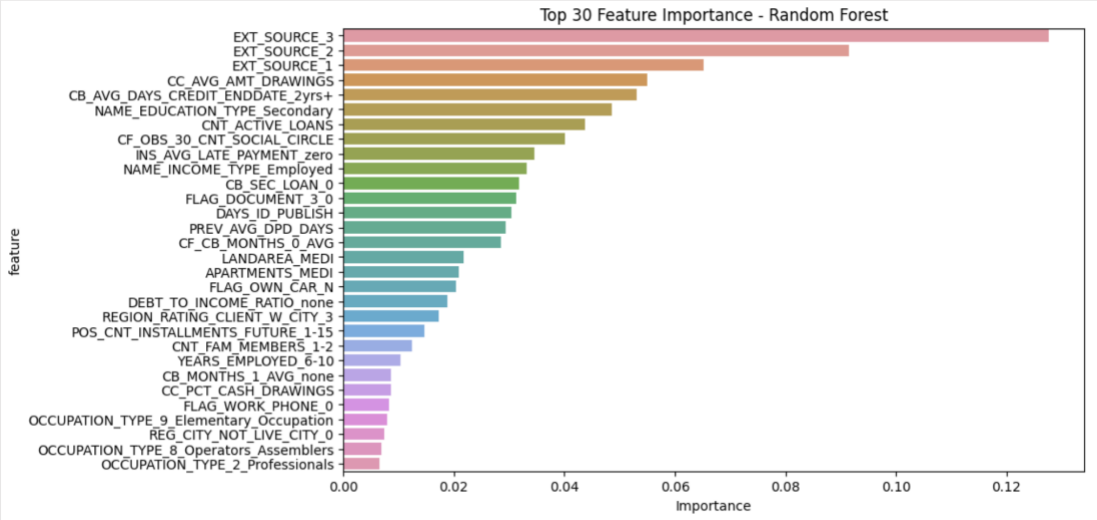
7.1 Summary of results

Model	ROC-AUC	F1_score	Accuracy
Random_Forest	0.69	0.26	0.77
SGD	0.69	0.26	0.68
Logistic_Regression_Tuned	0.69	0.26	0.64
Random_Forest_tuned	0.69	0.21	0.84
XGBoost_tuned	0.69	0.2	0.86
XGBoost	0.67	0.2	0.86
Decision_Tree	0.55	0.19	0.75
Decision_Tree_Tuned	0.53	0.2	0.74

7.2 Best model

In selecting the best model, we use ROC-AUC score as the primary metric because it gives us a more comprehensive view of model performance, especially our dataset highly imbalanced which could lead to misleading conclusions when we based it solely on accuracy. Next, we rank it based on the F1 score since it gives us an idea of how well the model balances between correctly identifying positive instances ("Default" in this case) and minimizing the number of false negatives. And lastly, we use accuracy as the third metric since we want to know regardless of the class, how accurate is our model. Given the above criteria, the best model we have is the Random Forest which has an ROC-AUC score of 0.69, F1-score of 0.26 and can accurately predict our data 77% of the time. This model can capture 41% of the default and 19% of the time the model is right in identifying a default.

Figure 36 Top 30 Features - Best Model



The most important features of the best model are the external credit score (EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3) followed by the average amount drawings from credit card (CC_AVG_AMT_DRAWINGS), remaining days before the credit ends in credit bureau is more than 2yrs (CB_AVG_DAYS_CREDIT_ENDDATE_2yrs+), and highest educational attainment is secondary education (NAME_EDUCATION_TYPE_Secondary).

8 CONCLUSION AND RECOMMENDATION

8.1 Recommendations

8.1.1 For the analytics team

Tailor the model to Canadian market

We built the model based on the dataset from Home Credit which is outside of Canada. Therefore, before cannot implement the model as is. We have to gather similar data of the features used in this model from the Canadian market, refit the model to the new data, evaluate and optimize.

Improve model scores.

We have also seen that the model has relatively low scores, so it must be improved first especially the recall and precision scores. Some ways to improve the model are by creating high-quality features that can better predict or has higher correlation to the target variable, and modifying the threshold in instead of classifying a default by 0.5.

Create models for new clients.

The analytics team can also continue to explore the use of alternative data for new clients in order to understand what are the key characteristics of a client to default in the absence of a credit score.

8.1.2 For the management team

Review regulatory laws about using alternative data.

In Canada, there are laws about the use of specific data like demographics in loan applications in order to avoid discrimination. Therefore, it is always best to review if the chosen alternative data to be used in the final model is allowed in Canada.

Explore the use of alternative data.

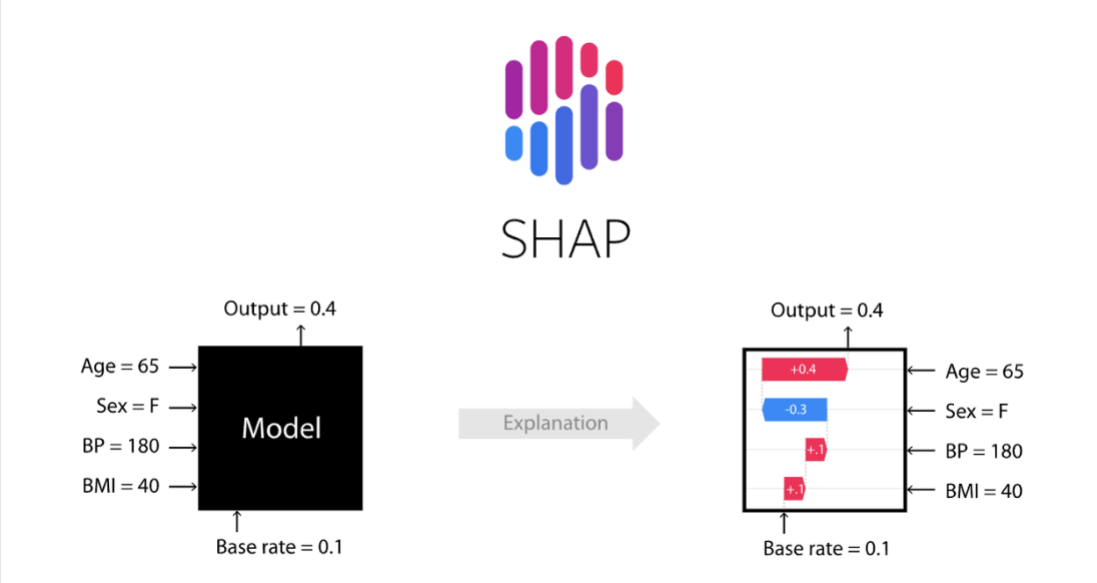
We have seen that aside from credit scores, there are several alternative data that can also predict the default of a client. With this, it is recommended to explore the use of alternative data to evaluate the client's 5Cs (capacity, character, collateral, capital, conditions). For example, the feature

INS_AVG_LATE_PAYMENT_0 talks about the client's payment behavior. It is also recommended to explore other alternative data outside this dataset like rental, and online behavior.

Use of explainable AI (e.g. SHAP)

We noticed that this type of problem and large dataset needs to be conducted using complex machine learning algorithms which means that the results will be harder to interpret. In the financial industry, prediction results need to be explainable so the use of an explainable AI can help interpret the results to the stakeholders and customers. One type of explainable AI is SHAP or Shapely Additive Explanations which is used to explain complex black box models. It computes and shows how a feature predicted the outcome as shown in the example below.

Figure 37 Explainable AI - SHAP



8.2 Conclusion

In conclusion, this proof of concept we used the alternative data from Home credit to predict underserved customers who has the capacity to repay their loans. We did this by creating several predictive models, where the best model as of now is a Random Forest with a ROC-AUC of 70% and is accurate 77% of the time. Aside from credit scores, we identified that the top 3 features to predict a default are the average amount drawings from credit card, remaining days before the credit ends in credit bureau is more than 2yrs, and highest educational attainment is secondary education.

Through our exploration, it's become evident that harnessing alternative data can be a transformative avenue to gain deeper insights into the financial capacities of our clients. This strategic approach can equip our company to make informed lending decisions and empower financial inclusion by offering credit opportunities to individuals who may have been overlooked by traditional methods.

9 APPENDIX

Table 14 Application_df_v1 - Final table to be analyzed for EDA

Table	Column	Description
application_train.csv	SK_ID_CURR	ID of loan in our sample
application_train.csv	TARGET	Target variable (1 - client with payment difficulties: he/she had late payment more than X days on at least one of the first Y installments of the loan in our sample, 0 - all other cases)
application_train.csv	NAME_CONTRACT_TYPE	Identification if loan is cash or revolving
application_train.csv	CODE_GENDER	Gender of the client
application_train.csv	FLAG_OWN_CAR	Flag if the client owns a car
application_train.csv	FLAG_OWN_REALTY	Flag if client owns a house or flat
application_train.csv	CNT_CHILDREN	Number of children the client has
application_train.csv	AMT_INCOME_TOTAL	Income of the client (Monthly)
application_train.csv	AMT_CREDIT	Credit amount of the loan (Includes the price of insurance if any)
application_train.csv	AMT_ANNUITY	Loan annuity
application_train.csv	AMT_GOODS_PRICE	For consumer loans it is the price of the goods for which the loan is given
application_train.csv	NAME_TYPE_SUITE	Who was accompanying client when he was applying for the loan
application_train.csv	NAME_INCOME_TYPE	Clients income type (businessman, working, maternity leave,Ö)
application_train.csv	NAME_EDUCATION_TYPE	Level of highest education the client achieved
application_train.csv	NAME_FAMILY_STATUS	Family status of the client
application_train.csv	NAME_HOUSING_TYPE	What is the housing situation of the client (renting, living with parents, ...)
application_train.csv	REGION_POPULATION_RELATIVE	Normalized population of region where client lives (higher number means the client lives in more populated region)
application_train.csv	DAYS_BIRTH	Client's age in days at the time of application
application_train.csv	DAYS_EMPLOYED	How many days before the application the person started current employment

application_train.csv	DAYS_REGISTRATION	How many days before the application did client change his registration
application_train.csv	DAYS_ID_PUBLISH	How many days before the application did client change the identity document with which he applied for the loan
application_train.csv	OWN_CAR_AGE	Age of client's car
application_train.csv	FLAG_MOBIL	Did client provide mobile phone (1=YES, 0=NO)
application_train.csv	FLAG_EMP_PHONE	Did client provide employer phone (1=YES, 0=NO)
application_train.csv	FLAG_WORK_PHONE	Did client provide work phone (1=YES, 0=NO)
application_train.csv	FLAG_CONT_MOBILE	Was mobile phone reachable (1=YES, 0=NO)
application_train.csv	FLAG_PHONE	Did client provide home phone (1=YES, 0=NO)
application_train.csv	FLAG_EMAIL	Did client provide email (1=YES, 0=NO)
application_train.csv	OCCUPATION_TYPE	What kind of occupation does the client have
application_train.csv	CNT_FAM_MEMBERS	How many family members does client have
application_train.csv	REGION_RATING_CLIENT	Our rating of the region where client lives (1,2,3)
application_train.csv	REGION_RATING_CLIENT_W_CITY	Our rating of the region where client lives with taking city into account (1,2,3)
application_train.csv	WEEKDAY_APPR_PROCESS_START	On which day of the week did the client apply for the loan
application_train.csv	HOUR_APPR_PROCESS_START	Approximately at what hour did the client apply for the loan
application_train.csv	REG_REGION_NOT_LIVE_REGION	Flag if client's permanent address does not match contact address (1=different, 0=same, at region level)
application_train.csv	REG_REGION_NOT_WORK_REGION	Flag if client's permanent address does not match work address (1=different, 0=same, at region level)
application_train.csv	LIVE_REGION_NOT_WORK_REGION	Flag if client's contact address does not match work address (1=different, 0=same, at region level)
application_train.csv	REG_CITY_NOT_LIVE_CITY	Flag if client's permanent address does not match contact address (1=different, 0=same, at city level)
application_train.csv	REG_CITY_NOT_WORK_CITY	Flag if client's permanent address does not match work address (1=different, 0=same, at city level)
application_train.csv	LIVE_CITY_NOT_WORK_CITY	Flag if client's contact address does not match work address (1=different, 0=same, at city level)
application_train.csv	ORGANIZATION_TYPE	Type of organization where client works
application_train.csv	EXT_SOURCE_1	Normalized score from external data source

application_train.csv	EXT_SOURCE_2	Normalized score from external data source
application_train.csv	EXT_SOURCE_3	Normalized score from external data source
application_train.csv	APARTMENTS_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	BASEMENTAREA_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	YEARS_BEGINEXPLUATATION_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	YEARS_BUILD_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	COMMONAREA_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	ELEVATORS_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	ENTRANCES_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

application_train.csv	FLOORSMAX_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	FLOORSMIN_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	LANDAREA_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	LIVINGAPARTMENTS_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	LIVINGAREA_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	NONLIVINGAPARTMENTS_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	NONLIVINGAREA_AVG	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	APARTMENTS_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of

		elevators, number of entrances, state of the building, number of floor
application_train.csv	BASEMENTAREA_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	YEARS_BEGINEXPLUATATION_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	YEARS_BUILD_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	COMMONAREA_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	ELEVATORS_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	ENTRANCES_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	FLOORSMAX_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

application_train.csv	FLOORSMIN_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	LANDAREA_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	LIVINGAPARTMENTS_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	LIVINGAREA_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	NONLIVINGAPARTMENTS_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	NONLIVINGAREA_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	APARTMENTS_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	BASEMENTAREA_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of

		elevators, number of entrances, state of the building, number of floor
application_train.csv	YEARS_BEGINEXPLUATATION_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	YEARS_BUILD_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	COMMONAREA_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	ELEVATORS_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	ENTRANCES_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	FLOORSMAX_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	FLOORSMIN_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

application_train.csv	LANDAREA_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	LIVINGAPARTMENTS_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	LIVINGAREA_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	NONLIVINGAPARTMENTS_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	NONLIVINGAREA_MEDI	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	FONDKAPREMONT_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	HOUSETYPE_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	TOTALAREA_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of

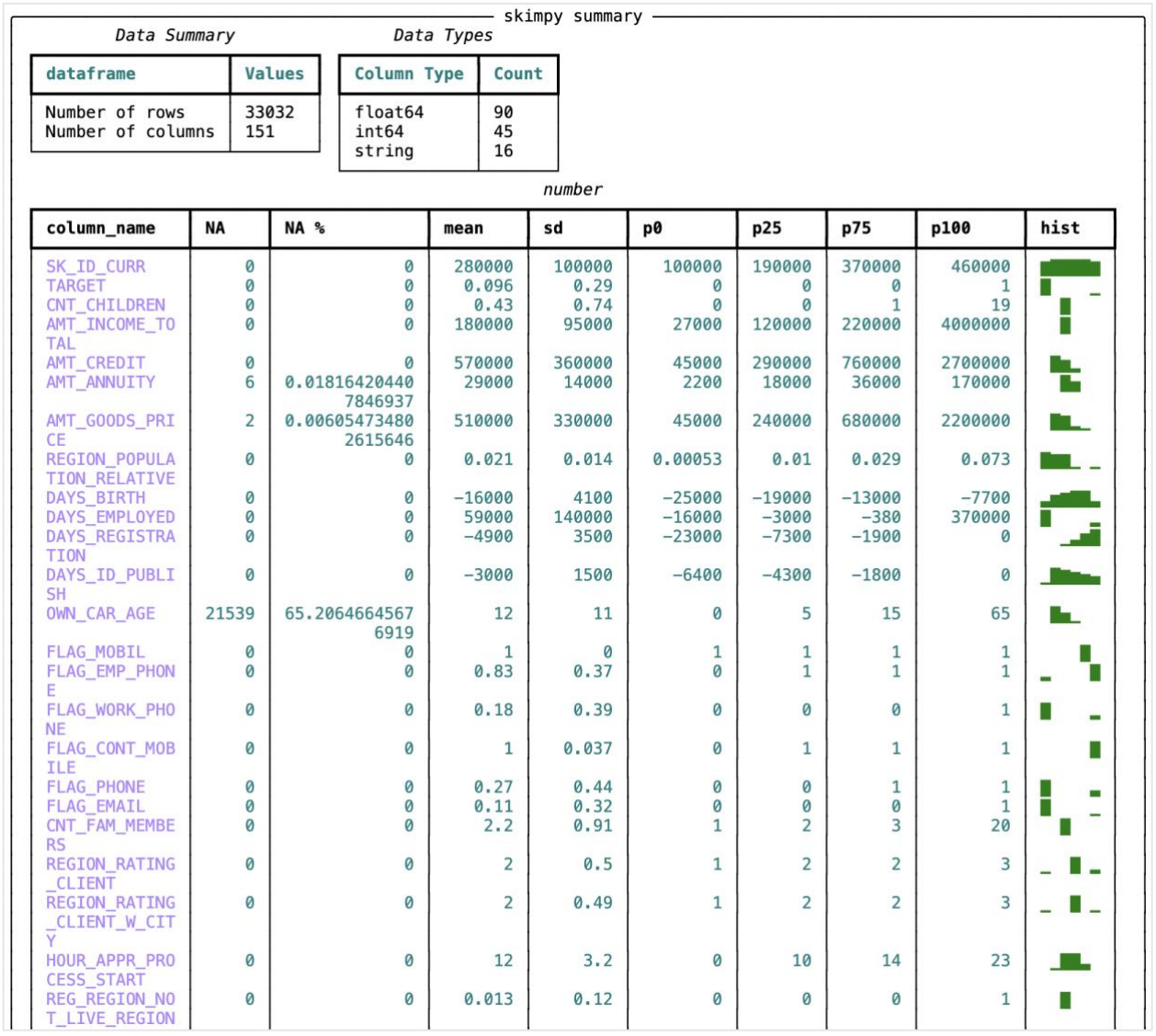
		elevators, number of entrances, state of the building, number of floor
application_train.csv	WALLSMATERIAL_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	EMERGENCYSTATE_MODE	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
application_train.csv	OBS_30_CNT_SOCIAL_CIRCLE	How many observation of client's social surroundings with observable 30 DPD (days past due) default
application_train.csv	DEF_30_CNT_SOCIAL_CIRCLE	How many observation of client's social surroundings defaulted on 30 DPD (days past due)
application_train.csv	OBS_60_CNT_SOCIAL_CIRCLE	How many observation of client's social surroundings with observable 60 DPD (days past due) default
application_train.csv	DEF_60_CNT_SOCIAL_CIRCLE	How many observation of client's social surroundings defaulted on 60 (days past due) DPD
application_train.csv	DAYS_LAST_PHONE_CHANGE	How many days before application did client change phone
application_train.csv	FLAG_DOCUMENT_2	Did client provide document 2
application_train.csv	FLAG_DOCUMENT_3	Did client provide document 3
application_train.csv	FLAG_DOCUMENT_4	Did client provide document 4
application_train.csv	FLAG_DOCUMENT_5	Did client provide document 5
application_train.csv	FLAG_DOCUMENT_6	Did client provide document 6
application_train.csv	FLAG_DOCUMENT_7	Did client provide document 7
application_train.csv	FLAG_DOCUMENT_8	Did client provide document 8
application_train.csv	FLAG_DOCUMENT_9	Did client provide document 9
application_train.csv	FLAG_DOCUMENT_10	Did client provide document 10
application_train.csv	FLAG_DOCUMENT_11	Did client provide document 11
application_train.csv	FLAG_DOCUMENT_12	Did client provide document 12
application_train.csv	FLAG_DOCUMENT_13	Did client provide document 13

application_train.csv	FLAG_DOCUMENT_14	Did client provide document 14
application_train.csv	FLAG_DOCUMENT_15	Did client provide document 15
application_train.csv	FLAG_DOCUMENT_16	Did client provide document 16
application_train.csv	FLAG_DOCUMENT_17	Did client provide document 17
application_train.csv	FLAG_DOCUMENT_18	Did client provide document 18
application_train.csv	FLAG_DOCUMENT_19	Did client provide document 19
application_train.csv	FLAG_DOCUMENT_20	Did client provide document 20
application_train.csv	FLAG_DOCUMENT_21	Did client provide document 21
application_train.csv	AMT_REQ_CREDIT_BUREAU_HOUR	Number of enquiries to Credit Bureau about the client one hour before application
application_train.csv	AMT_REQ_CREDIT_BUREAU_DAY	Number of enquiries to Credit Bureau about the client one day before application (excluding one hour before application)
application_train.csv	AMT_REQ_CREDIT_BUREAU_WEEK	Number of enquiries to Credit Bureau about the client one week before application (excluding one day before application)
application_train.csv	AMT_REQ_CREDIT_BUREAU_MON	Number of enquiries to Credit Bureau about the client one month before application (excluding one week before application)
application_train.csv	AMT_REQ_CREDIT_BUREAU_QRT	Number of enquiries to Credit Bureau about the client 3 month before application (excluding one month before application)
application_train.csv	AMT_REQ_CREDIT_BUREAU_YEAR	Number of enquiries to Credit Bureau about the client one day year (excluding last 3 months before application)
bureau.csv	CB_TOTAL_LOANS	Number of total loans in credit bureau for each customer.
bureau.csv	CB_PCT_ACTIVE_LOANS	Percentage of active loans of each customer in credit bureau
bureau.csv	CB_SEC_LOAN	Number of secured loans in credit bureau. CREDIT_TYPE in ('Mortgage', 'Car loan', 'Real estate loan', 'Loan for the purchase of equipment', 'Loan for purchase of shares (margin lending)')
bureau.csv	CB_UNSEC_LOAN	Number of unsecured loans in credit bureau. CREDIT_TYPE in ('Consumer credit', 'Credit card','Microloan', 'Loan for working capital replenishment', 'Loan for business development', 'Unknown type of loan', 'Another type of loan', 'Cash loan (non-earmarked)', 'Mobile operator loan', 'Interbank credit')

bureau.csv	CB_MIN_DAYS_CREDIT	Minimum number of days before current application did client apply for Credit Bureau credit. Time only relative to the application.
bureau.csv	CB_AVG_DAYS_CREDIT_ENDDATE	Average remaining duration of CB credit (in days) at the time of application in Home Credit. Time only relative to the application.
bureau.csv	CB_AVG_DAYS_ENDDATE_FACT	Average days since CB credit ended at the time of application in Home Credit (only for closed credit). Time only relative to the application.
bureau.csv	CB_TOTAL_CNT_CREDIT_PROLONG	Total number of times the Credit Bureau credit was prolonged.
bureau.csv	CB_AMT_CREDIT_SUM_OVERDUE	Total current amount overdue on Credit Bureau credit.
bureau.csv	CB_PCT_DEBT	CB_AMT_CREDIT_SUM_DEBT over CB_AMT_CREDIT_SUM
bureau.csv	CB_AMT_CREDIT_SUM_DEBT	Total current debt on Credit Bureau credit.
bureau_balance.csv	CB_MONTHS_0_AVG	Average number of times the client has paid on time. Count(distinct SK_ID_BUREAU) where status is 0.
bureau_balance.csv	CB_MONTHS_1_AVG	Average number of times the client has delayed payment within 30 days. Count(distinct SK_ID_BUREAU) where status is 1.
bureau_balance.csv	CB_MONTHS_2_AVG	Average number of times the client has delayed payment within 31-60 days. Count(distinct SK_ID_BUREAU) where status is 2.
bureau_balance.csv	CB_MONTHS_3_AVG	Average number of times the client has delayed payment within 61-90 days. Count(distinct SK_ID_BUREAU) where status is 3.
bureau_balance.csv	CB_MONTHS_4_AVG	Average number of times the client has delayed payment within 91-120 days. Count(distinct SK_ID_BUREAU) where status is 4.
bureau_balance.csv	CB_MONTHS_5_AVG	Average number of times the client has delayed payment for more than 120 days, or sold or written off. Count(distinct SK_ID_BUREAU) where status is 5.
previous_application.csv	CNT_ACCOUNT_TYPES	Count of distinct NAME_CONTRACT_TYPE
previous_application.csv	CNT_ACTIVE_LOANS	Count of unique active loans

previous_application.csv	PREV_AVG_DPD_DAYS	Average number of DPD (days past due) during the month of previous credit.
previous_application.csv	PREV_AVG_CNT_DPD	Average count of times the customer has days past due.
previous_application.csv	PREV_SUM_AMT_ANNUITY	Total annuity amount of the current active accounts in Home Credit
previous_application.csv	PREV_SUM_AMT_CREDIT	Total final credit amount on the previous application.
Credit_card_balance.csv	PREV_DEBT_TO_CREDIT_RATIO	Total balance over total actual credit limit. SUM(AMT_BALANCE) / SUM(AMT_CREDIT_LIMIT_ACTUAL)
credit_card_balance.csv	CC_PCT_CASH_DRAWINGS	Total amount drawing at ATM over total amount drawings (POS,ATM, other current) during the month of the previous credit. SUM(AMT_DRAWINGS_ATM_CURRENT) / SUM(AMT_DRAWINGS_CURRENT)
credit_card_balance.csv	CC_AVG_AMT_DRAWINGS	Total amount drawings over the total number of drawings. SUM(AMT_DRAWINGS_CURRENT) / SUM(CNT_DRAWINGS_CURRENT)
POS_CASH_balance.csv	POS_CNT_INSTALLMENTS_FUTURE	Number of remaining installments left to pay on the previous credit
installments_payments.csv	INS_AVG_LATE_PAYMENT	Average number of days the payment was delayed. DAYS_ENTRY_PAYMENT - DAYS_INSTALMENT.
installments_payments.csv	INS_AVG_LESS_PAYMENT	Average amount of less payment made. AMT_INSTALMENT - AMT_PAYMENT
previous_application.csv	APPROVAL_RATE	Total count of SK_ID_PREV with NAME_CONTRACT_STATUS IN ('Approved', 'Unused offer') over total count of SK_ID_PREV per customer.

Figure 38 Skimpy summary - application_df



skimpy summary

Data Summary

dataframe	Values
Number of rows	33032
Number of columns	151

Data Types

Column Type	Count
float64	90
int64	45
string	16

number

column_name	NA	NA %	mean	sd	p0	p25	p75	p100	hist
SK_ID_CURR	0	0	280000	100000	100000	190000	370000	460000	
TARGET	0	0	0.096	0.29	0	0	0	1	
CNT_CHILDREN	0	0	0.43	0.74	0	0	1	19	
AMT_INCOME_TOTAL	0	0	180000	95000	27000	120000	220000	4000000	
AMT_CREDIT	0	0	570000	360000	45000	290000	760000	2700000	
AMT_ANNUITY	6	0.018164204407846937	29000	14000	2200	18000	36000	170000	
AMT_GOODS_PRICE	2	0.006054734802615646	510000	330000	45000	240000	680000	2200000	
REGION_POPULATION_RELATIVE	0	0	0.021	0.014	0.00053	0.01	0.029	0.073	
DAYS_BIRTH	0	0	-16000	4100	-25000	-19000	-13000	-7700	
DAYS_EMPLOYED	0	0	59000	140000	-16000	-3000	-380	370000	
DAYS_REGISTRATION	0	0	-4900	3500	-23000	-7300	-1900	0	
DAYS_ID_PUBLISH	0	0	-3000	1500	-6400	-4300	-1800	0	
OWN_CAR_AGE	21539	65.20646645676919	12	11	0	5	15	65	
FLAG_MOBIL	0	0	1	0	1	1	1	1	
FLAG_EMP_PHONE	0	0	0.83	0.37	0	1	1	1	
FLAG_WORK_PHONE	0	0	0.18	0.39	0	0	0	1	
FLAG_CONT_MOBILE	0	0	1	0.037	0	1	1	1	
FLAG_PHONE	0	0	0.27	0.44	0	0	1	1	
FLAG_EMAIL	0	0	0.11	0.32	0	0	0	1	
CNT_FAM_MEMBERS	0	0	2.2	0.91	1	2	3	20	
REGION_RATING_CLIENT	0	0	2	0.5	1	2	2	3	
REGION_RATING_CLIENT_W_CITY	0	0	2	0.49	1	2	2	3	
HOUR_APPR_PROCESS_START	0	0	12	3.2	0	10	14	23	
REG_REGION_NOT_LIVE_REGION	0	0	0.013	0.12	0	0	0	1	

LIVE_REGION_NOT_WORK_REGION	0	0	0.04	0.19	0	0	0	1	
REG_CITY_NOT_LIVE_CITY	0	0	0.075	0.26	0	0	0	1	
REG_CITY_NOT_WORK_CITY	0	0	0.23	0.42	0	0	0	1	
LIVE_CITY_NOT_WORK_CITY	0	0	0.18	0.39	0	0	0	1	
EXT_SOURCE_1	16668	50.46015984499879	0.5	0.21	0.015	0.34	0.67	0.94	
EXT_SOURCE_2	2	0.006054734802615646	0.51	0.19	1e-05	0.39	0.66	0.85	
EXT_SOURCE_3	1997	6.045652700411722	0.48	0.2	0.00053	0.33	0.64	0.88	
APARTMENTS_AVG	16853	51.02022281424074	0.12	0.11	0	0.057	0.15	1	
BASEMENTAREA_AVG	19337	58.54020343908937	0.087	0.08	0	0.044	0.11	1	
YEARS_BEGINEXPLUATATION_AVG	16174	48.964640348752724	0.98	0.053	0	0.98	0.99	1	
YEARS_BUILD_AVG	21862	66.184306127391162	0.75	0.11	0	0.69	0.82	1	
COMMONAREA_AVG	22993	69.608258658270777	0.044	0.072	0	0.0079	0.052	1	
ELEVATORS_AVG	17709	53.611649309760236	0.077	0.13	0	0	0.12	1	
ENTRANCES_AVG	16714	50.599418745458955	0.15	0.097	0	0.069	0.21	1	
FLOORSMAX_AVG	16520	50.012109469605230	0.22	0.14	0	0.17	0.33	1	
FLOORSMIN_AVG	22333	67.610196173407600	0.23	0.16	0	0.083	0.38	1	
LANDAREA_AVG	19673	59.557398885928800	0.066	0.081	0	0.019	0.085	1	
LIVINGAPARTMENTS_AVG	22526	68.194478081860010	0.1	0.093	0	0.05	0.12	1	
LIVINGAREA_AVG	16640	50.375393557762170	0.11	0.11	0	0.044	0.13	1	
NONLIVINGAPARTMENTS_AVG	22882	69.272220876725600	0.0097	0.056	0	0	0.0039	1	
NONLIVINGAREA_AVG	18306	55.418987648341000	0.027	0.064	0	0	0.027	1	
APARTMENTS_MODE	16853	51.02022281424074	0.11	0.11	0	0.05	0.14	1	
BASEMENTAREA_MODE	19337	58.54020343908937	0.086	0.081	0	0.041	0.11	1	
YEARS_BEGINEXPLUATATION_MODE	16174	48.964640348752724	0.98	0.057	0	0.98	0.99	1	
YEARS_BUILD_MODE	21862	66.184306127391162	0.76	0.11	0	0.69	0.82	1	
COMMONAREA_MODE	22993	69.608258658270777	0.042	0.071	0	0.0075	0.049	1	

ENTRANCES_MODE	16714	50.5994187454 5895	0.14	0.098	0	0.069	0.21	1	
FLOORSMAX_MODE	16520	50.0121094696 0523	0.22	0.14	0	0.17	0.33	1	
FLOORSMIN_MODE	22333	67.6101961734 076	0.23	0.16	0	0.083	0.38	1	
LANDAREA_MODE	19673	59.5573988859 288	0.064	0.082	0	0.016	0.083	1	
LIVINGAPARTMENTS_MODE	22526	68.1944780818 6001	0.1	0.097	0	0.053	0.13	1	
LIVINGAREA_MODE	16640	50.3753935577 6217	0.1	0.11	0	0.042	0.12	1	
NONLIVINGAPARTMENTS_MODE	22882	69.2722208767 256	0.0089	0.054	0	0	0.0039	1	
NONLIVINGAREA_MODE	18306	55.4189876483 41	0.026	0.065	0	0	0.023	1	
APARTMENTS_MEDI	16853	51.0202228142 4074	0.12	0.11	0	0.057	0.15	1	
BASEMENTAREA_MEDI	19337	58.5402034390 8937	0.087	0.08	0	0.044	0.11	1	
YEARS_BEGINEXPLUATATION_MEDI	16174	48.9646403487 52724	0.98	0.053	0	0.98	0.99	1	
YEARS_BUILD_MEDI	21862	66.1843061273 9162	0.75	0.11	0	0.69	0.82	1	
COMMONAREA_MEDI	22993	69.6082586582 7077	0.044	0.072	0	0.0079	0.051	1	
ELEVATORS_MEDI	17709	53.6116493097 6023	0.076	0.13	0	0	0.12	1	
ENTRANCES_MEDI	16714	50.5994187454 5895	0.15	0.098	0	0.069	0.21	1	
FLOORSMAX_MEDI	16520	50.0121094696 0523	0.22	0.14	0	0.17	0.33	1	
FLOORSMIN_MEDI	22333	67.6101961734 076	0.23	0.16	0	0.083	0.38	1	
LANDAREA_MEDI	19673	59.5573988859 288	0.066	0.083	0	0.018	0.086	1	
LIVINGAPARTMENTS_MEDI	22526	68.1944780818 6001	0.1	0.093	0	0.051	0.12	1	
LIVINGAREA_MEDI	16640	50.3753935577 6217	0.11	0.11	0	0.045	0.13	1	
NONLIVINGAPARTMENTS_MEDI	22882	69.2722208767 256	0.0095	0.056	0	0	0.0039	1	
NONLIVINGAREA_MEDI	18306	55.4189876483 41	0.027	0.065	0	0	0.026	1	
TOTALAREA_MODE	16002	48.4439331557 2778	0.1	0.1	0	0.041	0.12	1	
OBS_30_CNT_SOCIAL_CIRCLE	3	0.00908210220 3923468	1.5	2.4	0	0	2	29	
DEF_30_CNT_SOCIAL_CIRCLE	3	0.00908210220 3923468	0.16	0.46	0	0	0	5	
OBS_60_CNT_SOCIAL_CIRCLE	3	0.00908210220 3923468	1.5	2.4	0	0	2	29	
DEF_60_CNT_SOCIAL_CIRCLE	3	0.00908210220 3923468	0.11	0.38	0	0	0	5	

DAYS_LAST_PHONE_CHANGE	0	0	-1100	850	-4200	-1800	-390	0
FLAG_DOCUMENT_2	0	0	0	0	0	0	0	0
FLAG_DOCUMENT_3	0	0	0.8	0.4	0	1	1	1
FLAG_DOCUMENT_4	0	0	0.00012	0.011	0	0	0	1
FLAG_DOCUMENT_5	0	0	0.013	0.11	0	0	0	1
FLAG_DOCUMENT_6	0	0	0.078	0.27	0	0	0	1
FLAG_DOCUMENT_7	0	0	3e-05	0.0055	0	0	0	1
FLAG_DOCUMENT_8	0	0	0.09	0.29	0	0	0	1
FLAG_DOCUMENT_9	0	0	0.0036	0.06	0	0	0	1
FLAG_DOCUMENT_10	0	0	0	0	0	0	0	0
FLAG_DOCUMENT_11	0	0	0.00073	0.027	0	0	0	1
FLAG_DOCUMENT_12	0	0	0	0	0	0	0	0
FLAG_DOCUMENT_13	0	0	0.0013	0.036	0	0	0	1
FLAG_DOCUMENT_14	0	0	0.0011	0.033	0	0	0	1
FLAG_DOCUMENT_15	0	0	0.00036	0.019	0	0	0	1
FLAG_DOCUMENT_16	0	0	0.0032	0.056	0	0	0	1
FLAG_DOCUMENT_17	0	0	6.1e-05	0.0078	0	0	0	1
FLAG_DOCUMENT_18	0	0	0.0025	0.05	0	0	0	1
FLAG_DOCUMENT_19	0	0	0.00012	0.011	0	0	0	1
FLAG_DOCUMENT_20	0	0	0.00021	0.015	0	0	0	1
FLAG_DOCUMENT_21	0	0	0	0	0	0	0	0
AMT_REQ_CREDIT_BUREAU_HOUR	0	0	0.0081	0.094	0	0	0	3
AMT_REQ_CREDIT_BUREAU_DAY	0	0	0.0085	0.12	0	0	0	6
AMT_REQ_CREDIT_BUREAU_WEEK	0	0	0.052	0.24	0	0	0	6
AMT_REQ_CREDIT_BUREAU_MON	0	0	0.3	0.8	0	0	0	27
AMT_REQ_CREDIT_BUREAU_QRT	0	0	0.32	0.64	0	0	0	7
AMT_REQ_CREDIT_BUREAU_YEAR	0	0	2.8	2	0	1	4	25

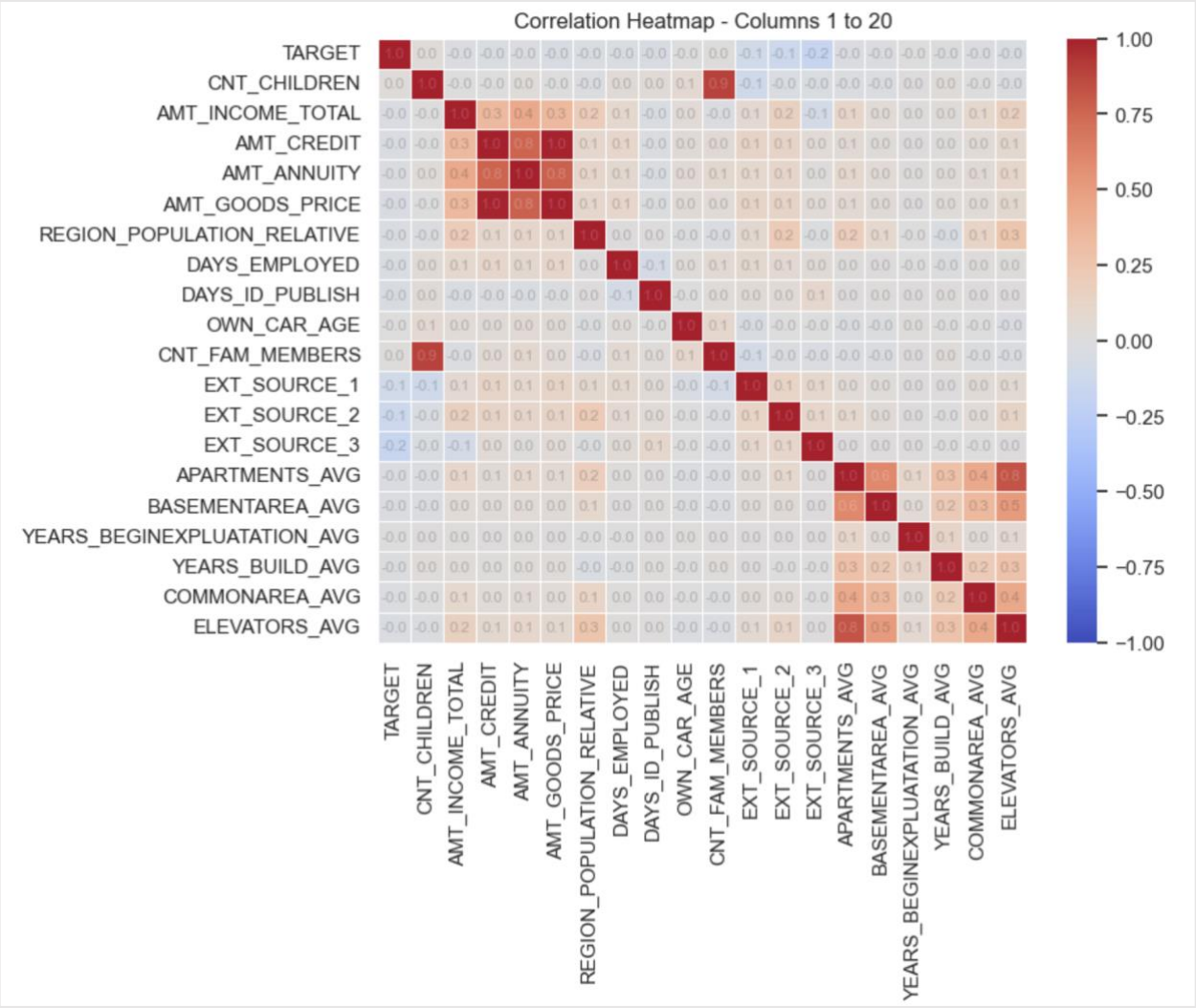
CB_TOTAL_LOANS	0	0	5.7	4.5	1	2	8	110	
CB_PCT_ACTIVE_LOANS	0	0	0.39	0.31	0	0.17	0.56	1	
CB_SEC_LOAN	0	0	0.14	0.44	0	0	0	8	
CB_UNSEC_LOAN	0	0	5.6	4.5	0	2	8	110	
CB_MIN_DAYS_CREDIT	0	0	-1800	840	-2900	-2600	-1100	-2	
CB_AVG_DAYS_CREDIT_ENDDATE	0	0	2600	5800	0	65	1300	31000	
CB_AVG_DAYS_ENDDATE_FACT	0	0	-850	580	-8400	-1200	-420	0	
CB_TOTAL_CNT_CREDIT_PROLONG	0	0	0.045	0.27	0	0	0	9	
CB_AMT_CREDIT_SUM_OVERDUE	0	0	57	2200	0	0	0	270000	
CB_PCT_DEBT	0	0	0.47	0.38	-3.9	0	0.81	7.8	
CB_AMT_CREDIT_SUM_DEBT	0	0	640000	1400000	-2000000	0	690000	37000000	
CB_MONTHS_0_AVG	0	0	9.4	6.9	0	5	12	90	
CB_MONTHS_1_AVG	0	0	0.39	1.1	0	0	0.33	29	
CB_MONTHS_2_AVG	0	0	0.039	0.22	0	0	0	9	
CB_MONTHS_3_AVG	0	0	0.016	0.13	0	0	0	6.3	
CB_MONTHS_4_AVG	0	0	0.011	0.11	0	0	0	8	
CB_MONTHS_5_AVG	0	0	0.11	1.4	0	0	0	65	
CNT_ACCOUNT_TYPES	0	0	1.2	0.42	1	1	1	3	
CNT_ACTIVE_LOANS	0	0	1.2	0.51	1	1	1	6	
PREV_AVG_DPD_DAYS	0	0	190	3000	0	0	0	110000	
PREV_AVG_CNT_DPD	0	0	0.85	4.1	0	0	0	87	
PREV_SUM_AMT_ANNUITY	0	0	19000	18000	0	6800	26000	230000	
PREV_SUM_AMT_CREDIT	0	0	360000	410000	0	89000	470000	4300000	
PREV_DEBT_TO_CREDIT_RATIO	19862	60.12957132477597	0.63	0.44	0	0.032	0.99	12	
CC_PCT_CASH_DRAWINGS	26872	81.35141680790438	0.39	0.46	0	0	1	1	
CC_AVG_AMT_DRAWINGS	26899	81.43315572777912	10000	25000	4.5	1600	9500	680000	
POS_CNT_INSTALLMENTS_FUTURE	0	0	6.4	11	0	0	8	100	
INS_AVG_LATE_PAYMENT	0	0	2.9	18	0	0	2.2	2200	
INS_AVG_LATE_PAYMENT	0	0	2.9	18	0	0	2.2	2200	
INS_AVG_LESS_PAYMENT	0	0	930	3200	0	0	390	99000	

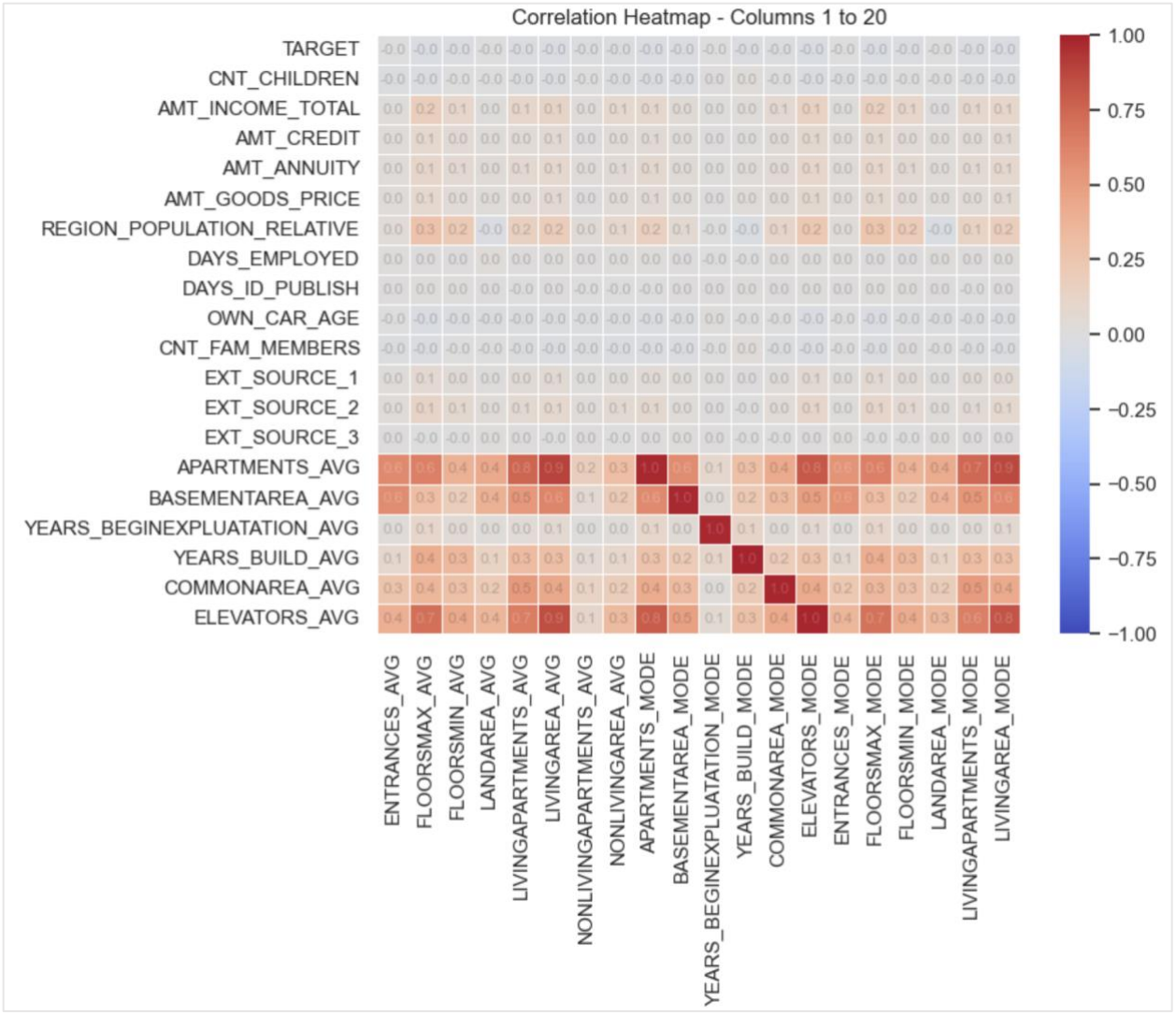
string

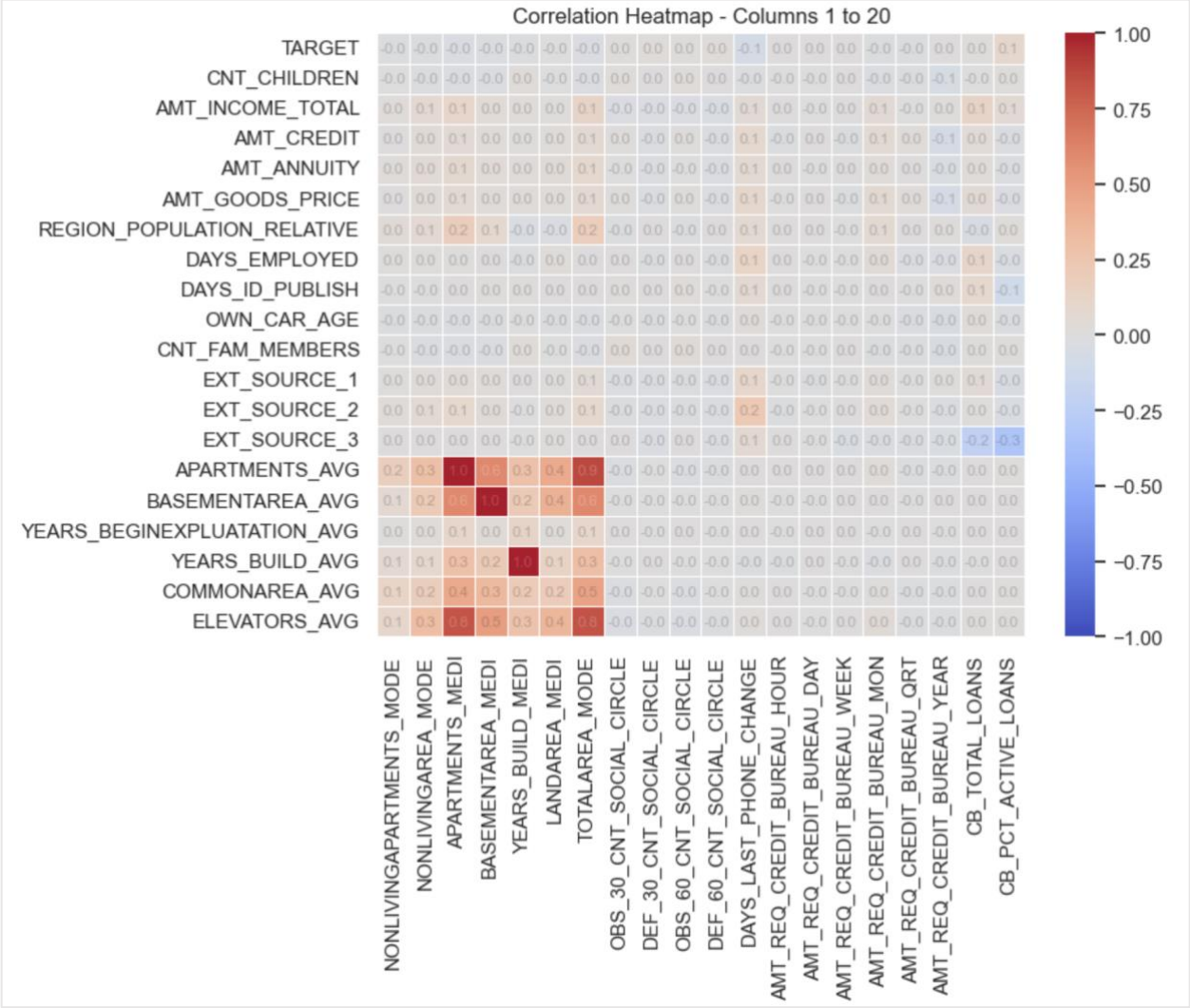
column_name	NA	NA %	words per row	total words
NAME_CONTRACT_TYPE	0	0	2	66064
CODE_GENDER	0	0	2	66064
FLAG_OWN_CAR	0	0	2	66064
FLAG_OWN_REALTY	0	0	2	66064
NAME_TYPE_SUITE	31	0.09384838944054251	2	66064
NAME_INCOME_TYPE	0	0	2	66064
NAME_EDUCATION_TYPE	0	0	2	66064
NAME_FAMILY_STATUS	0	0	2	66064
NAME_HOUSING_TYPE	0	0	2	66064
OCCUPATION_TYPE	9789	29.634899491402276	2	66064
WEEKDAY_APPR_PROCESS_START	0	0	2	66064
ORGANIZATION_TYPE	0	0	2	66064
FONDKAPREMONT_MODE	22518	68.17025914264956	2	66064
HOUSETYPE_MODE	16646	50.39355776217002	2	66064
WALLSMATERIAL_MODE	16878	51.09590699927343	2	66064
EMERGENCYSTATE_MODE	15724	47.6023250181642	2	66064

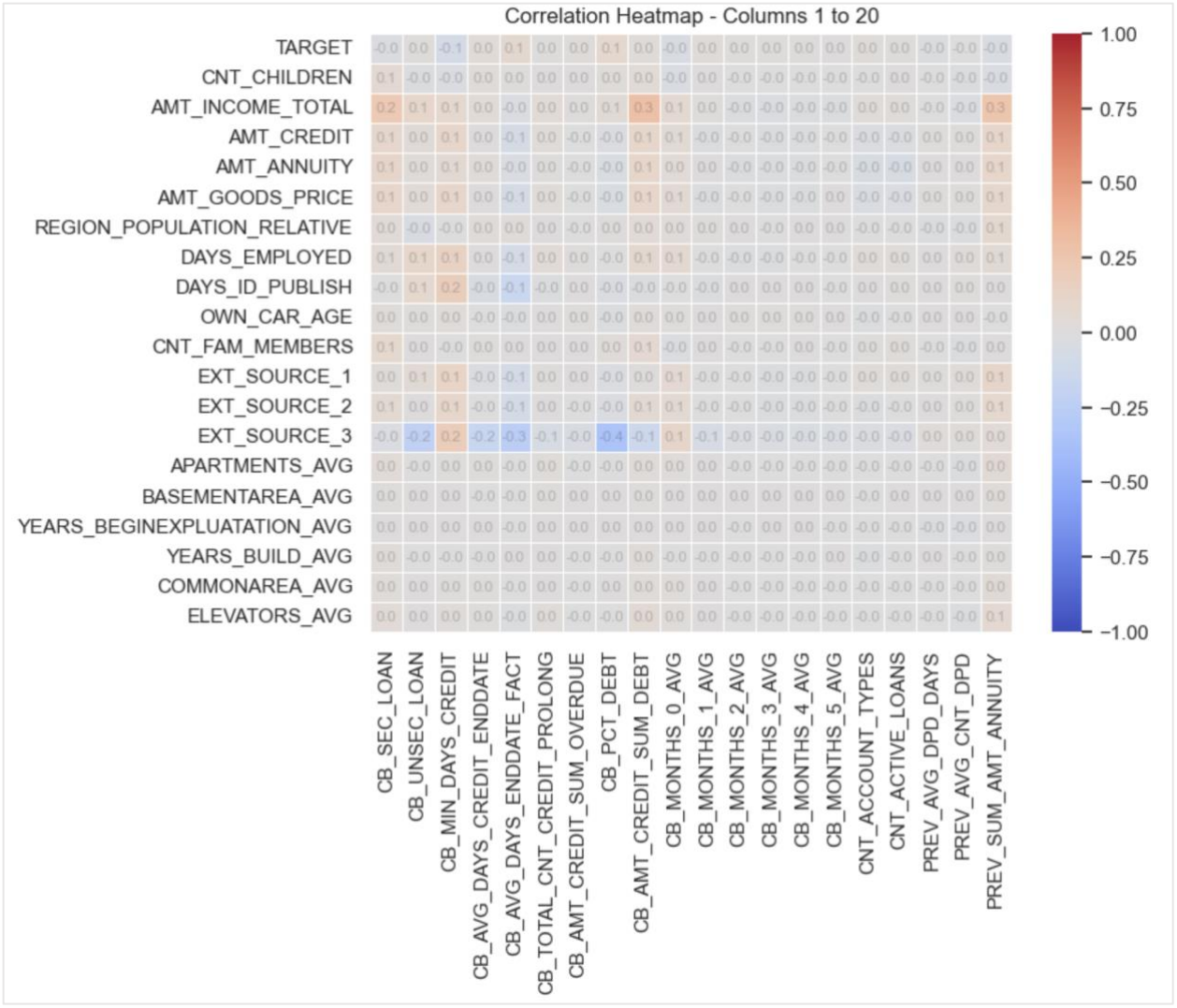
End

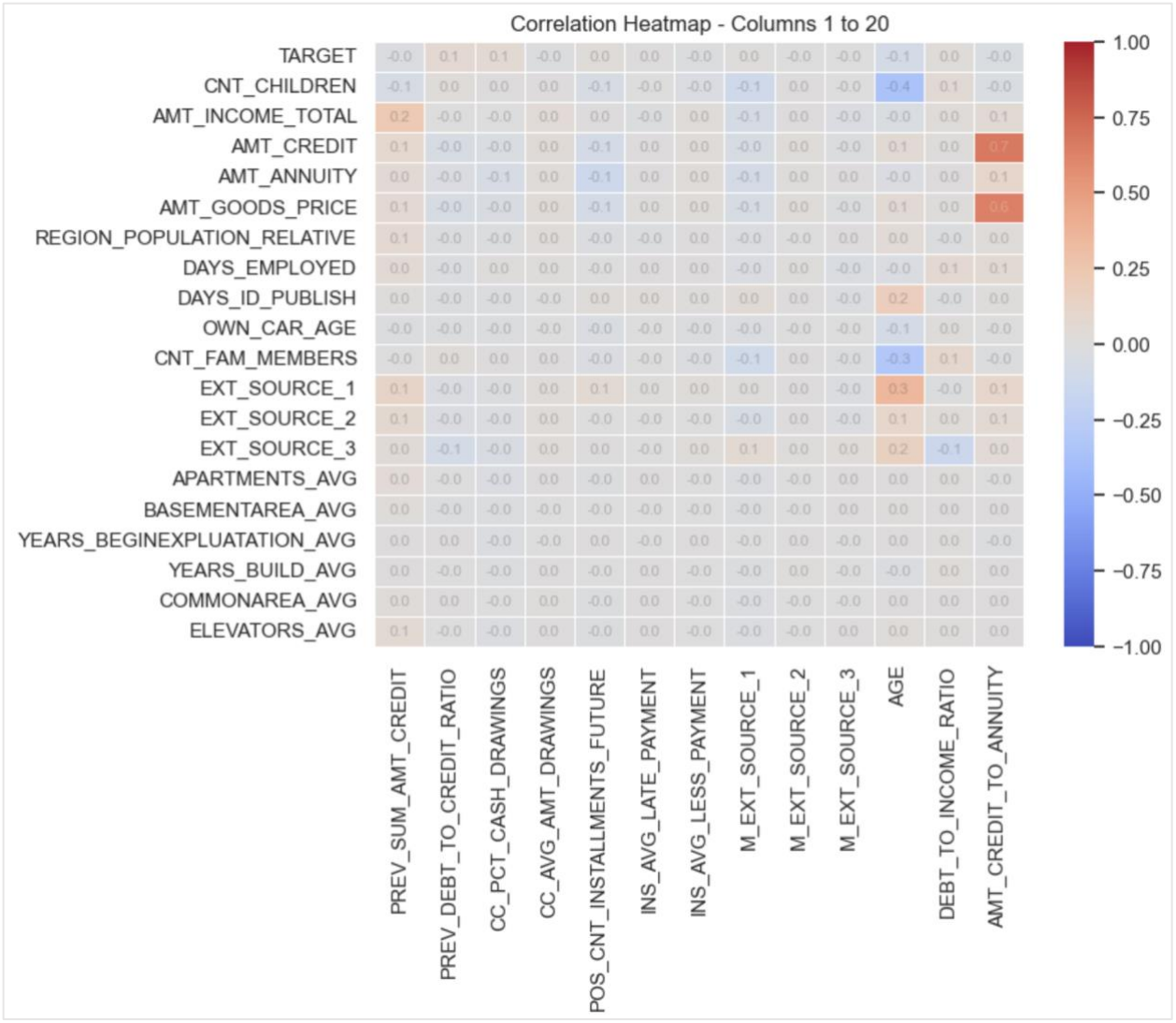
Figure 39 Correlation heatmap for numerical variables

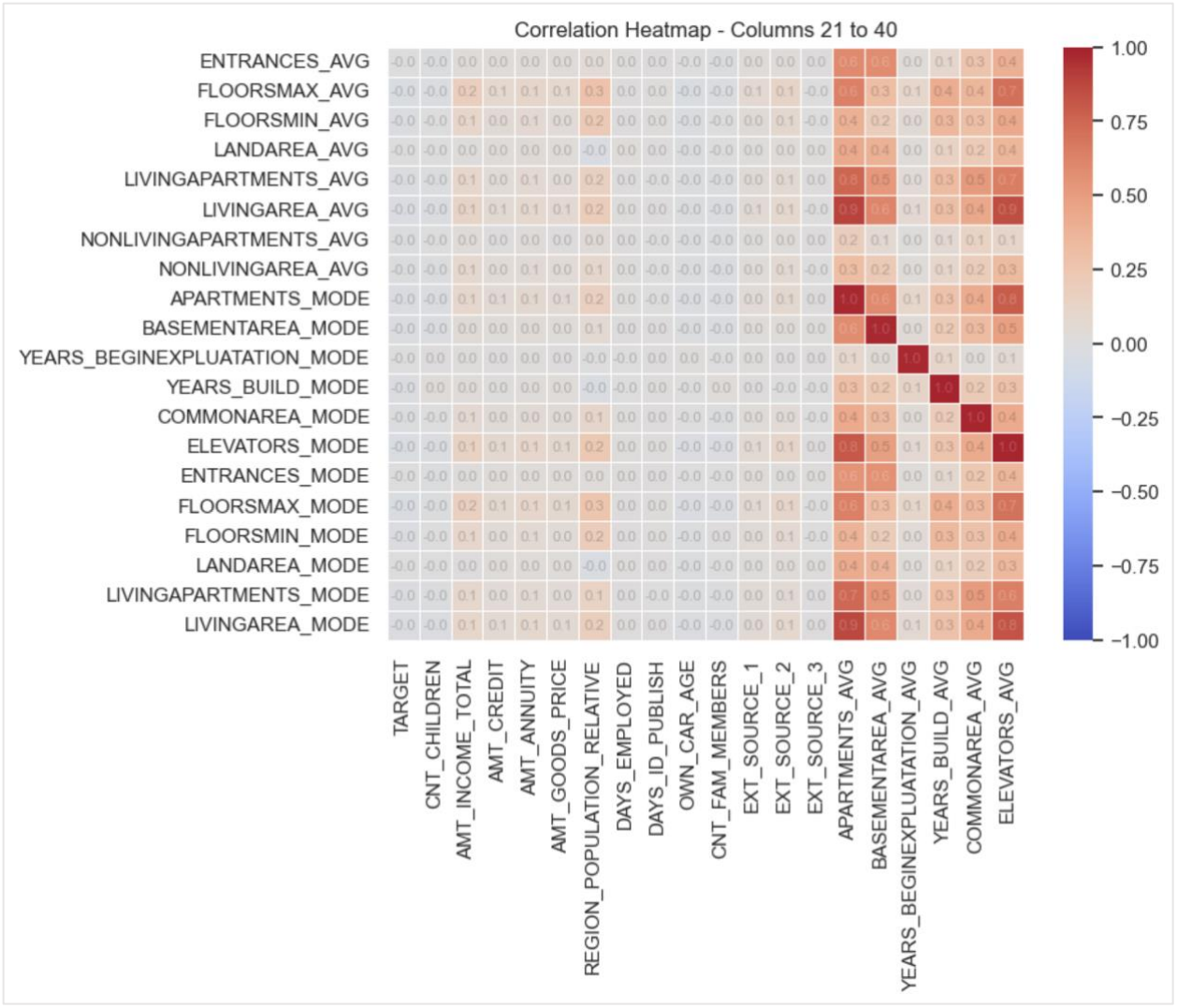


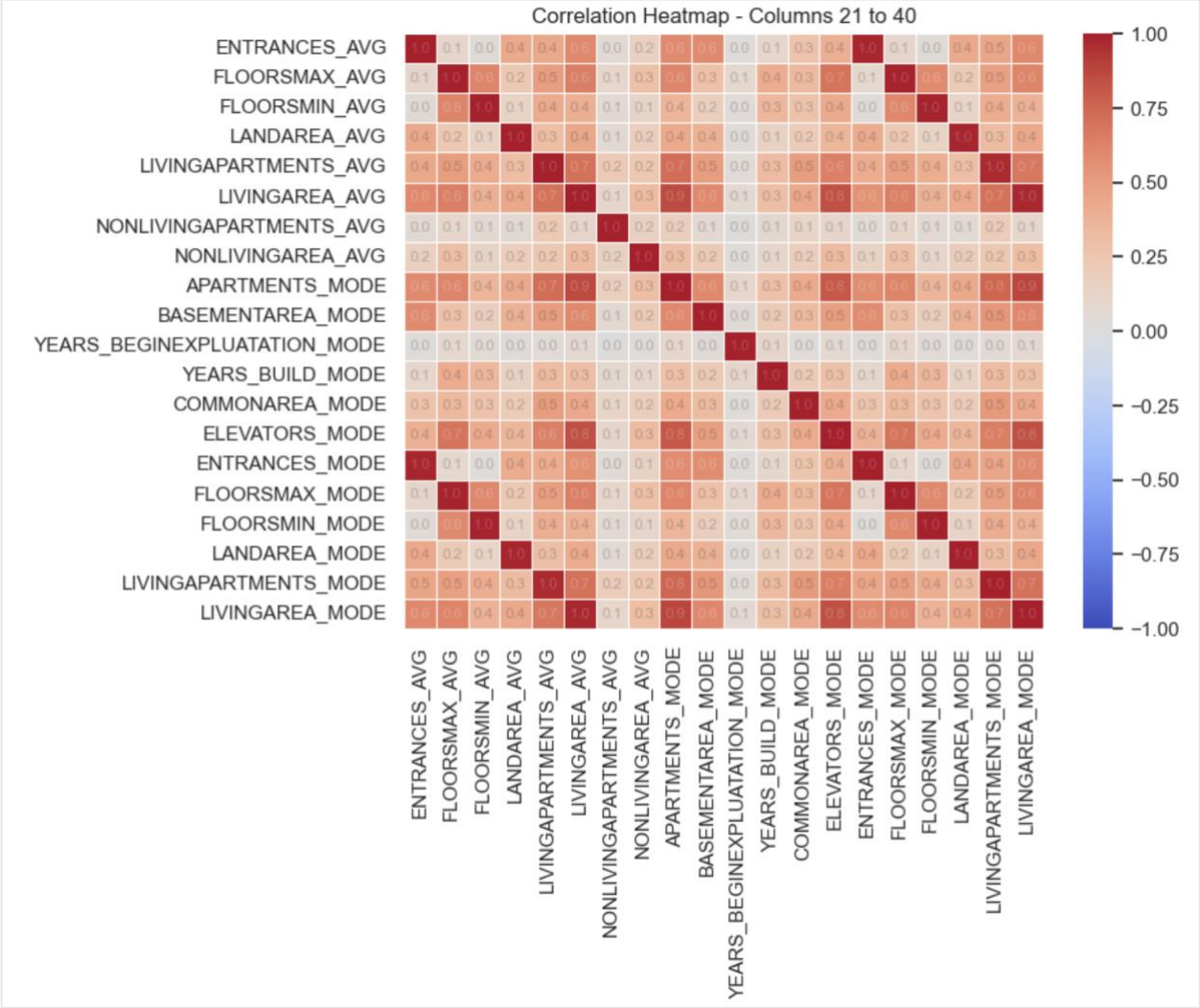

Correlation Heatmap - Columns 1 to 20
TARGET
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_EMPLOYED
DAYS_ID_PUBLISH
OWN_CAR_AGE
CNT_FAM_MEMBERS
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
APARTMENTS_AVG
BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG
YEARS_BUILD_AVG
COMMONAREA_AVG
ELEVATORS_AVG
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

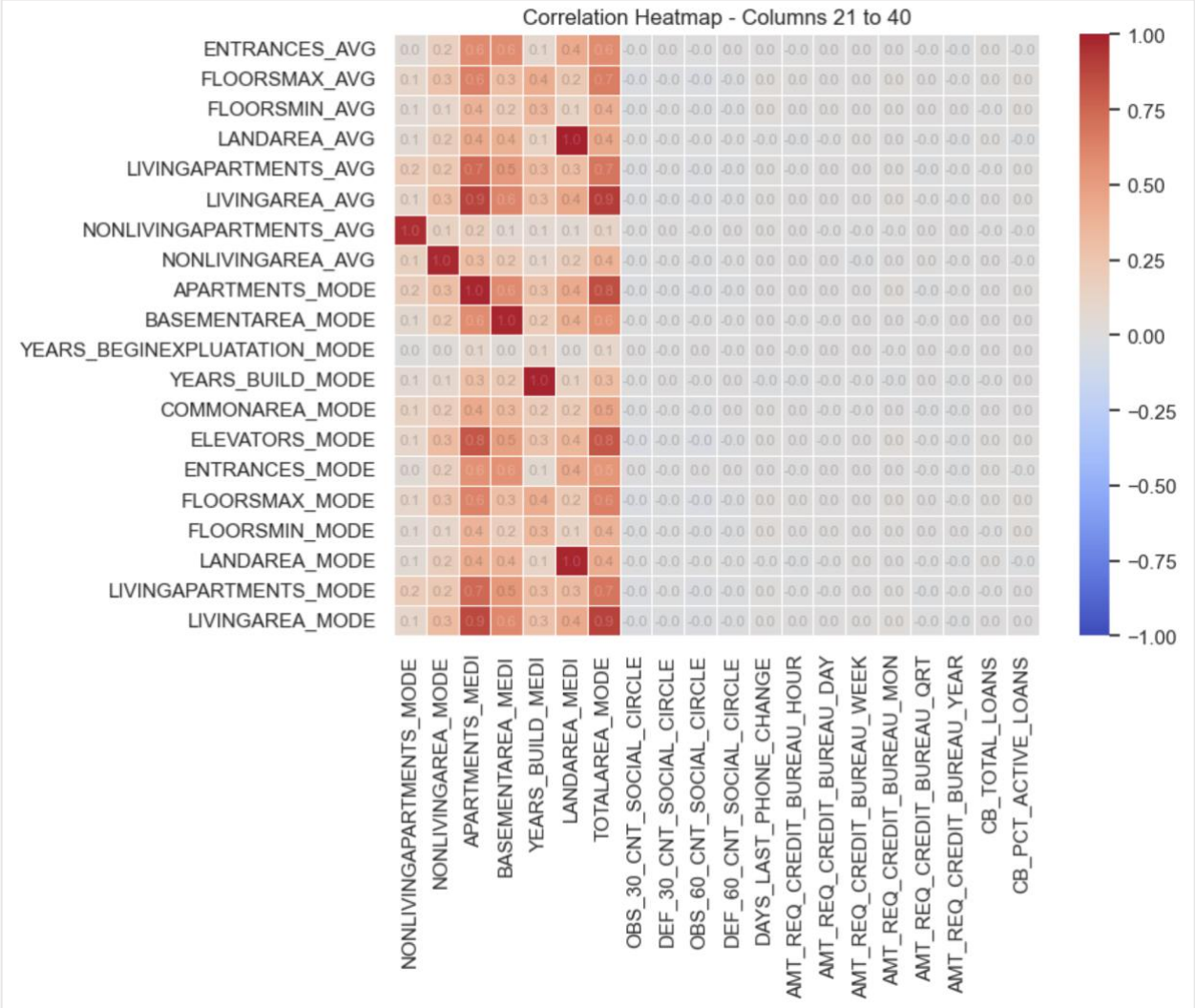

Correlation Heatmap - Columns 1 to 20
TARGET
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_EMPLOYED
DAYS_ID_PUBLISH
OWN_CAR_AGE
CNT_FAM_MEMBERS
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
APARTMENTS_AVG
BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG
YEARS_BUILD_AVG
COMMONAREA_AVG
ELEVATORS_AVG
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BUILD_MEDI
LANDAREA_MEDI
TOTALAREA_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
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
CB_TOTAL_LOANS
CB_PCT_ACTIVE_LOANS
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

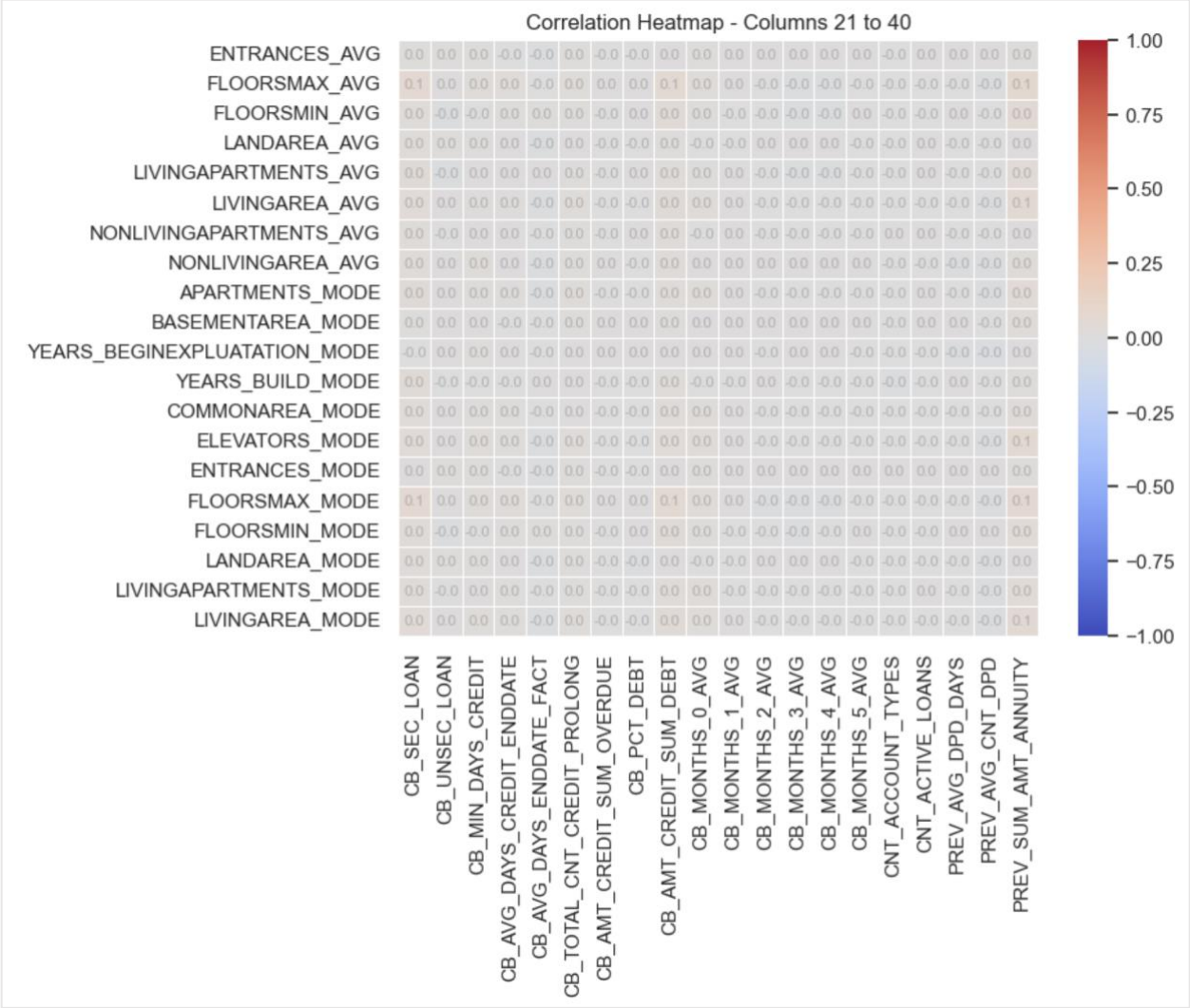

Correlation Heatmap - Columns 1 to 20
TARGET
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_EMPLOYED
DAYS_ID_PUBLISH
OWN_CAR_AGE
CNT_FAM_MEMBERS
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
APARTMENTS_AVG
BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG
YEARS_BUILD_AVG
COMMONAREA_AVG
ELEVATORS_AVG
CB_SEC_LOAN
CB_UNSEC_LOAN
CB_MIN_DAYS_CREDIT
CB_AVG_DAYS_CREDIT_ENDDATE
CB_AVG_DAYS_ENDDATE_FACT
CB_TOTAL_CNT_CREDIT_PROLONG
CB_AMT_CREDIT_SUM_OVERDUE
CB_PCT_DEBT
CB_AMT_CREDIT_SUM_DEBT
CB_MONTHS_0_AVG
CB_MONTHS_1_AVG
CB_MONTHS_2_AVG
CB_MONTHS_3_AVG
CB_MONTHS_4_AVG
CB_MONTHS_5_AVG
CNT_ACCOUNT_TYPES
CNT_ACTIVE_LOANS
PREV_AVG_DPD_DAYS
PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

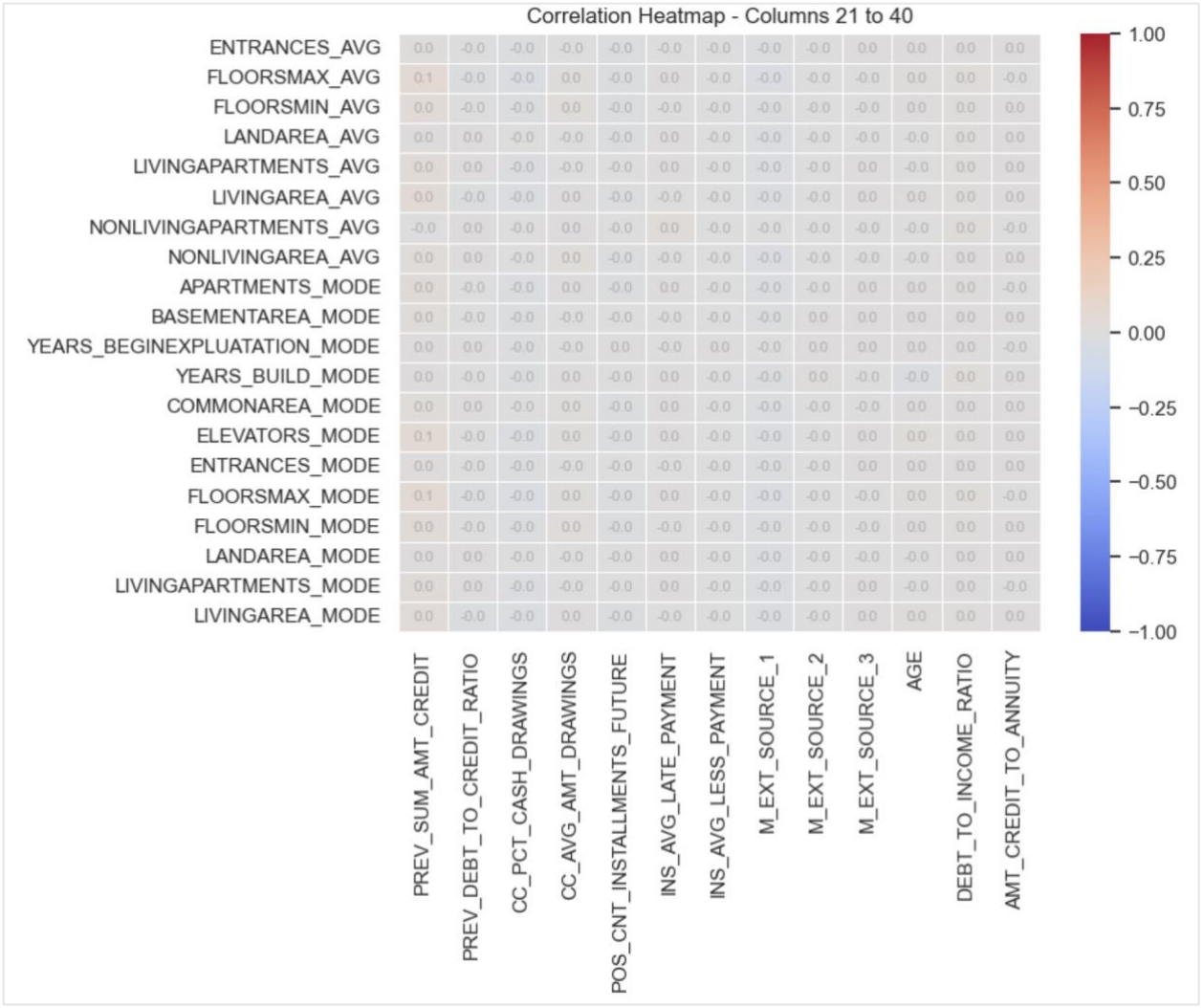

Correlation Heatmap - Columns 1 to 20
TARGET
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_EMPLOYED
DAYS_ID_PUBLISH
OWN_CAR_AGE
CNT_FAM_MEMBERS
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
APARTMENTS_AVG
BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG
YEARS_BUILD_AVG
COMMONAREA_AVG
ELEVATORS_AVG
PREV_SUM_AMT_CREDIT
PREV_DEBT_TO_CREDIT_RATIO
CC_PCT_CASH_DRAWINGS
CC_AVG_AMT_DRAWINGS
POS_CNT_INSTALLMENTS_FUTURE
INS_AVG_LATE_PAYMENT
INS_AVG_LESS_PAYMENT
M_EXT_SOURCE_1
M_EXT_SOURCE_2
M_EXT_SOURCE_3
AGE
DEBT_TO_INCOME_RATIO
AMT_CREDIT_TO_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

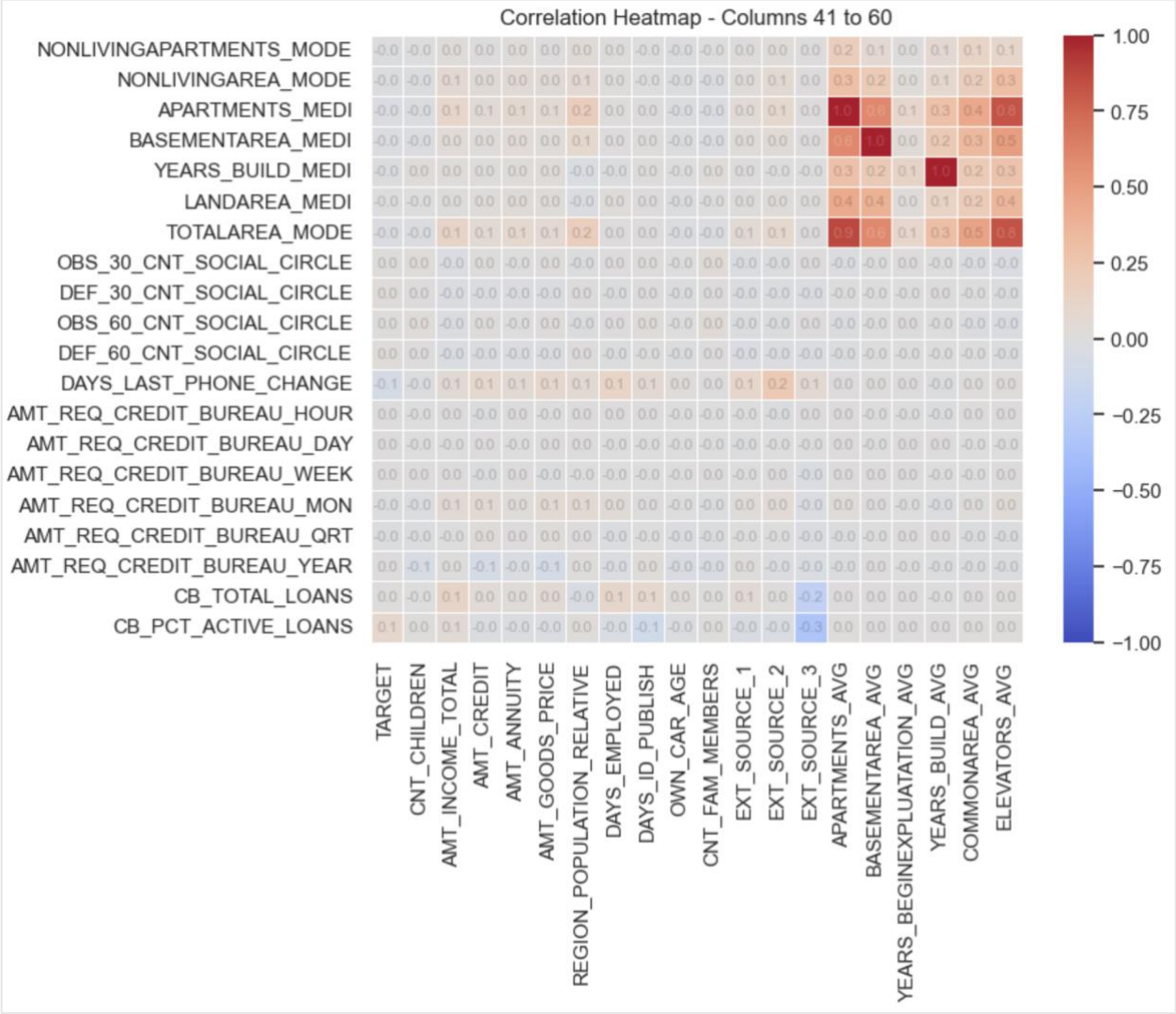

Correlation Heatmap - Columns 21 to 40
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
TARGET
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_EMPLOYED
DAYS_ID_PUBLISH
OWN_CAR_AGE
CNT_FAM_MEMBERS
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
APARTMENTS_AVG
BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG
YEARS_BUILD_AVG
COMMONAREA_AVG
ELEVATORS_AVG
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

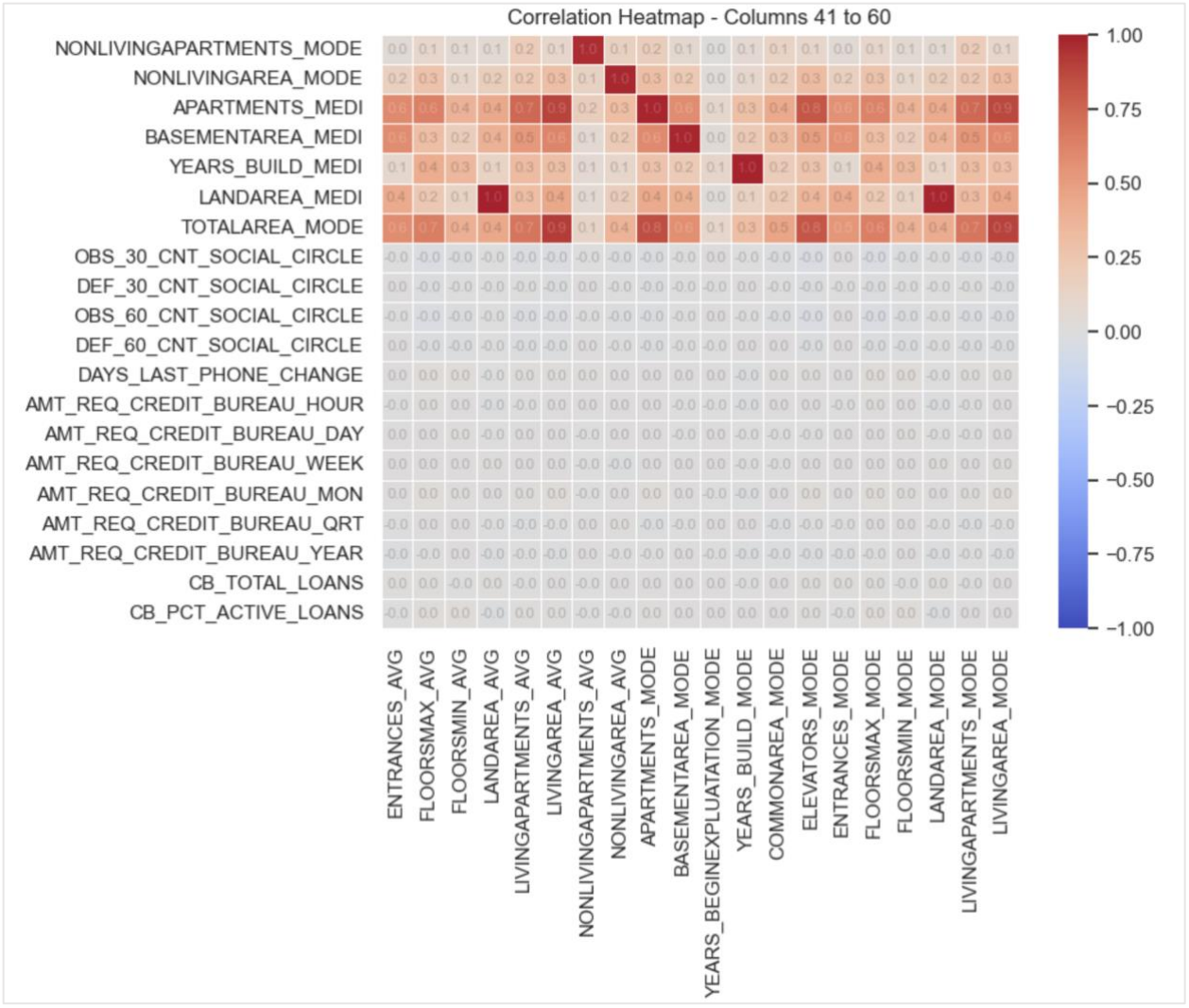

Correlation Heatmap - Columns 21 to 40
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

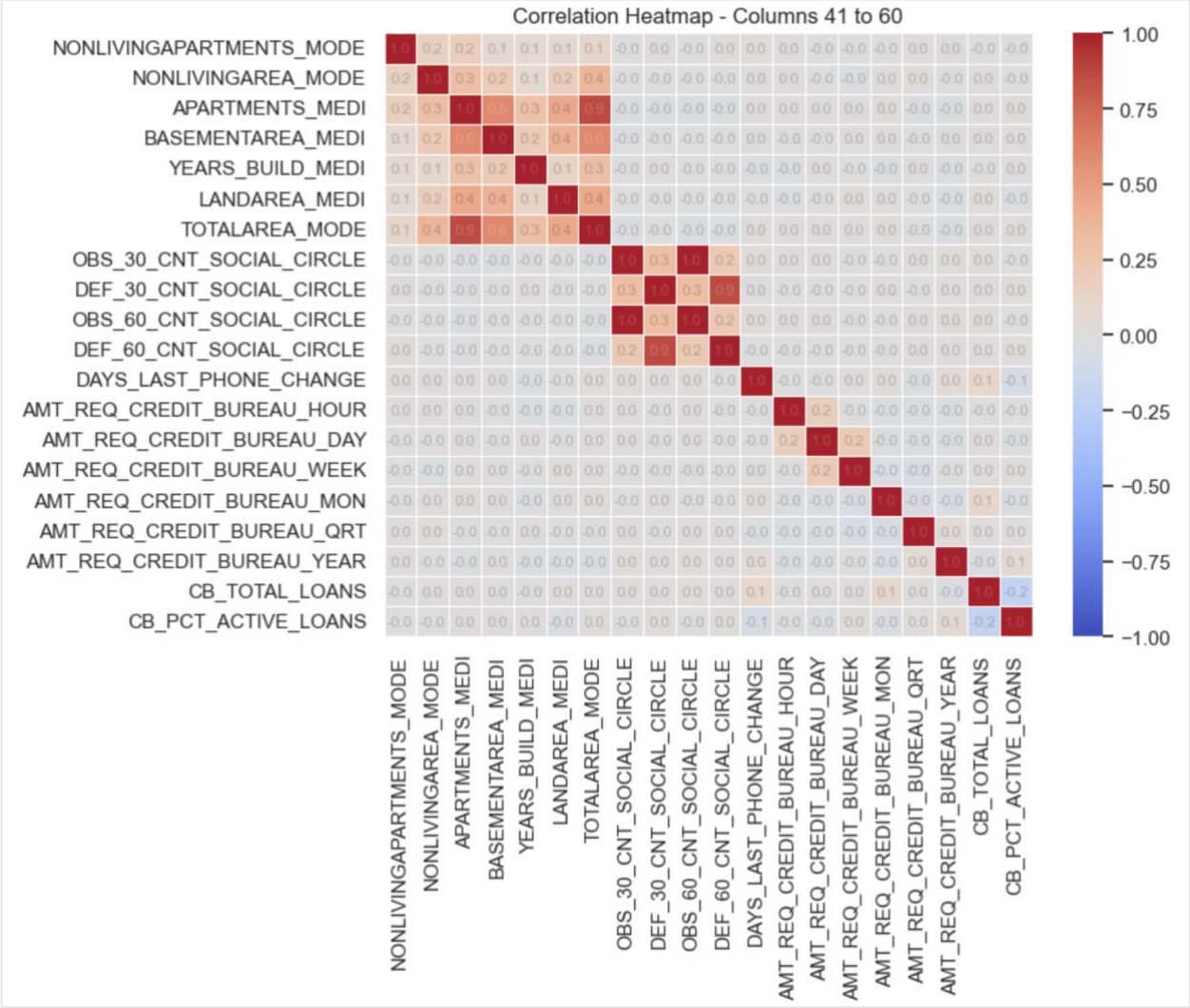

Correlation Heatmap - Columns 21 to 40
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BUILD_MEDI
LANDAREA_MEDI
TOTALAREA_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
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
CB_TOTAL_LOANS
CB_PCT_ACTIVE_LOANS
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

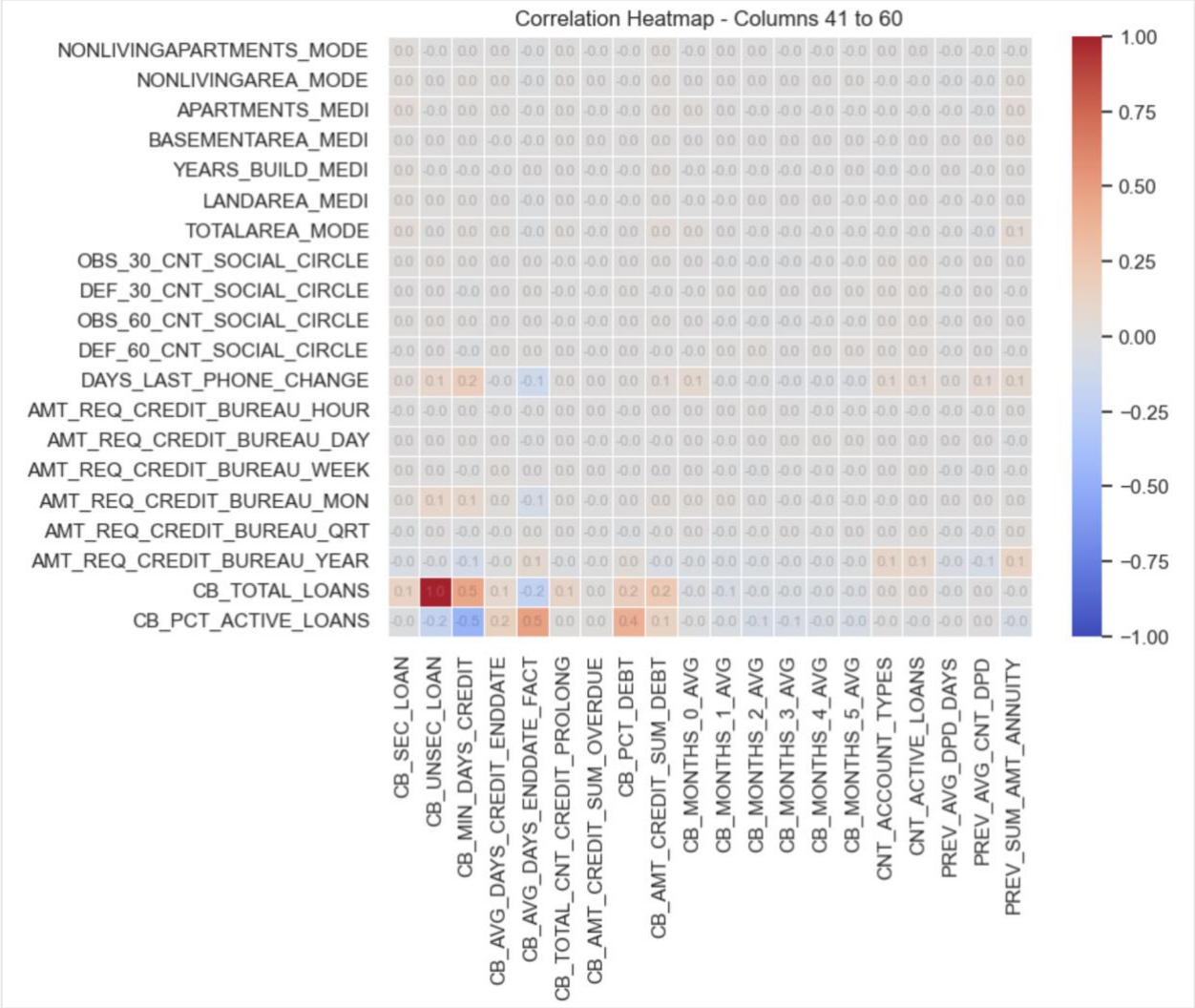

Correlation Heatmap - Columns 21 to 40
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
CB_SEC_LOAN
CB_UNSEC_LOAN
CB_MIN_DAYS_CREDIT
CB_AVG_DAYS_CREDIT_ENDDATE
CB_AVG_DAYS_ENDDATE_FACT
CB_TOTAL_CNT_CREDIT_PROLONG
CB_AMT_CREDIT_SUM_OVERDUE
CB_PCT_DEBT
CB_AMT_CREDIT_SUM_DEBT
CB_MONTHS_0_AVG
CB_MONTHS_1_AVG
CB_MONTHS_2_AVG
CB_MONTHS_3_AVG
CB_MONTHS_4_AVG
CB_MONTHS_5_AVG
CNT_ACCOUNT_TYPES
CNT_ACTIVE_LOANS
PREV_AVG_DPD_DAYS
PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

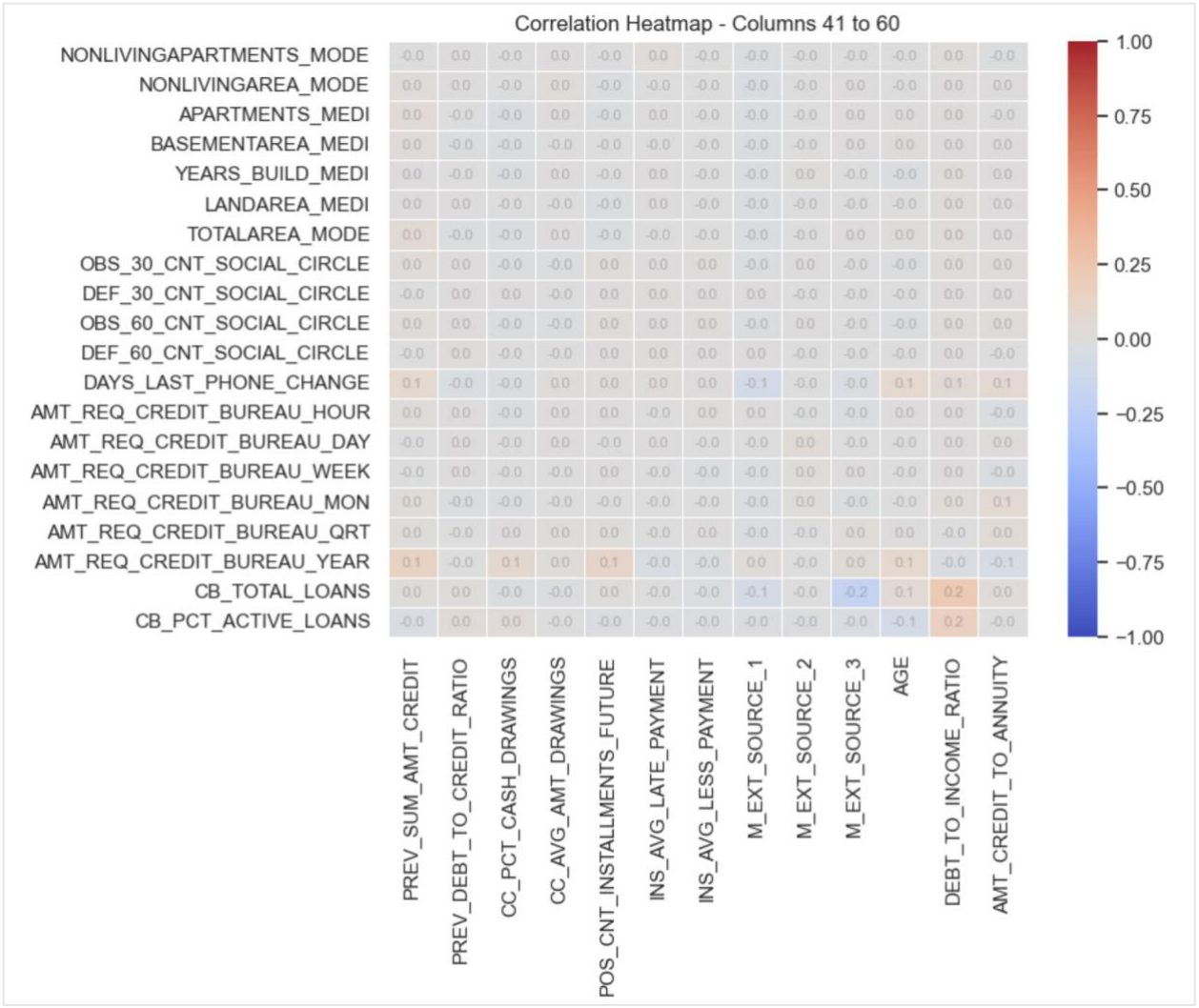

Correlation Heatmap - Columns 21 to 40
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
PREV_SUM_AMT_CREDIT
PREV_DEBT_TO_CREDIT_RATIO
CC_PCT_CASH_DRAWINGS
CC_AVG_AMT_DRAWINGS
POS_CNT_INSTALLMENTS_FUTURE
INS_AVG_LATE_PAYMENT
INS_AVG_LESS_PAYMENT
M_EXT_SOURCE_1
M_EXT_SOURCE_2
M_EXT_SOURCE_3
AGE
DEBT_TO_INCOME_RATIO
AMT_CREDIT_TO_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

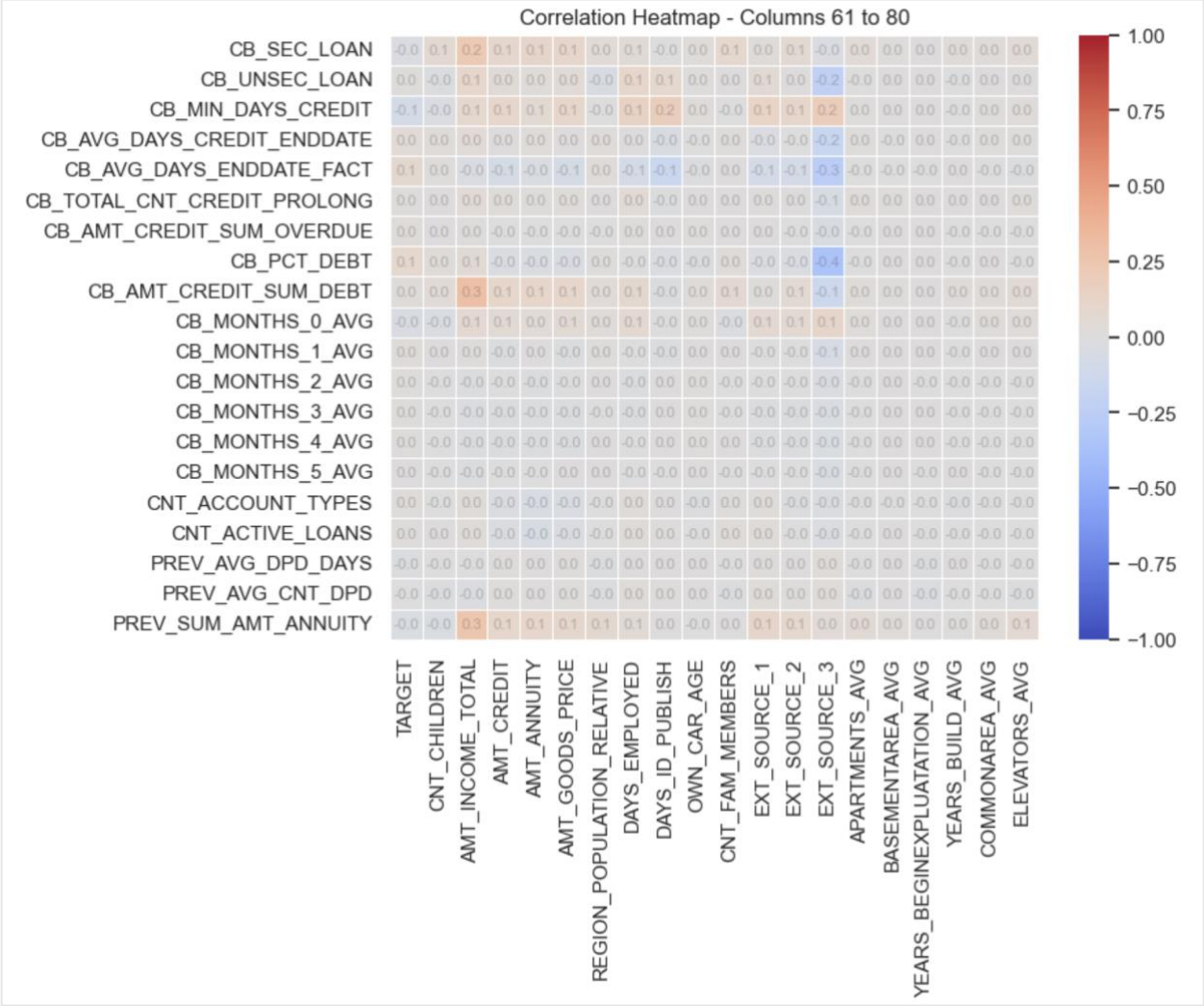

Correlation Heatmap - Columns 41 to 60
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BUILD_MEDI
LANDAREA_MEDI
TOTALAREA_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
AMT_REQ_CREDIT_BUREAU_HOUR
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
CB_TOTAL_LOANS
CB_PCT_ACTIVE_LOANS
TARGET
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_EMPLOYED
DAYS_ID_PUBLISH
OWN_CAR_AGE
CNT_FAM_MEMBERS
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
APARTMENTS_AVG
BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG
YEARS_BUILD_AVG
COMMONAREA_AVG
ELEVATORS_AVG
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

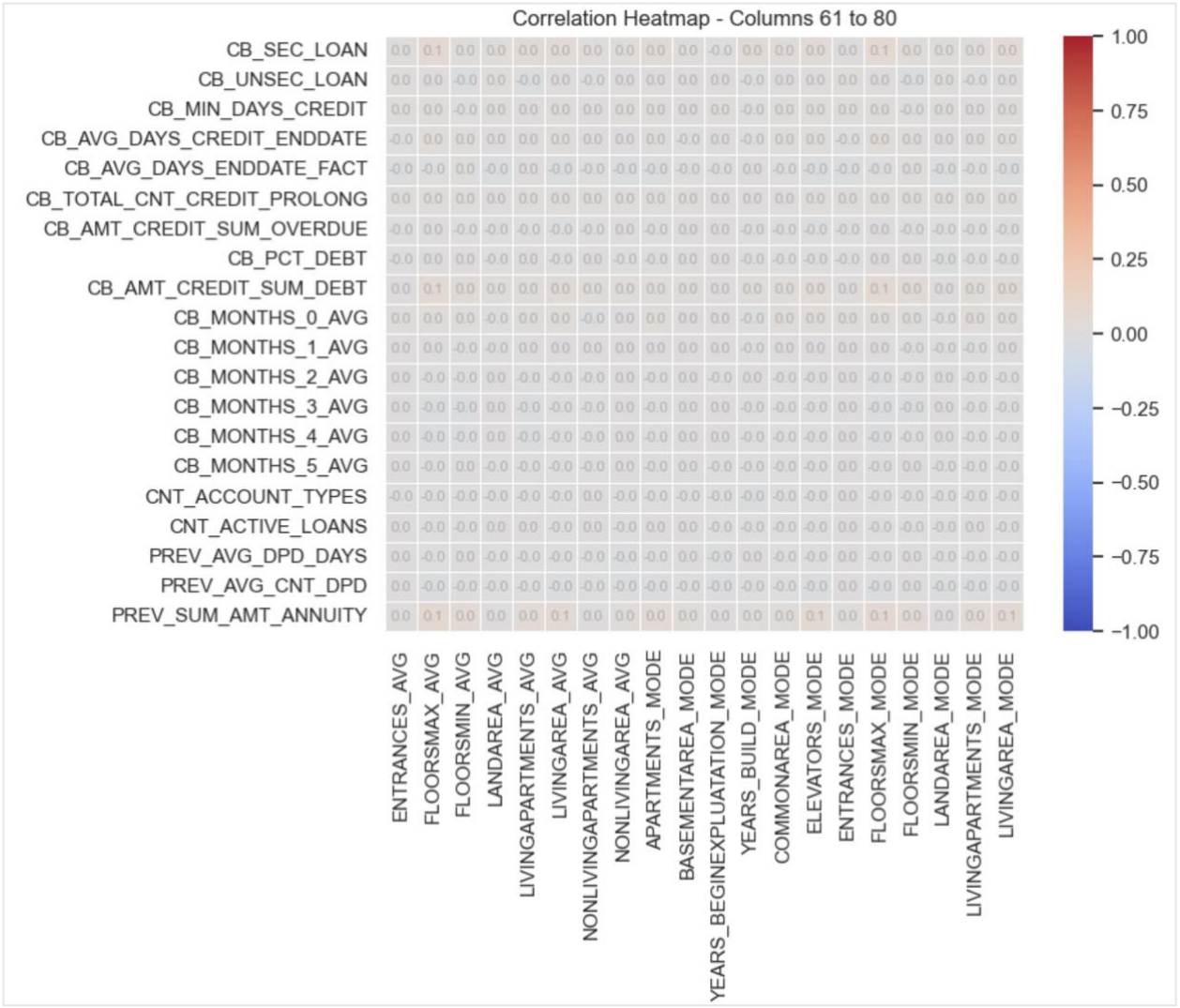

Correlation Heatmap - Columns 41 to 60
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BUILD_MEDI
LANDAREA_MEDI
TOTALAREA_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
AMT_REQ_CREDIT_BUREAU_HOUR
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
CB_TOTAL_LOANS
CB_PCT_ACTIVE_LOANS
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

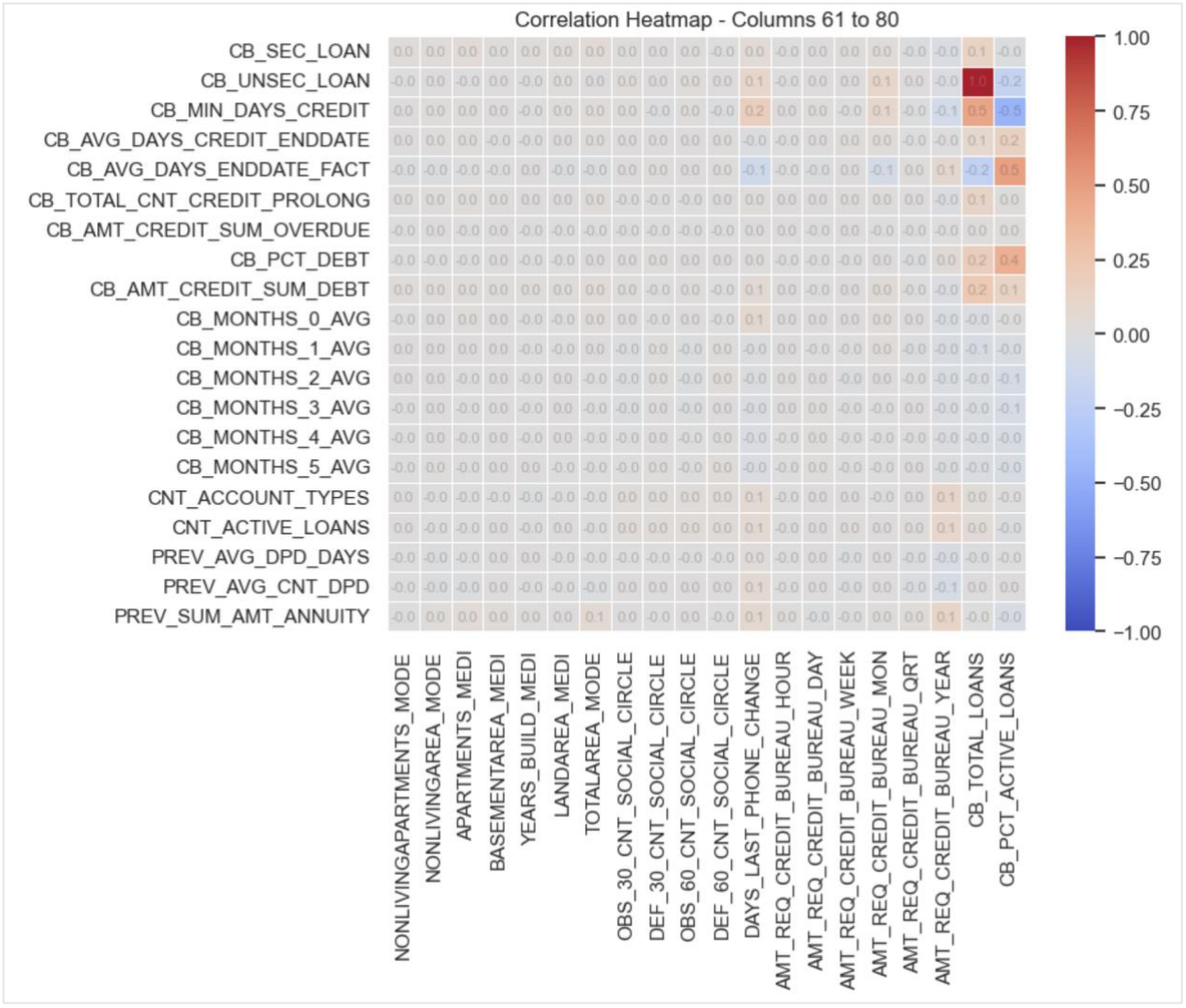

Correlation Heatmap - Columns 41 to 60
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BUILD_MEDI
LANDAREA_MEDI
TOTALAREA_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
AMT_REQ_CREDIT_BUREAU_HOUR
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
CB_TOTAL_LOANS
CB_PCT_ACTIVE_LOANS
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

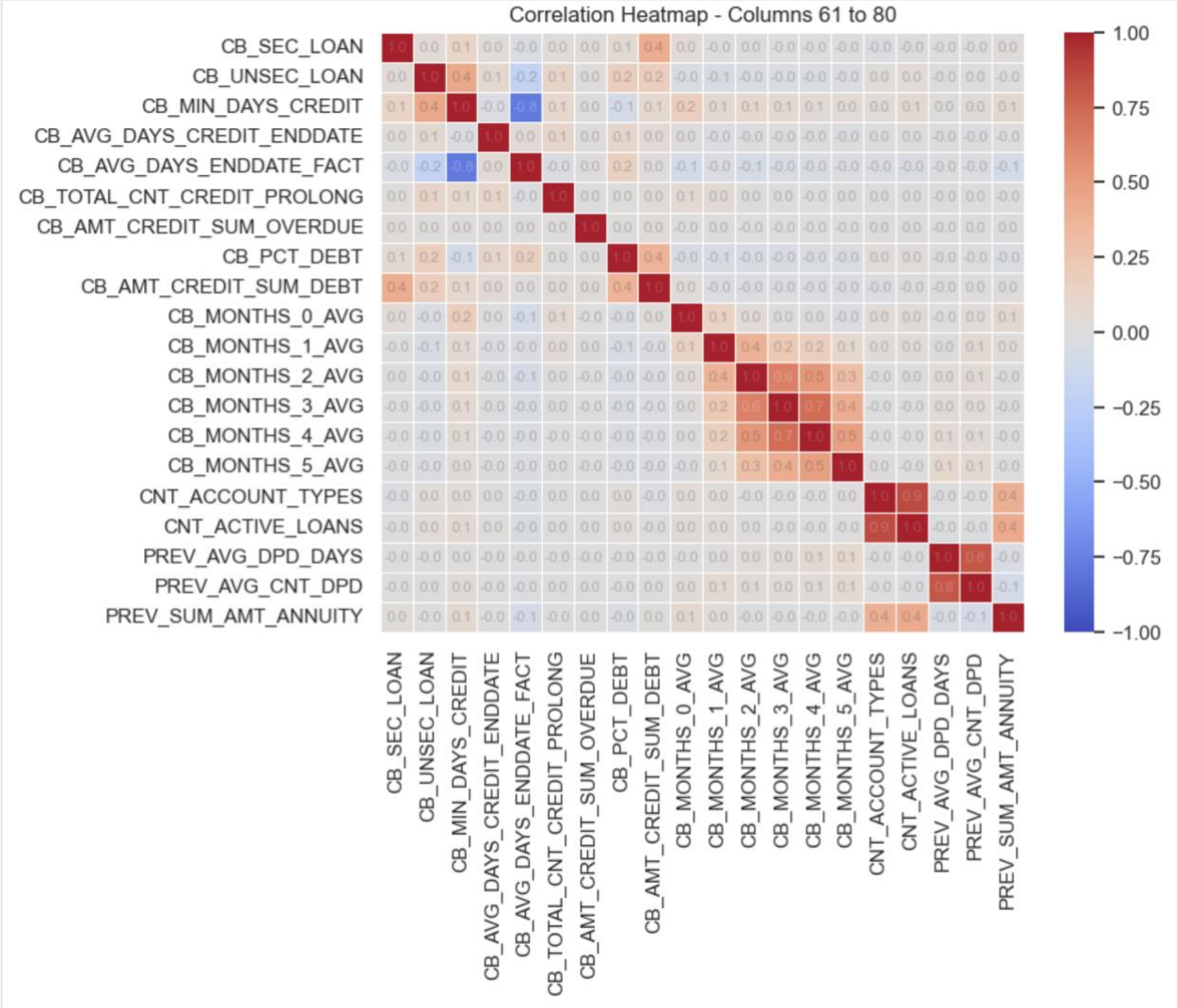

Correlation Heatmap - Columns 41 to 60
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BUILD_MEDI
LANDAREA_MEDI
TOTALAREA_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
AMT_REQ_CREDIT_BUREAU_HOUR
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
CB_TOTAL_LOANS
CB_PCT_ACTIVE_LOANS
CB_SEC_LOAN
CB_UNSEC_LOAN
CB_MIN_DAYS_CREDIT
CB_AVG_DAYS_CREDIT_ENDDATE
CB_AVG_DAYS_ENDDATE_FACT
CB_TOTAL_CNT_CREDIT_PROLONG
CB_AMT_CREDIT_SUM_OVERDUE
CB_PCT_DEBT
CB_AMT_CREDIT_SUM_DEBT
CB_MONTHS_0_AVG
CB_MONTHS_1_AVG
CB_MONTHS_2_AVG
CB_MONTHS_3_AVG
CB_MONTHS_4_AVG
CB_MONTHS_5_AVG
CNT_ACCOUNT_TYPES
CNT_ACTIVE_LOANS
PREV_AVG_DPD_DAYS
PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

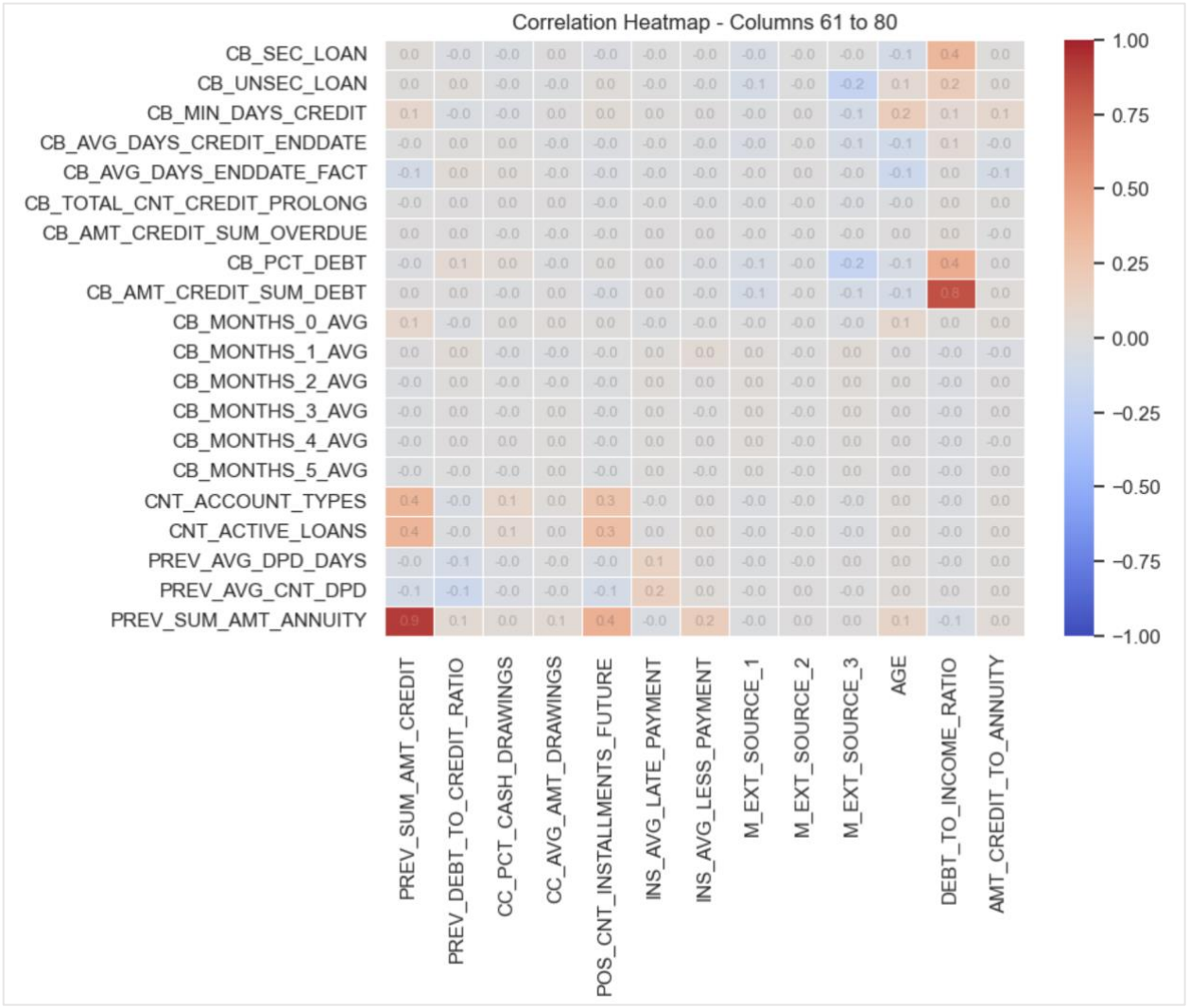

Correlation Heatmap - Columns 41 to 60
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BUILD_MEDI
LANDAREA_MEDI
TOTALAREA_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
AMT_REQ_CREDIT_BUREAU_HOUR
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
CB_TOTAL_LOANS
CB_PCT_ACTIVE_LOANS
PREV_SUM_AMT_CREDIT
PREV_DEBT_TO_CREDIT_RATIO
CC_PCT_CASH_DRAWINGS
CC_AVG_AMT_DRAWINGS
POS_CNT_INSTALLMENTS_FUTURE
INS_AVG_LATE_PAYMENT
INS_AVG_LESS_PAYMENT
M_EXT_SOURCE_1
M_EXT_SOURCE_2
M_EXT_SOURCE_3
AGE
DEBT_TO_INCOME_RATIO
AMT_CREDIT_TO_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

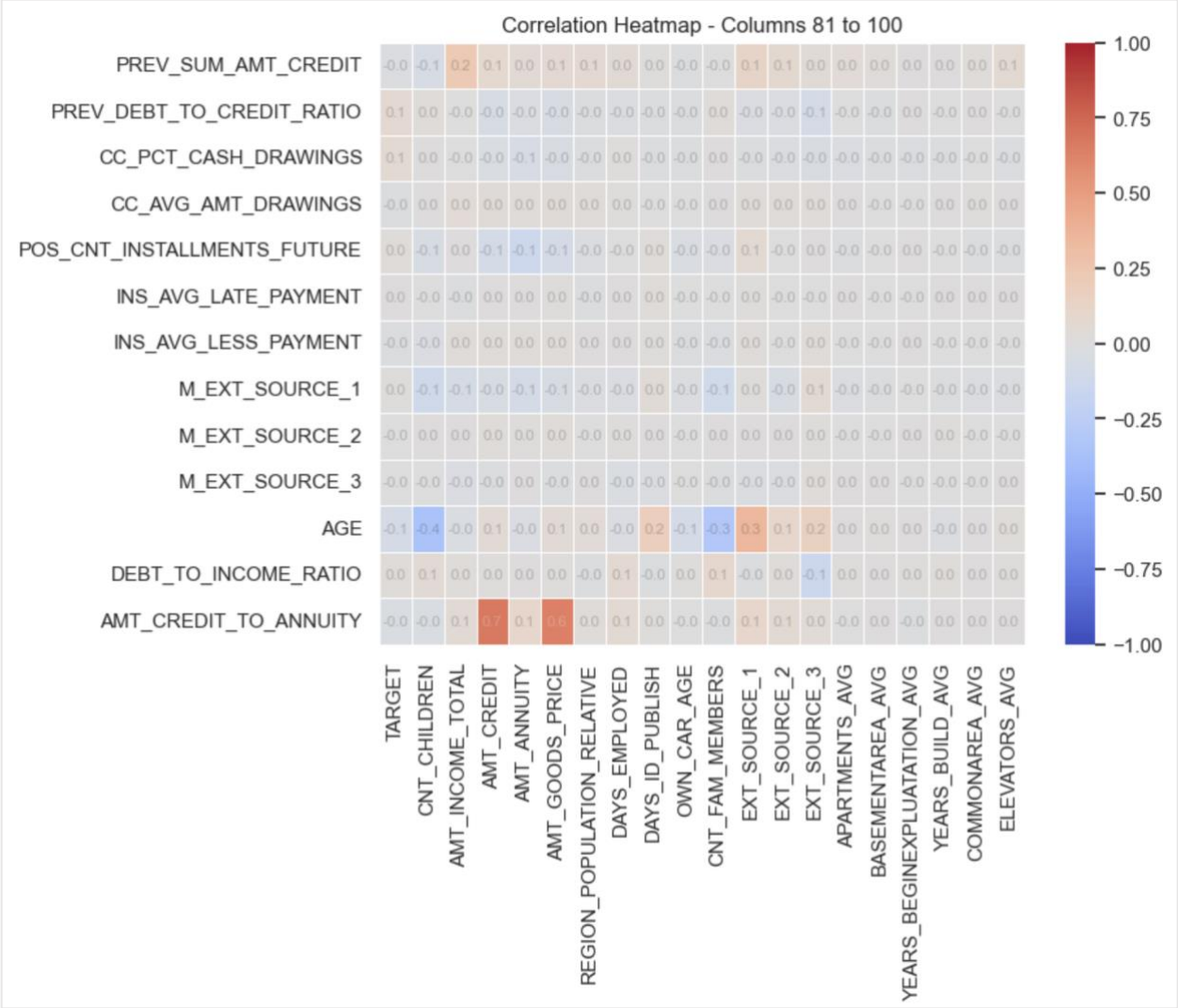

Correlation Heatmap - Columns 61 to 80
CB_SEC_LOAN
CB_UNSEC_LOAN
CB_MIN_DAYS_CREDIT
CB_AVG_DAYS_CREDIT_ENDDATE
CB_AVG_DAYS_ENDDATE_FACT
CB_TOTAL_CNT_CREDIT_PROLONG
CB_AMT_CREDIT_SUM_OVERDUE
CB_PCT_DEBT
CB_AMT_CREDIT_SUM_DEBT
CB_MONTHS_0_AVG
CB_MONTHS_1_AVG
CB_MONTHS_2_AVG
CB_MONTHS_3_AVG
CB_MONTHS_4_AVG
CB_MONTHS_5_AVG
CNT_ACCOUNT_TYPES
CNT_ACTIVE_LOANS
PREV_AVG_DPD_DAYS
PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY
TARGET
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_EMPLOYED
DAYS_ID_PUBLISH
OWN_CAR_AGE
CNT_FAM_MEMBERS
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
APARTMENTS_AVG
BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG
YEARS_BUILD_AVG
COMMONAREA_AVG
ELEVATORS_AVG
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

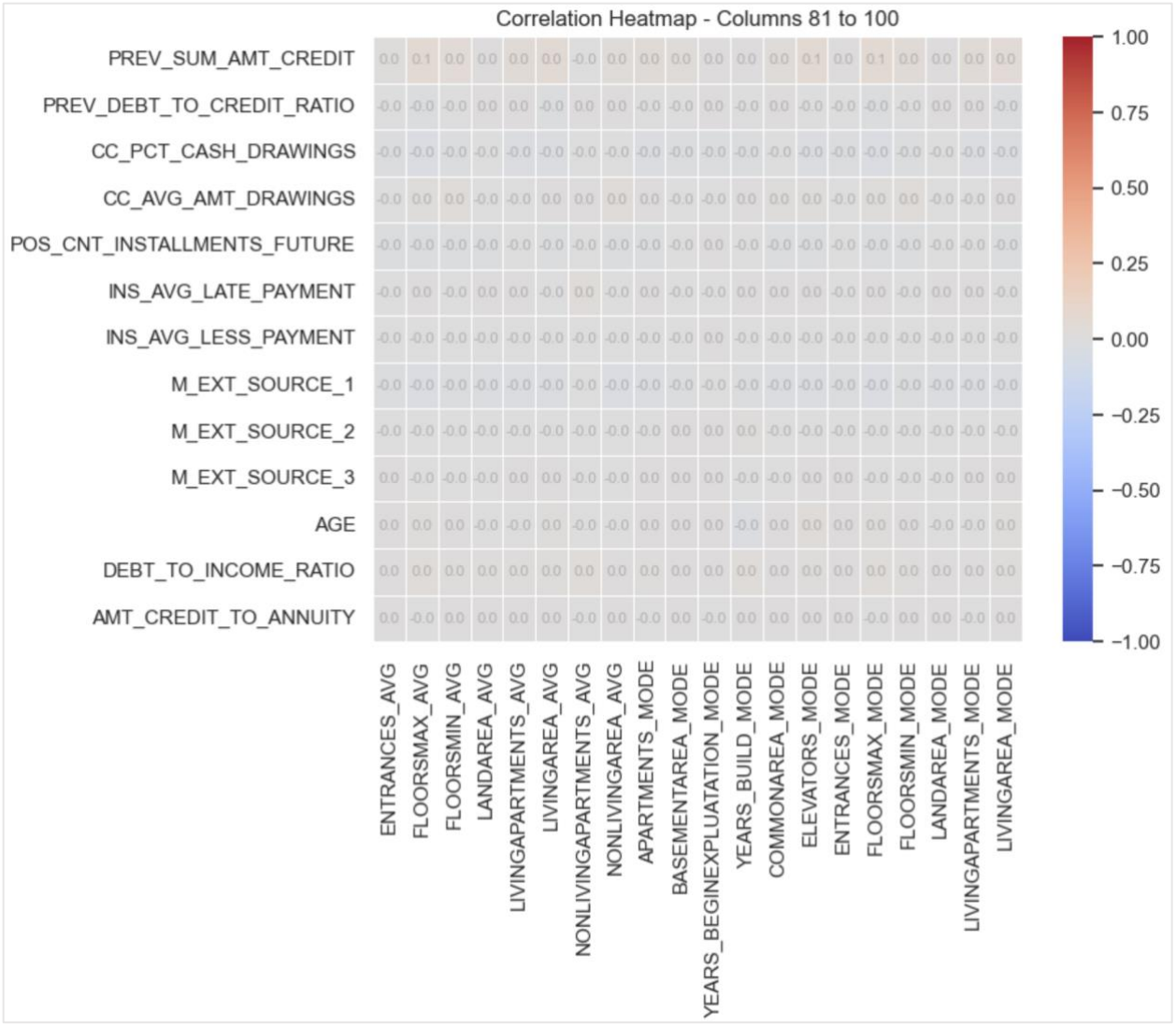

Correlation Heatmap - Columns 61 to 80
CB_SEC_LOAN
CB_UNSEC_LOAN
CB_MIN_DAYS_CREDIT
CB_AVG_DAYS_CREDIT_ENDDATE
CB_AVG_DAYS_ENDDATE_FACT
CB_TOTAL_CNT_CREDIT_PROLONG
CB_AMT_CREDIT_SUM_OVERDUE
CB_PCT_DEBT
CB_AMT_CREDIT_SUM_DEBT
CB_MONTHS_0_AVG
CB_MONTHS_1_AVG
CB_MONTHS_2_AVG
CB_MONTHS_3_AVG
CB_MONTHS_4_AVG
CB_MONTHS_5_AVG
CNT_ACCOUNT_TYPES
CNT_ACTIVE_LOANS
PREV_AVG_DPD_DAYS
PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

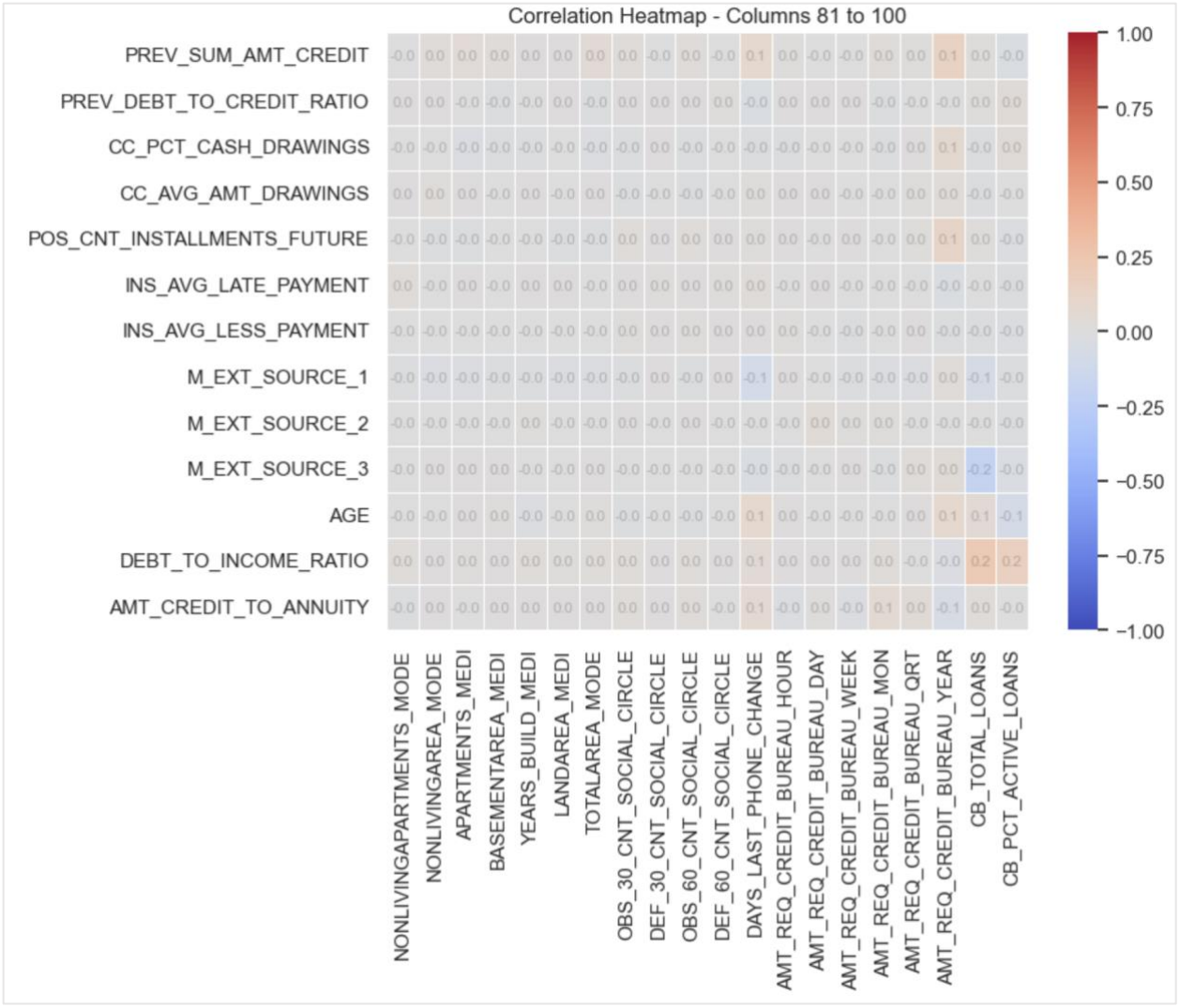

Correlation Heatmap - Columns 61 to 80
CB_SEC_LOAN
CB_UNSEC_LOAN
CB_MIN_DAYS_CREDIT
CB_AVG_DAYS_CREDIT_ENDDATE
CB_AVG_DAYS_ENDDATE_FACT
CB_TOTAL_CNT_CREDIT_PROLONG
CB_AMT_CREDIT_SUM_OVERDUE
CB_PCT_DEBT
CB_AMT_CREDIT_SUM_DEBT
CB_MONTHS_0_AVG
CB_MONTHS_1_AVG
CB_MONTHS_2_AVG
CB_MONTHS_3_AVG
CB_MONTHS_4_AVG
CB_MONTHS_5_AVG
CNT_ACCOUNT_TYPES
CNT_ACTIVE_LOANS
PREV_AVG_DPD_DAYS
PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BUILD_MEDI
LANDAREA_MEDI
TOTALAREA_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
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
CB_TOTAL_LOANS
CB_PCT_ACTIVE_LOANS
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

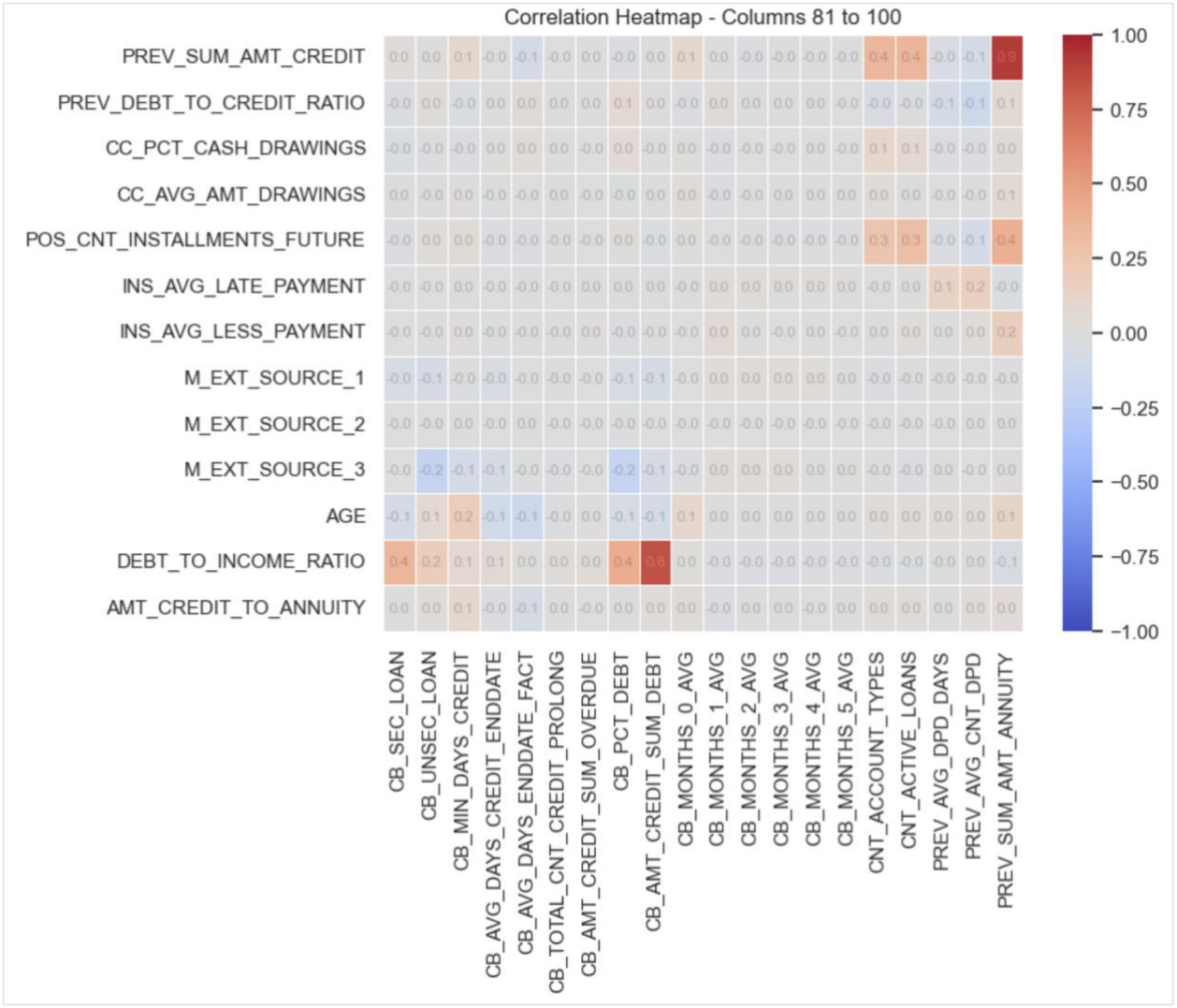

Correlation Heatmap - Columns 61 to 80
CB_SEC_LOAN
CB_UNSEC_LOAN
CB_MIN_DAYS_CREDIT
CB_AVG_DAYS_CREDIT_ENDDATE
CB_AVG_DAYS_ENDDATE_FACT
CB_TOTAL_CNT_CREDIT_PROLONG
CB_AMT_CREDIT_SUM_OVERDUE
CB_PCT_DEBT
CB_AMT_CREDIT_SUM_DEBT
CB_MONTHS_0_AVG
CB_MONTHS_1_AVG
CB_MONTHS_2_AVG
CB_MONTHS_3_AVG
CB_MONTHS_4_AVG
CB_MONTHS_5_AVG
CNT_ACCOUNT_TYPES
CNT_ACTIVE_LOANS
PREV_AVG_DPD_DAYS
PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Correlation Heatmap - Columns 61 to 80
CB_SEC_LOAN
CB_UNSEC_LOAN
CB_MIN_DAYS_CREDIT
CB_AVG_DAYS_CREDIT_ENDDATE
CB_AVG_DAYS_ENDDATE_FACT
CB_TOTAL_CNT_CREDIT_PROLONG
CB_AMT_CREDIT_SUM_OVERDUE
CB_PCT_DEBT
CB_AMT_CREDIT_SUM_DEBT
CB_MONTHS_0_AVG
CB_MONTHS_1_AVG
CB_MONTHS_2_AVG
CB_MONTHS_3_AVG
CB_MONTHS_4_AVG
CB_MONTHS_5_AVG
CNT_ACCOUNT_TYPES
CNT_ACTIVE_LOANS
PREV_AVG_DPD_DAYS
PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY
PREV_SUM_AMT_CREDIT
PREV_DEBT_TO_CREDIT_RATIO
CC_PCT_CASH_DRAWINGS
CC_AVG_AMT_DRAWINGS
POS_CNT_INSTALLMENTS_FUTURE
INS_AVG_LATE_PAYMENT
INS_AVG_LESS_PAYMENT
M_EXT_SOURCE_1
M_EXT_SOURCE_2
M_EXT_SOURCE_3
AGE
DEBT_TO_INCOME_RATIO
AMT_CREDIT_TO_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Correlation Heatmap - Columns 81 to 100
PREV_SUM_AMT_CREDIT
PREV_DEBT_TO_CREDIT_RATIO
CC_PCT_CASH_DRAWINGS
CC_AVG_AMT_DRAWINGS
POS_CNT_INSTALLMENTS_FUTURE
INS_AVG_LATE_PAYMENT
INS_AVG_LESS_PAYMENT
M_EXT_SOURCE_1
M_EXT_SOURCE_2
M_EXT_SOURCE_3
AGE
DEBT_TO_INCOME_RATIO
AMT_CREDIT_TO_ANNUITY
TARGET
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
DAYS_EMPLOYED
DAYS_ID_PUBLISH
OWN_CAR_AGE
CNT_FAM_MEMBERS
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
APARTMENTS_AVG
BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG
YEARS_BUILD_AVG
COMMONAREA_AVG
ELEVATORS_AVG
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Correlation Heatmap - Columns 81 to 100
PREV_SUM_AMT_CREDIT
PREV_DEBT_TO_CREDIT_RATIO
CC_PCT_CASH_DRAWINGS
CC_AVG_AMT_DRAWINGS
POS_CNT_INSTALLMENTS_FUTURE
INS_AVG_LATE_PAYMENT
INS_AVG_LESS_PAYMENT
M_EXT_SOURCE_1
M_EXT_SOURCE_2
M_EXT_SOURCE_3
AGE
DEBT_TO_INCOME_RATIO
AMT_CREDIT_TO_ANNUITY
ENTRANCES_AVG
FLOORSMAX_AVG
FLOORSMIN_AVG
LANDAREA_AVG
LIVINGAPARTMENTS_AVG
LIVINGAREA_AVG
NONLIVINGAPARTMENTS_AVG
NONLIVINGAREA_AVG
APARTMENTS_MODE
BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE
COMMONAREA_MODE
ELEVATORS_MODE
ENTRANCES_MODE
FLOORSMAX_MODE
FLOORSMIN_MODE
LANDAREA_MODE
LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Correlation Heatmap - Columns 81 to 100
PREV_SUM_AMT_CREDIT
PREV_DEBT_TO_CREDIT_RATIO
CC_PCT_CASH_DRAWINGS
CC_AVG_AMT_DRAWINGS
POS_CNT_INSTALLMENTS_FUTURE
INS_AVG_LATE_PAYMENT
INS_AVG_LESS_PAYMENT
M_EXT_SOURCE_1
M_EXT_SOURCE_2
M_EXT_SOURCE_3
AGE
DEBT_TO_INCOME_RATIO
AMT_CREDIT_TO_ANNUITY
NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE
APARTMENTS_MEDI
BASEMENTAREA_MEDI
YEARS_BUILD_MEDI
LANDAREA_MEDI
TOTALAREA_MODE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE
DEF_60_CNT_SOCIAL_CIRCLE
DAYS_LAST_PHONE_CHANGE
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
CB_TOTAL_LOANS
CB_PCT_ACTIVE_LOANS
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Correlation Heatmap - Columns 81 to 100
PREV_SUM_AMT_CREDIT
PREV_DEBT_TO_CREDIT_RATIO
CC_PCT_CASH_DRAWINGS
CC_AVG_AMT_DRAWINGS
POS_CNT_INSTALLMENTS_FUTURE
INS_AVG_LATE_PAYMENT
INS_AVG_LESS_PAYMENT
M_EXT_SOURCE_1
M_EXT_SOURCE_2
M_EXT_SOURCE_3
AGE
DEBT_TO_INCOME_RATIO
AMT_CREDIT_TO_ANNUITY
CB_SEC_LOAN
CB_UNSEC_LOAN
CB_MIN_DAYS_CREDIT
CB_AVG_DAYS_CREDIT_ENDDATE
CB_AVG_DAYS_ENDDATE_FACT
CB_TOTAL_CNT_CREDIT_PROLONG
CB_AMT_CREDIT_SUM_OVERDUE
CB_PCT_DEBT
CB_AMT_CREDIT_SUM_DEBT
CB_MONTHS_0_AVG
CB_MONTHS_1_AVG
CB_MONTHS_2_AVG
CB_MONTHS_3_AVG
CB_MONTHS_4_AVG
CB_MONTHS_5_AVG
CNT_ACCOUNT_TYPES
CNT_ACTIVE_LOANS
PREV_AVG_DPD_DAYS
PREV_AVG_CNT_DPD
PREV_SUM_AMT_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

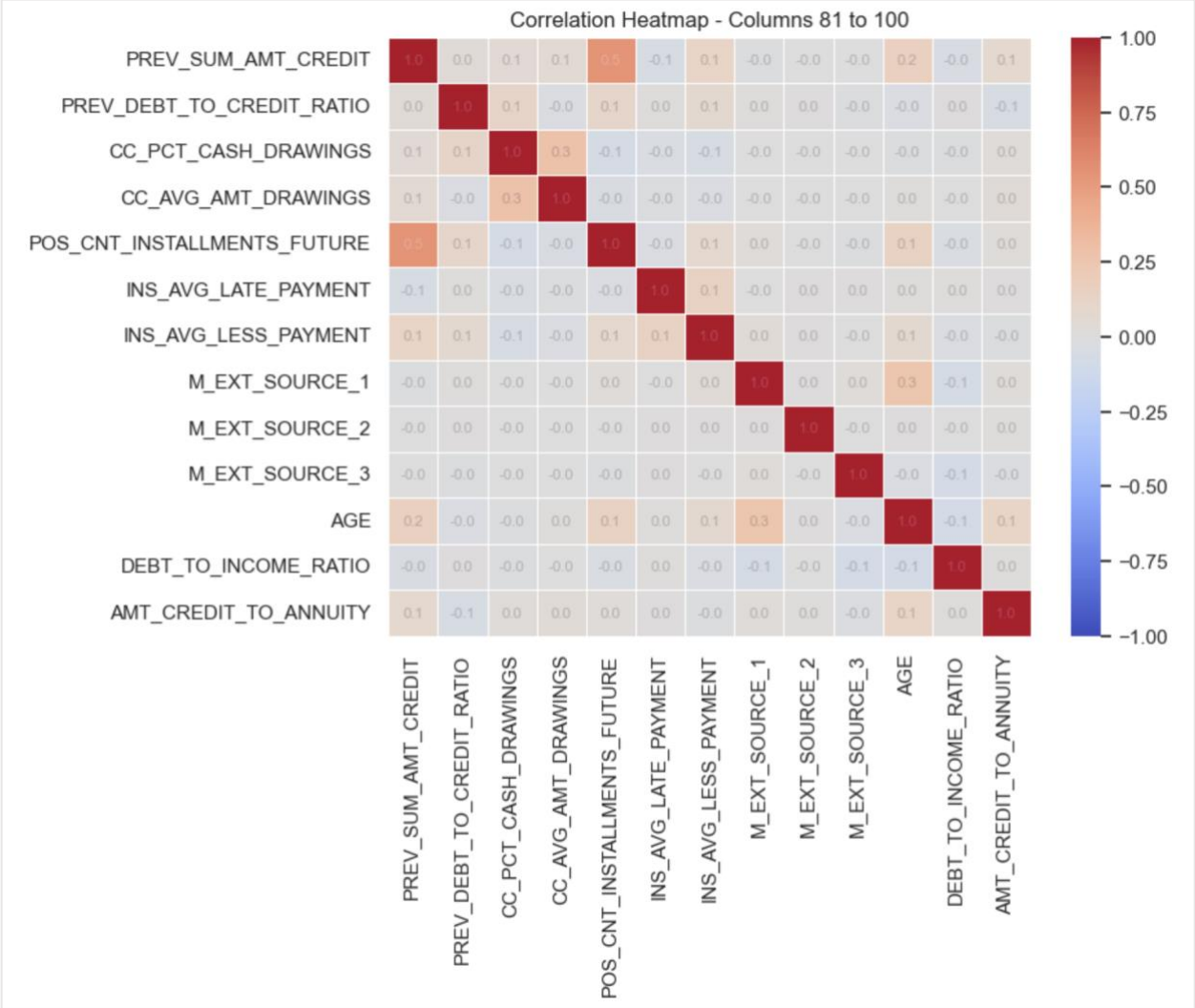
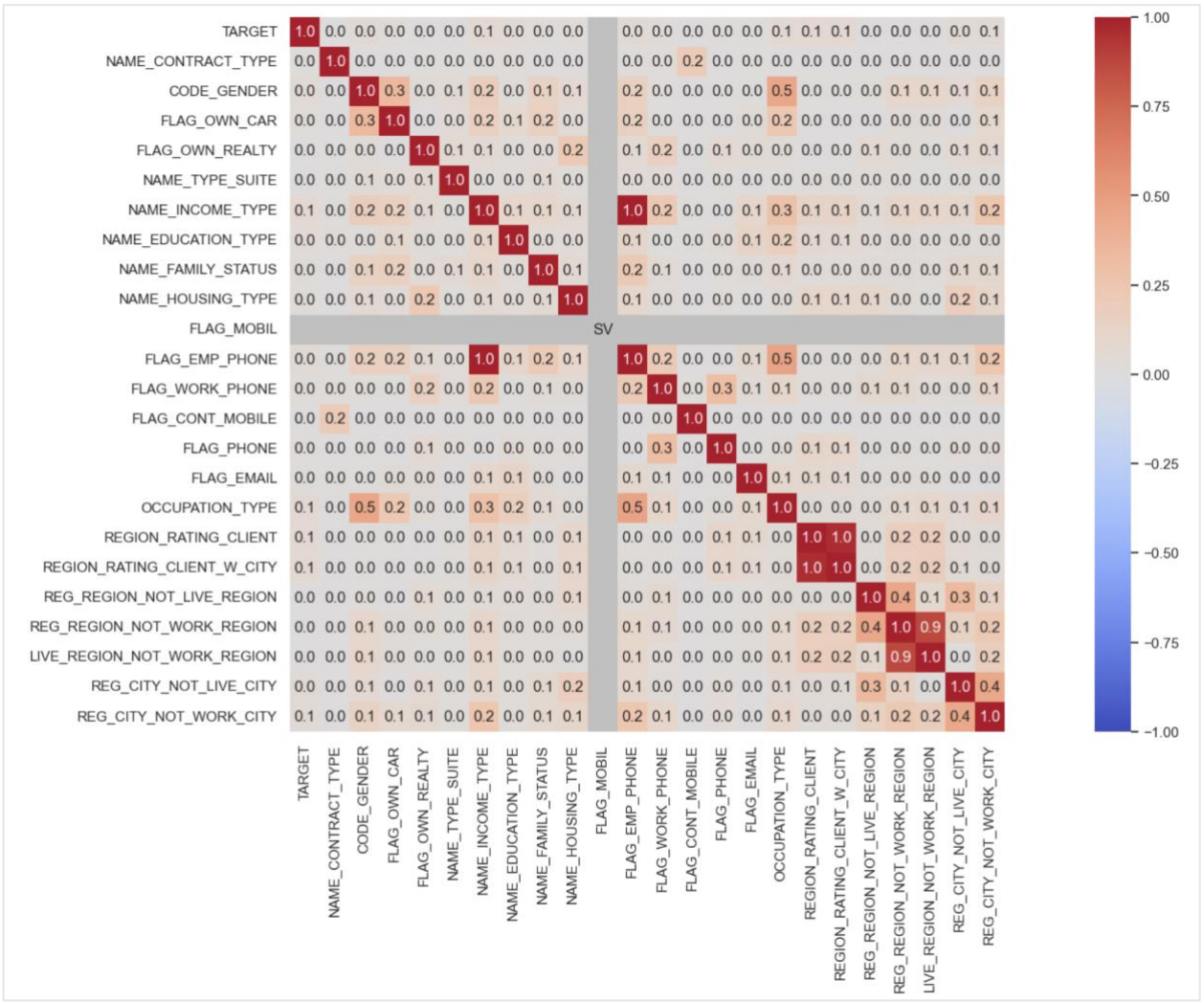

Correlation Heatmap - Columns 81 to 100
PREV_SUM_AMT_CREDIT
PREV_DEBT_TO_CREDIT_RATIO
CC_PCT_CASH_DRAWINGS
CC_AVG_AMT_DRAWINGS
POS_CNT_INSTALLMENTS_FUTURE
INS_AVG_LATE_PAYMENT
INS_AVG_LESS_PAYMENT
M_EXT_SOURCE_1
M_EXT_SOURCE_2
M_EXT_SOURCE_3
AGE
DEBT_TO_INCOME_RATIO
AMT_CREDIT_TO_ANNUITY
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Figure 40 Categorical correlations - Dython library



10 REFERENCES

Kaggle. Home Credit. https://www.kaggle.com/competitions/home-credit-default-risk/data

Fico.com. 2023. *Using alternative data in credit risk modeling*. https://www.fico.com/blogs/using-alternative-data-credit-risk-modelling

Github. Shap. https://github.com/shap/shap

TransUnion. 2022. More than 9 Million Canadians are either credit unserved or underserved. https://newsroom.transunion.ca/more-than-9-million-canadians-are-either-credit-unserved-or-underserved---approximately-14-migrate-to-being-credit-active-every-two-years