HOME CREDIT

Group 8

Data Analysis Ideas

Table of contents

01
Dataset Introduction

Overview

Brief Description

01 | Overview

Home Credit Default Risk

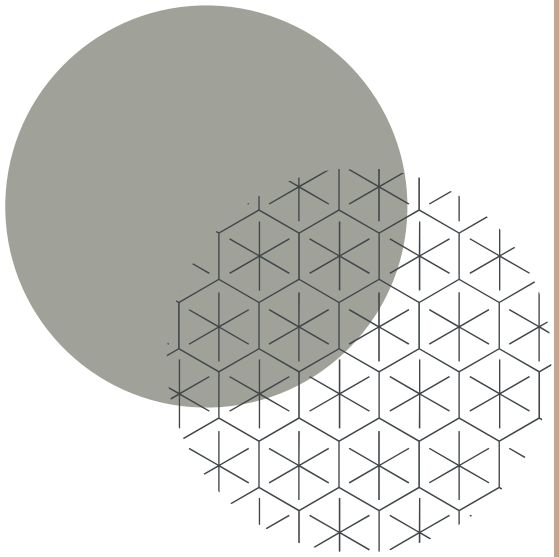
- A lot of people who apply for loans in Banks and similar financial institutions whereas only a few of them get approved

- Home Credit uses a lot of data to predict the what are the factors that seem to influence the being fraud or not

01 Brief Description

There are **7 datasets**

Application		
Bureau	Bureau_balance	Previous_application
Credit_card_balance	Installments_payments	POS_CASH_balance

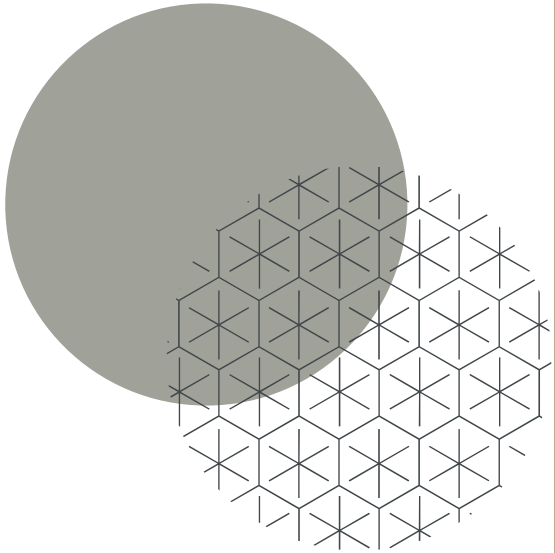


Application

- This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET).
- Static data for all applications. One row represents one loan in our data sample.

Application

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	...	FLAG_DOCUMENT_18
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5	...	0
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502.5	35698.5	...	0
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0	6750.0	...	0
3	100006	0	Cash loans	F	N	Y	0	135000.0	312682.5	29686.5	...	0
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000.0	21865.5	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
307506	456251	0	Cash loans	M	N	N	0	157500.0	254700.0	27558.0	...	0
307507	456252	0	Cash loans	F	N	Y	0	72000.0	269550.0	12001.5	...	0
307508	456253	0	Cash loans	F	N	Y	0	153000.0	677664.0	29979.0	...	0
307509	456254	1	Cash loans	F	N	Y	0	171000.0	370107.0	20205.0	...	0
307510	456255	0	Cash loans	F	N	N	0	157500.0	675000.0	49117.5	...	0



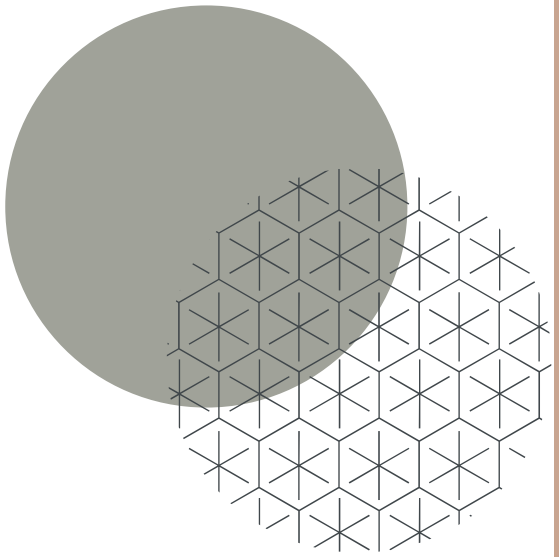
Bureau

- All client's previous credits provided by other financial institutions that were reported to Credit Bureau

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

Bureau

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_DAY_OVERDUE	DAYS_CREDIT_ENDDATE	DAYS_ENDDATE_FACT	AMT_CREDIT_MAX_OVERDUE
0	215354	5714462	Closed	currency 1	-497	0	-153.0	-153.0	NaN
1	215354	5714463	Active	currency 1	-208	0	1075.0	NaN	NaN
2	215354	5714464	Active	currency 1	-203	0	528.0	NaN	NaN
3	215354	5714465	Active	currency 1	-203	0	NaN	NaN	NaN
4	215354	5714466	Active	currency 1	-629	0	1197.0	NaN	77674.5
...	...	...	...	...	...	...	...	...	...
1716423	259355	5057750	Active	currency 1	-44	0	-30.0	NaN	0.0
1716424	100044	5057754	Closed	currency 1	-2648	0	-2433.0	-2493.0	5476.5
1716425	100044	5057762	Closed	currency 1	-1809	0	-1628.0	-970.0	NaN

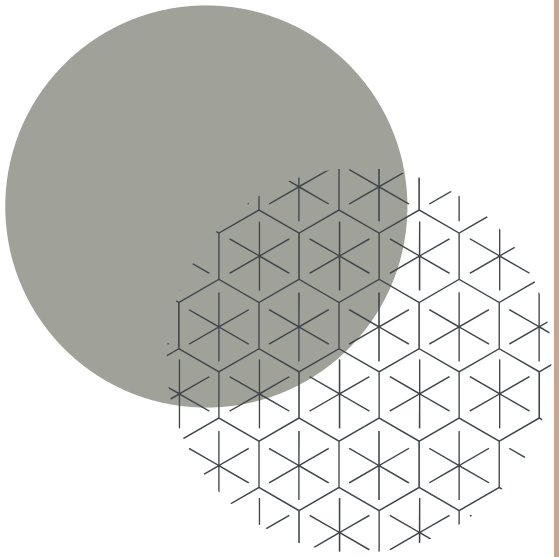


Bureau_balance

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

Bureau_balance

	SK_ID_BUREAU	MONTHS_BALANCE	STATUS
0	5715448	0	C
1	5715448	-1	C
2	5715448	-2	C
3	5715448	-3	C
4	5715448	-4	C
...	...	...	...
27299920	5041336	-47	X
27299921	5041336	-48	X
27299922	5041336	-49	X

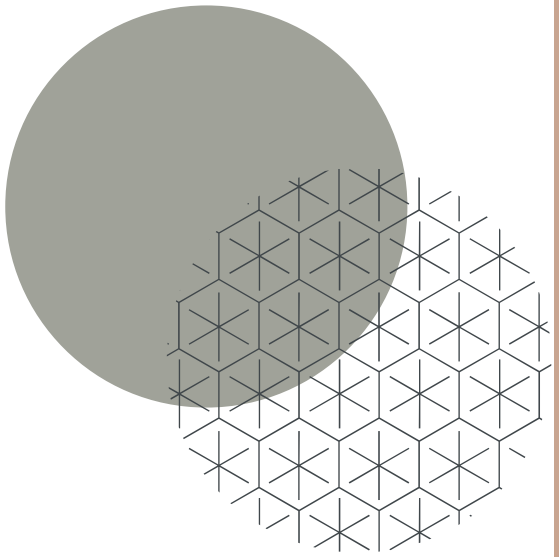


POS_CASH_balance

- Monthly balance snapshots of previous point of sales and cash loans the applicant had with Home Credit.
- This table has one row for each month of history of every previous credit in Home Credit related to loans.

POS_CASH_balance

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	CNT_INSTALMENT	CNT_INSTALMENT_FUTURE	NAME_CONTRACT_STATUS	SK_DPD	SK_DPD_DEF
0	1803195	182943	-31	48.0	45.0	Active	0	0
1	1715348	367990	-33	36.0	35.0	Active	0	0
2	1784872	397406	-32	12.0	9.0	Active	0	0
3	1903291	269225	-35	48.0	42.0	Active	0	0
4	2341044	334279	-35	36.0	35.0	Active	0	0
...	...	...	...	...	...	...	...	...
10001353	2448283	226558	-20	6.0	0.0	Active	843	0
10001354	1717234	141565	-19	12.0	0.0	Active	602	0
10001355	1283126	315695	-21	10.0	0.0	Active	609	0

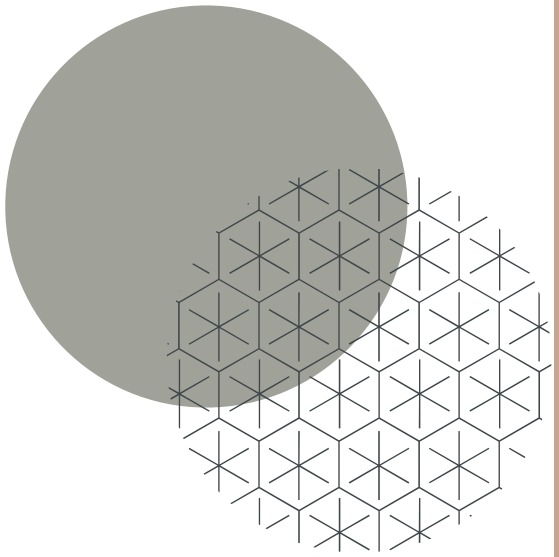


Credit_card_balance

- Monthly balance snapshots of previous credit cards that the applicant has with Home Credit.
- This table has one row for each month of history of every previous credit in Home Credit related to loans.

Credit_card_balance

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL	AMT_DRAWINGS_ATM_CURRENT	AMT_DRAWINGS_CURRENT	AMT_DRAWINGS_OTHER_CURRENT
0	2562384	378907	-6	56.970	135000	0.0	877.5	0.0
1	2582071	363914	-1	63975.555	45000	2250.0	2250.0	0.0
2	1740877	371185	-7	31815.225	450000	0.0	0.0	0.0
3	1389973	337855	-4	236572.110	225000	2250.0	2250.0	0.0
4	1891521	126868	-1	453919.455	450000	0.0	11547.0	0.0
...	...	...	...	...	...	...	...	...
3840307	1036507	328243	-9	0.000	45000	NaN	0.0	NaN
3840308	1714892	347207	-9	0.000	45000	0.0	0.0	0.0
3840309	1302323	215757	-9	275784.975	585000	270000.0	270000.0	0.0
3840310	1624872	430337	-10	0.000	450000	NaN	0.0	NaN
3840311	2411345	236760	-10	0.000	157500	0.0	0.0	0.0

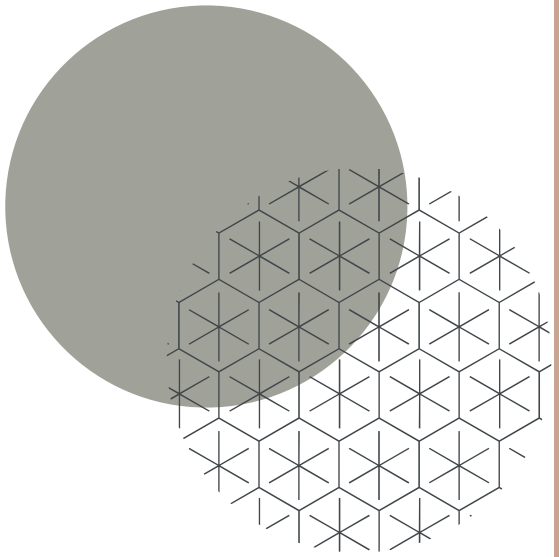


Previous_application

- All previous applications for Home Credit loans of clients who have loans
- There is one row for each previous application related to loans.

Previous_application

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PRO
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0	17145.0	SATURDAY	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN	607500.0	THURSDAY	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN	112500.0	TUESDAY	
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	NaN	450000.0	MONDAY	
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	NaN	337500.0	THURSDAY	
...	...	...	...	...	...	...	...	...	...	
1670209	2300464	352015	Consumer loans	14704.290	267295.5	311400.0	0.0	267295.5	WEDNESDAY	
1670210	2357031	334635	Consumer loans	6622.020	87750.0	64291.5	29250.0	87750.0	TUESDAY	
1670211	2659632	249544	Consumer loans	11520.855	105237.0	102523.5	10525.5	105237.0	MONDAY	
1670212	2785582	400317	Cash loans	18821.520	180000.0	191880.0	NaN	180000.0	WEDNESDAY	
1670213	2418762	261212	Cash loans	16431.300	360000.0	360000.0	NaN	360000.0	SUNDAY	



Installments_payments

- Repayment history for the previously disbursed credits
- One row for every payment that was made plus one row each for missed payment.
- One row is equivalent to one payment of one installment C one installment corresponding to one payment of one previous HC credit related to loans.

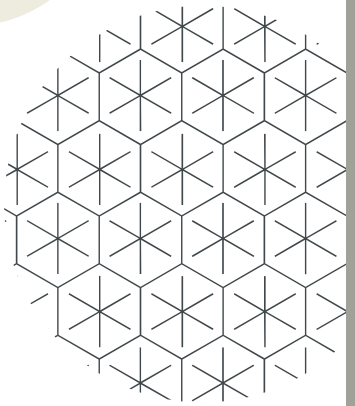
Installments_payments

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYS_INSTALMENT	DAYS_ENTRY_PAYMENT	AMT_INSTALMENT	AMT_PAYMENT
0	1054186	161674	1.0	6	-1180.0	-1187.0	6948.360	6948.360
1	1330831	151639	0.0	34	-2156.0	-2156.0	1716.525	1716.525
2	2085231	193053	2.0	1	-63.0	-63.0	25425.000	25425.000
3	2452527	199697	1.0	3	-2418.0	-2426.0	24350.130	24350.130
4	2714724	167756	1.0	2	-1383.0	-1366.0	2165.040	2160.585
...	...	...	...	...	...	...	...	...
13605396	2186857	428057	0.0	66	-1624.0	NaN	67.500	NaN
13605397	1310347	414406	0.0	47	-1539.0	NaN	67.500	NaN
13605398	1308766	402199	0.0	43	-7.0	NaN	43737.435	NaN
13605399	1062206	409297	0.0	43	-1986.0	NaN	67.500	NaN
13605400	2448869	434321	1.0	19	-27.0	NaN	11504.250	NaN

02

Data Exploration

Basic steps



Shape | Info

Check the data shape, info, columns

Duplicated

Duplicated checking

Null values

The percentage of null values of each column

Drop columns

Dropping unnecessary columns

02 Application

- Columns have more than 60% null values
- Drop these columns

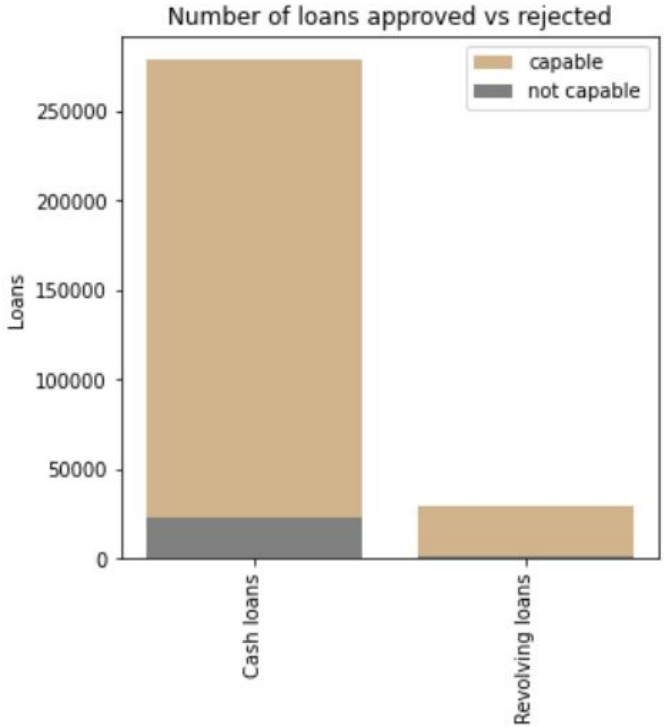
	Total	% of missing values
COMMONAREA_MEDI	92646	69.872
COMMONAREA_AVG	92646	69.872
COMMONAREA_MODE	92646	69.872
NONLIVINGAPARTMENTS_MODE	93997	69.433
NONLIVINGAPARTMENTS_AVG	93997	69.433
NONLIVINGAPARTMENTS_MEDI	93997	69.433
FONDKAPREMONT_MODE	97216	68.386
LIVINGAPARTMENTS_MODE	97312	68.355
LIVINGAPARTMENTS_AVG	97312	68.355
LIVINGAPARTMENTS_MEDI	97312	68.355
FLOORSMIN_AVG	98869	67.849
FLOORSMIN_MODE	98869	67.849
FLOORSMIN_MEDI	98869	67.849
YEARS_BUILD_MEDI	103023	66.498
YEARS_BUILD_MODE	103023	66.498
YEARS_BUILD_AVG	103023	66.498
OWN_CAR_AGE	104582	65.991

02 Application | Univariate analysis

TYPES OF LOANS

- Revolving loans : Arrangement which allows for the loan amount to be withdrawn, repaid, and redrawn again in any manner and any number of times, until the arrangement expires. Credit card loans and overdrafts are revolving loans.

- Most of the loans are Cash loans



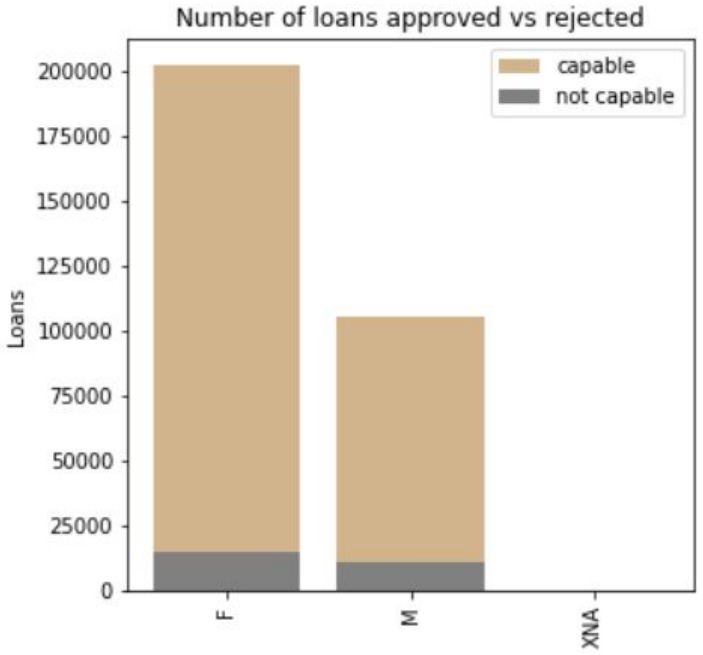
	NAME_CONTRACT_TYPE	TARGET	total	Avg
0	Cash loans	23221	278232	0.083
1	Revolving loans	1604	29279	0.055

02 Application | Univariate analysis

GENDER OF CLIENTS

- We can see that women took more number of loans when compared to men. Number of men who took loans is about half of the women who did that.

- There are 4 entries gender 'XNA'. Because it seems not to provide any information, we can remove it later on.



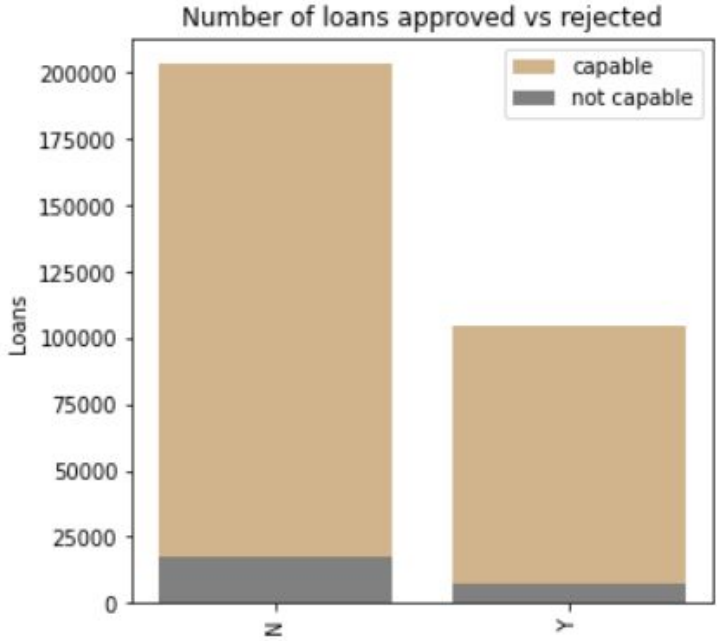
	CODE_GENDER	TARGET	total	Avg
0	F	14170	202448	0.070
1	M	10655	105059	0.101
2	XNA	0	4	0.000

02 Application | Univariate analysis

FLAG_OWN_CAR

- Most of the clients do not own car

- Since there is not much difference in the loan repayment status (8.5% and 7.2% respectively), we can conclude that this feature is not very useful -> Drop

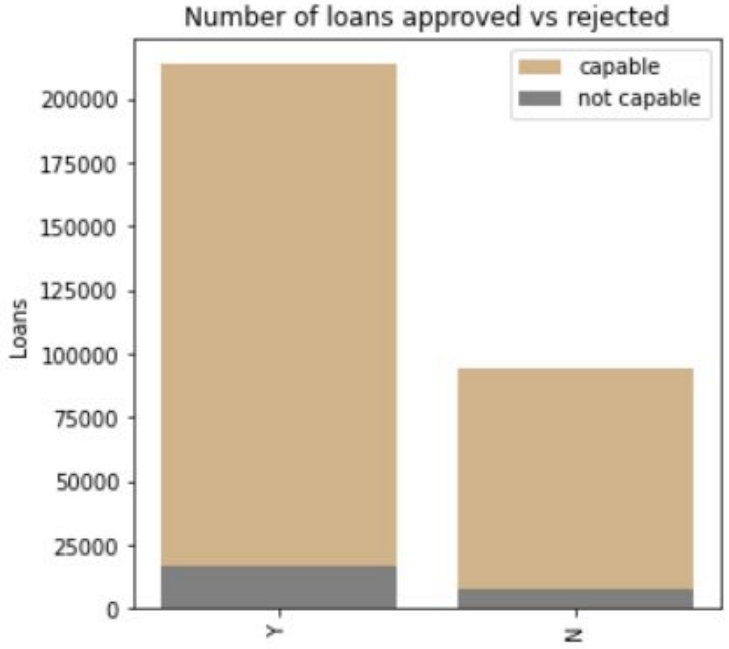


	FLAG_OWN_CAR	TARGET	total	Avg
0	N	17249	202924	0.085
1	Y	7576	104587	0.072

02 Application | Univariate analysis

FLAG_OWN_REALTY

- In contrast with the car owning, most of the clients own a house/flat.
- However, again, when taking a glance at the figures, we can conclude that this feature is not very useful
  -> Drop



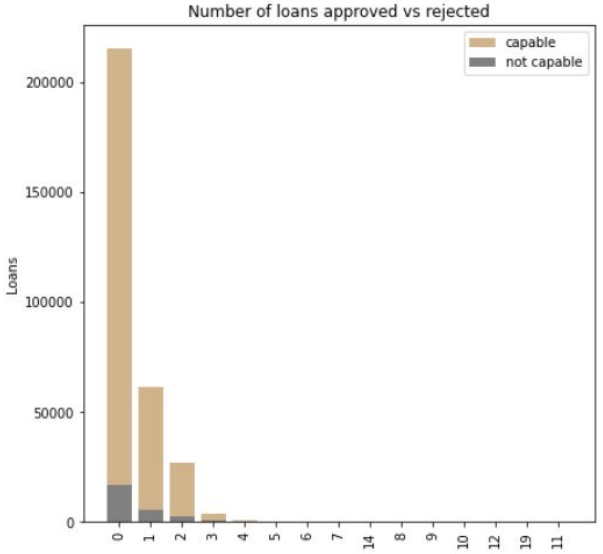
	FLAG_OWN_REALTY	TARGET	total	Avg
1	Y	16983	213312	0.080
0	N	7842	94199	0.083

02 Application | Univariate analysis

CNT_CHILDREN

- The clients who have no children account for the highest number of loans

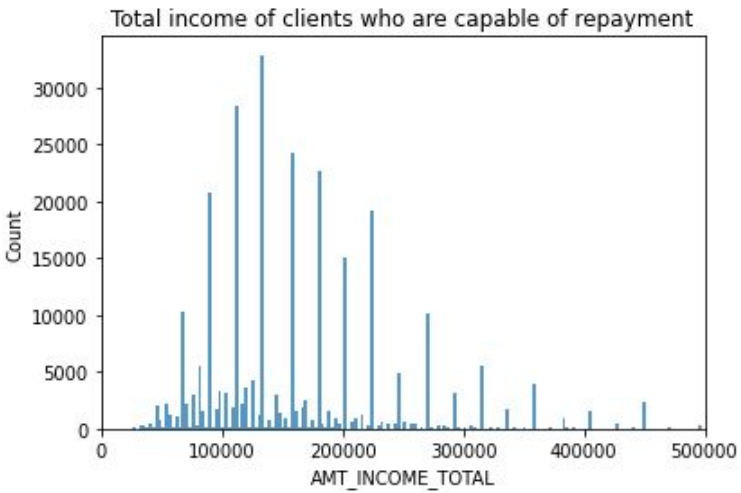
- However, the figures tell that this feature is not very useful -> Drop

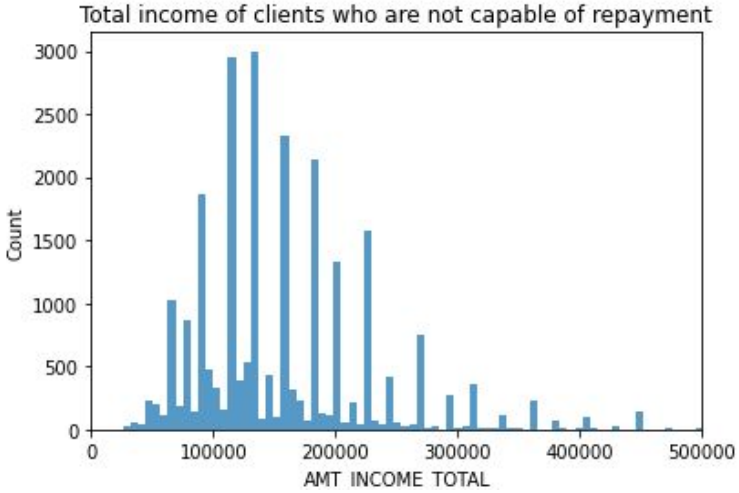


	CNT_CHILDREN	TARGET	total	Avg
0	0	16609	215371	0.077
1	1	5454	61119	0.089
2	2	2333	26749	0.087
3	3	358	3717	0.096
4	4	55	429	0.128
5	5	7	84	0.083
6	6	6	21	0.286
7	7	0	7	0.000
13	14	0	3	0.000
8	8	0	2	0.000
9	9	2	2	1.000
10	10	0	2	0.000
12	12	0	2	0.000
14	19	0	2	0.000
11	11	1	1	1.000

02 Application | Univariate analysis

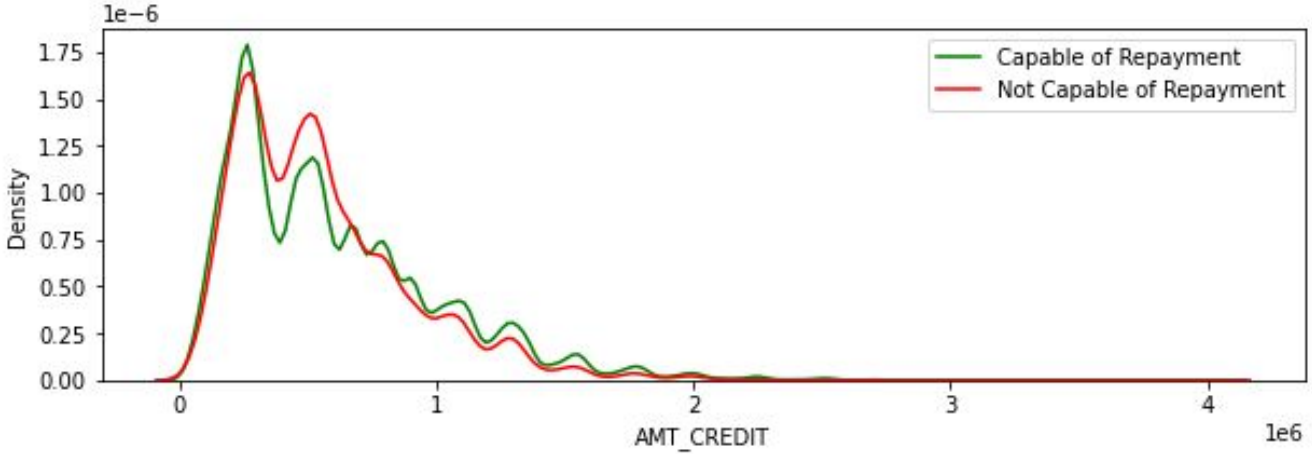
Income of the client





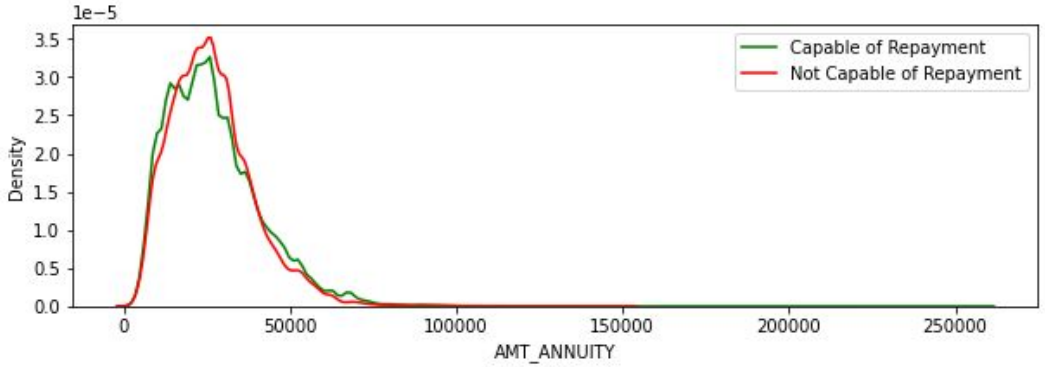
02 Application | Univariate analysis

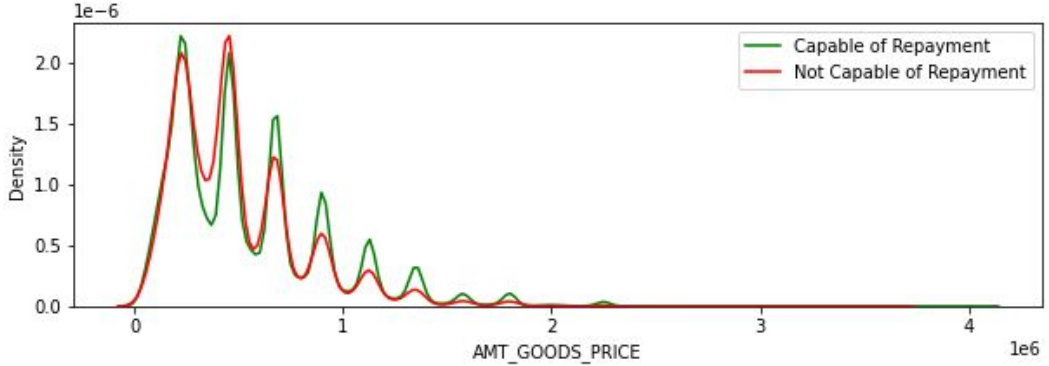
Credit amount of the loan



02 Application | Univariate analysis

Loan Annuity

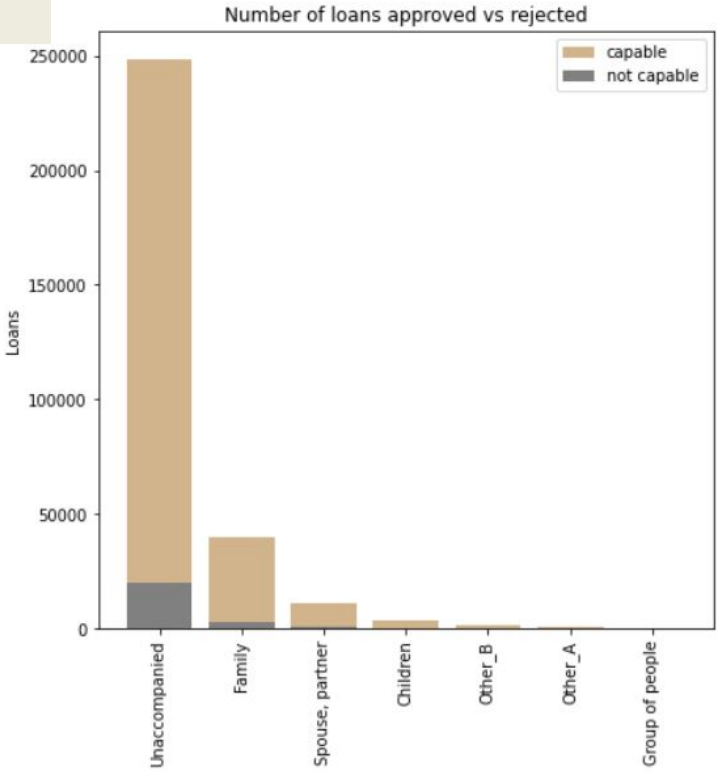




02 Application | Univariate analysis

Who was accompanying client when applying for the loan

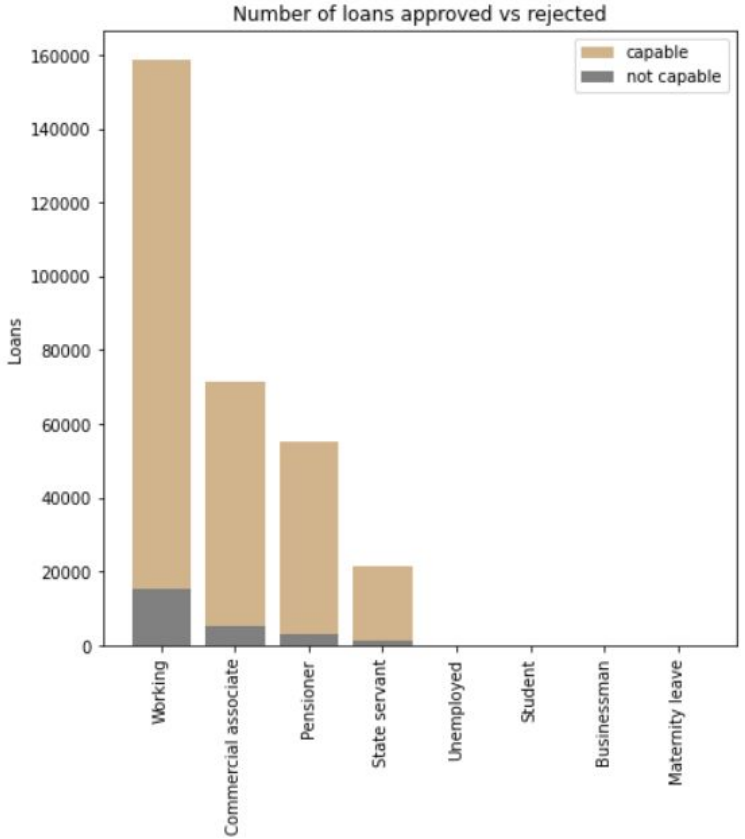
	NAME_TYPE_SUITE	TARGET	total	Avg
6	Unaccompanied	20337	248526	0.082
1	Family	3009	40149	0.075
5	Spouse, partner	895	11370	0.079
0	Children	241	3267	0.074
4	Other_B	174	1770	0.098
3	Other_A	76	866	0.088
2	Group of people	23	271	0.085



02 Application | Univariate analysis

Clients income type

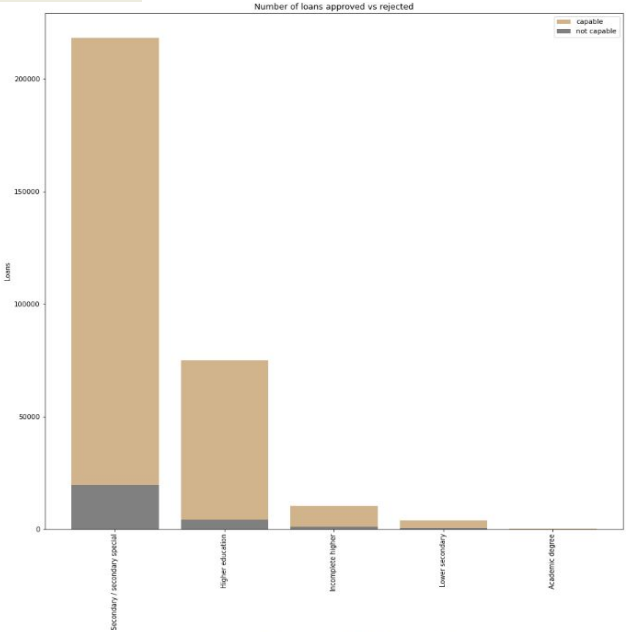
	NAME_INCOME_TYPE	TARGET	total	Avg
7	Working	15224	158774	0.096
1	Commercial associate	5360	71617	0.075
3	Pensioner	2982	55362	0.054
4	State servant	1249	21703	0.058
6	Unemployed	8	22	0.364
5	Student	0	18	0.000
0	Businessman	0	10	0.000
2	Maternity leave	2	5	0.400



02 Application | Univariate analysis

Level of highest education the client achieved

- People with Secondary/Secondary Special as the highest level of education apply for most number of loans and they are also the highest defaulters. However, the default percentage is not very different across various education levels.

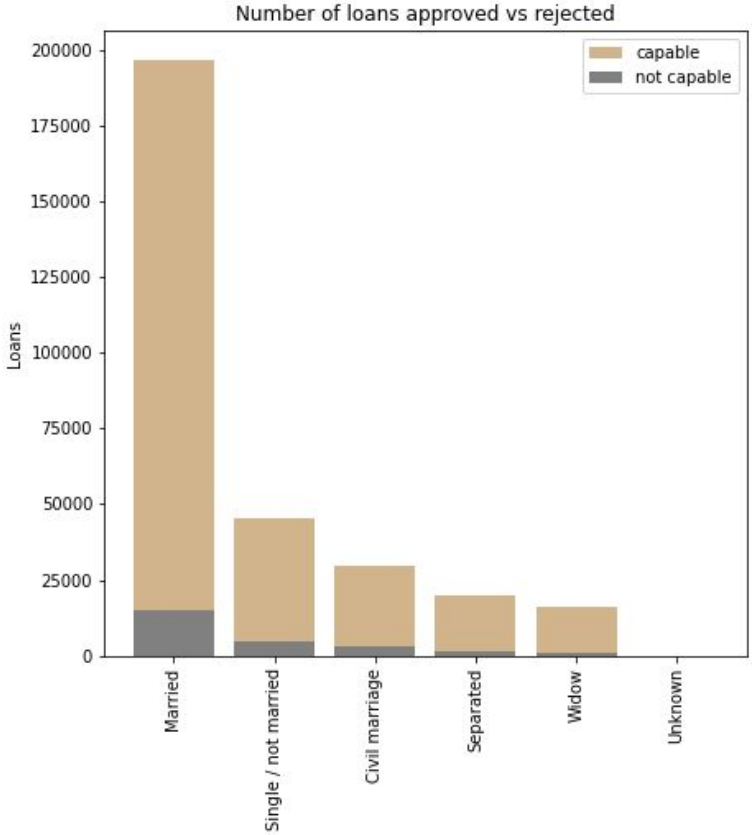


	NAME_EDUCATION_TYPE	TARGET	total	Avg
4	Secondary / secondary special	19524	218391	0.089
1	Higher education	4009	74863	0.054
2	Incomplete higher	872	10277	0.085
3	Lower secondary	417	3816	0.109
0	Academic degree	3	164	0.018

02 Application | Univariate analysis

Family status of the client

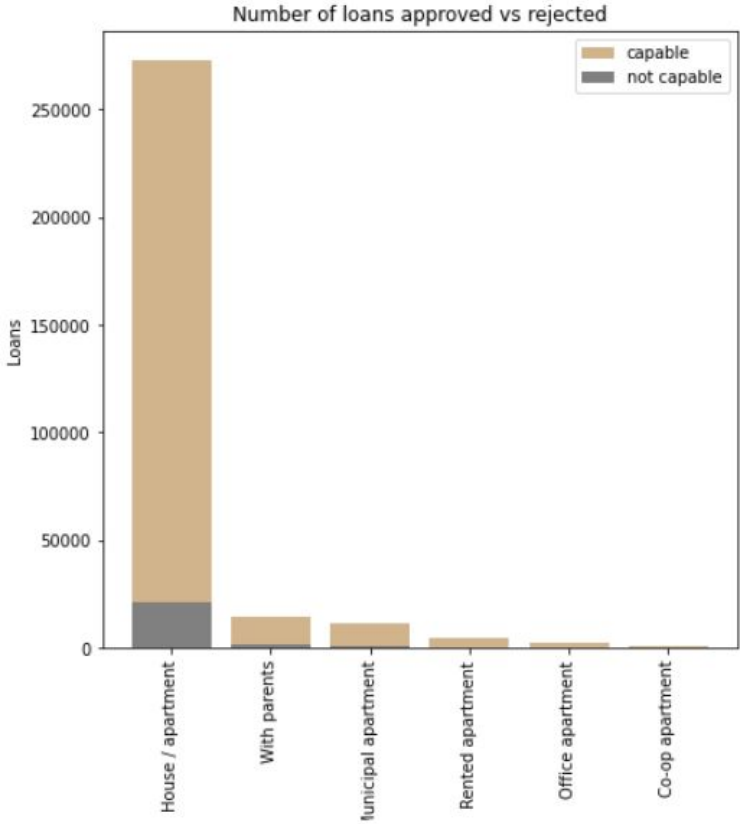
	NAME_FAMILY_STATUS	TARGET	total	Avg
1	Married	14850	196432	0.076
3	Single / not married	4457	45444	0.098
0	Civil marriage	2961	29775	0.099
2	Separated	1620	19770	0.082
5	Widow	937	16088	0.058
4	Unknown	0	2	0.000



02 Application | Univariate analysis

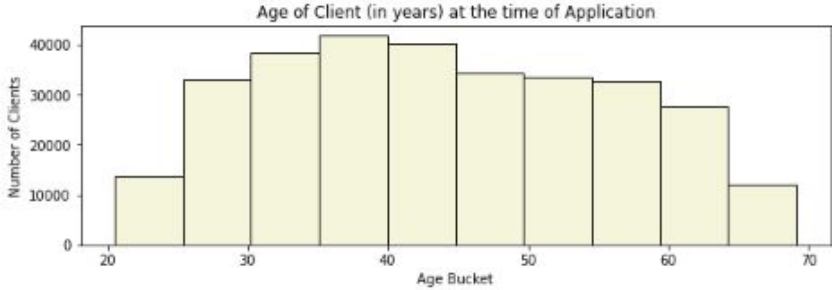
Housing situation of the client

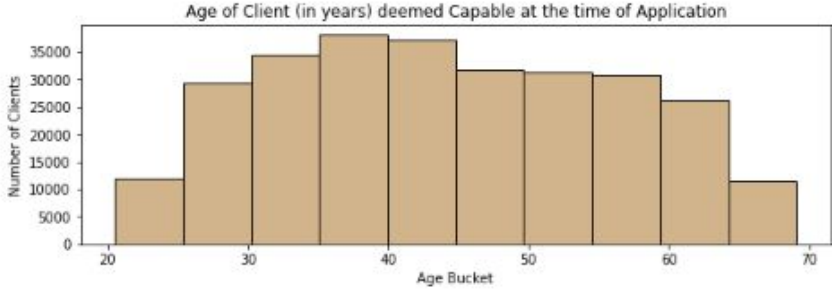
	NAME_HOUSING_TYPE	TARGET	total	Avg
1	House / apartment	21272	272868	0.078
5	With parents	1736	14840	0.117
2	Municipal apartment	955	11183	0.085
4	Rented apartment	601	4881	0.123
3	Office apartment	172	2617	0.066
0	Co-op apartment	89	1122	0.079

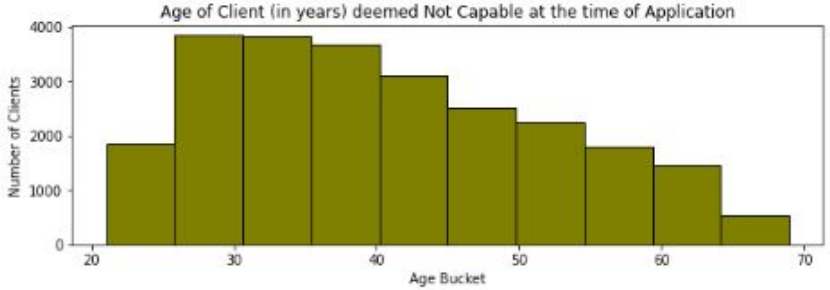


02 Application | Univariate analysis

Client's age

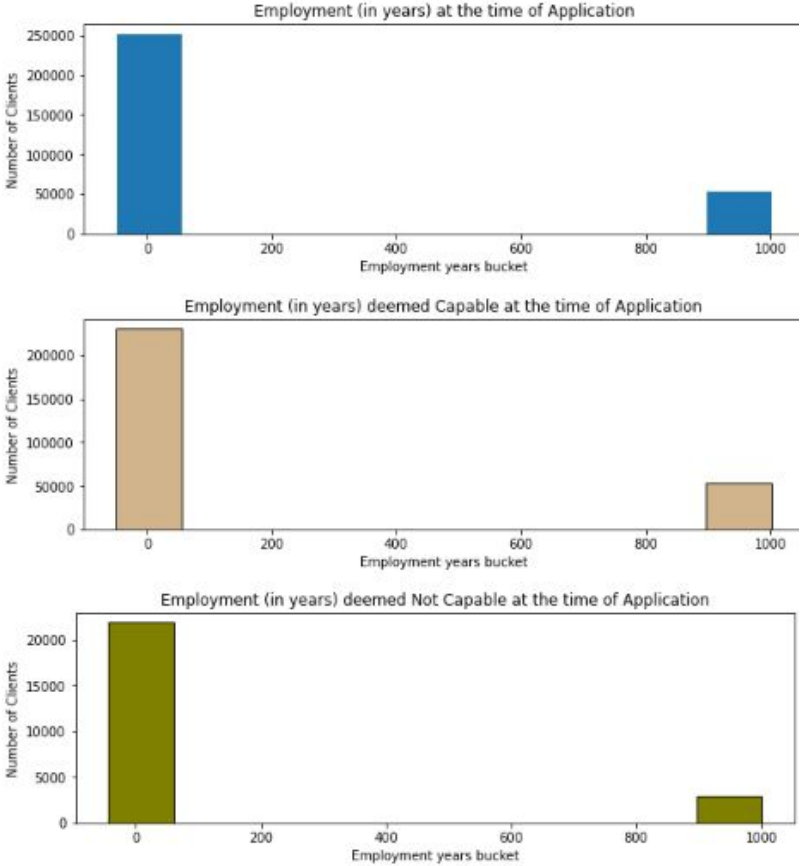






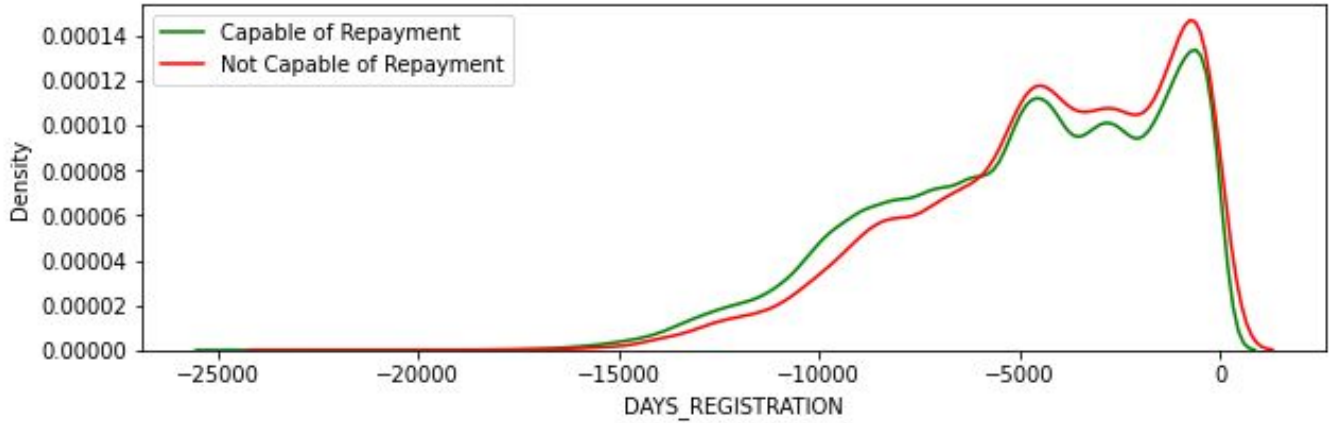
02 Application | Univariate analysis

How long the clients started the current job



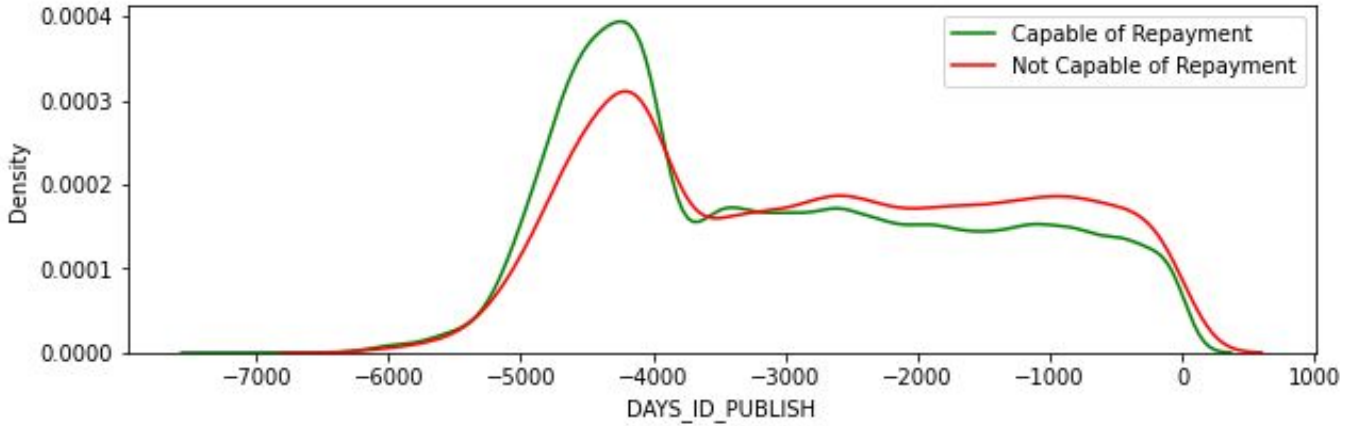
02 Application | Univariate analysis

Time clients change registration



02 Application | Univariate analysis

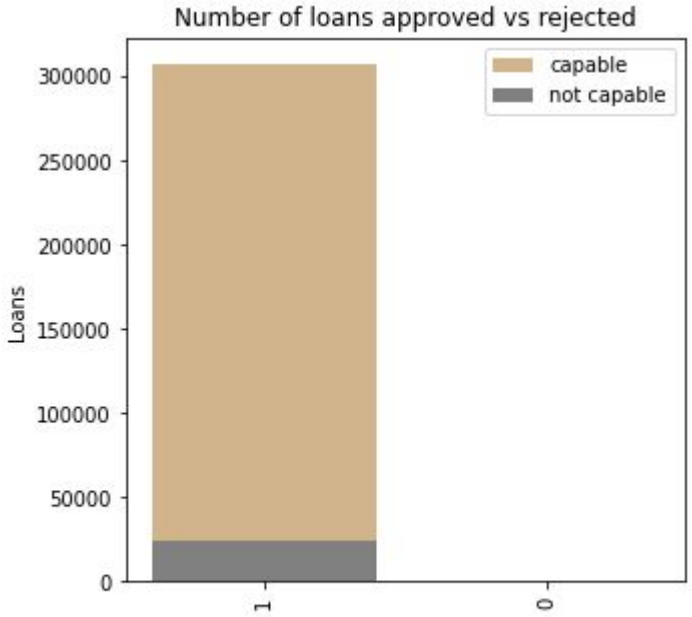
Time clients change identity document



02 Application | Univariate analysis

Clients provided mobile phone

- There is only 1 client in the training data that does not own a Mobile Phone

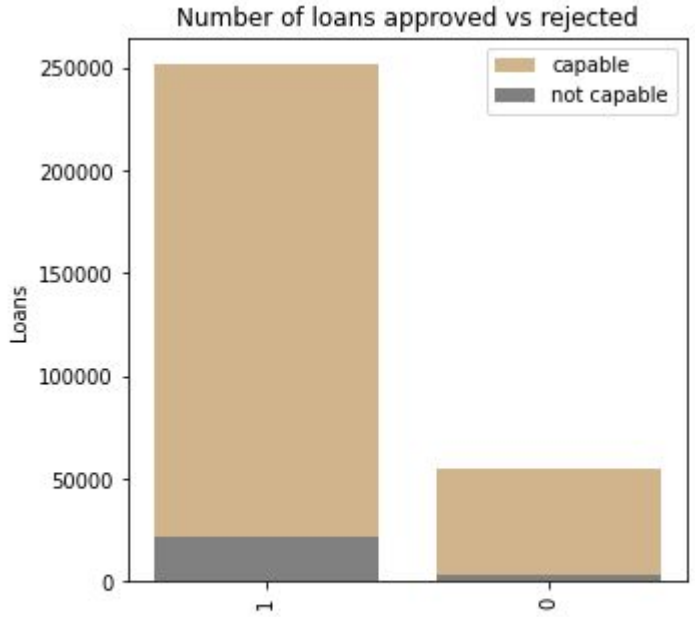


	FLAG_MOBIL	TARGET	total	Avg
1	1	24825	307510	0.081
0	0	0	1	0.000

02 Application | Univariate analysis

Clients provided work phone

- There is only 1 client in the training data that does not own work phone

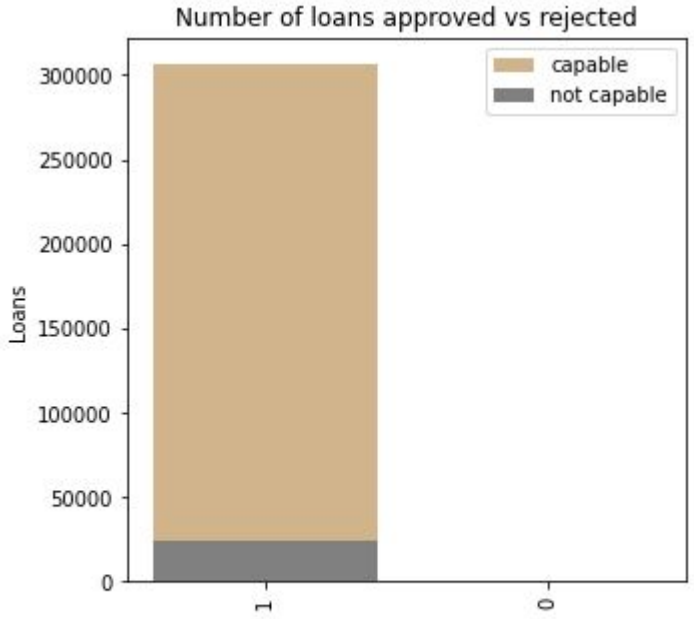


	FLAG_EMP_PHONE	TARGET	total	Avg
1	1	21834	252125	0.087
0	0	2991	55386	0.054

02 Application | Univariate analysis

Mobile phone reachable

- Most clients had their phone reachable.

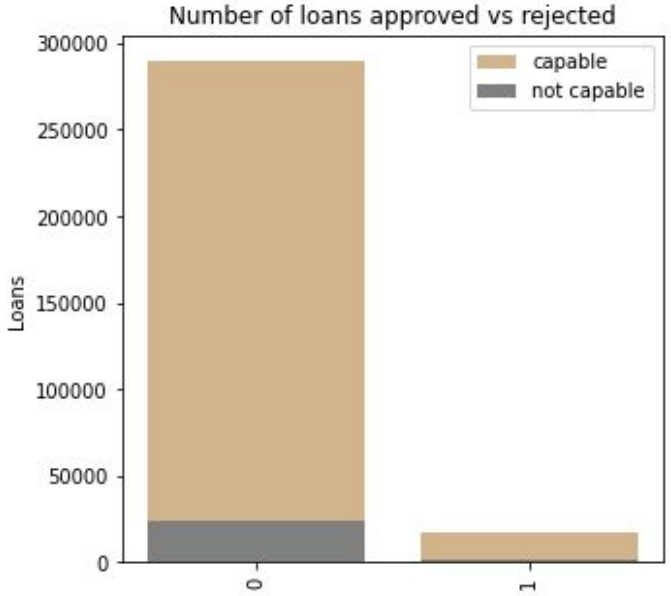


	FLAG_CONT_MOBILE	TARGET	total	Avg
1	1	24780	306937	0.081
0	0	45	574	0.078

02 Application | Univariate analysis

Clients provided email

- Most clients did not provide their emails

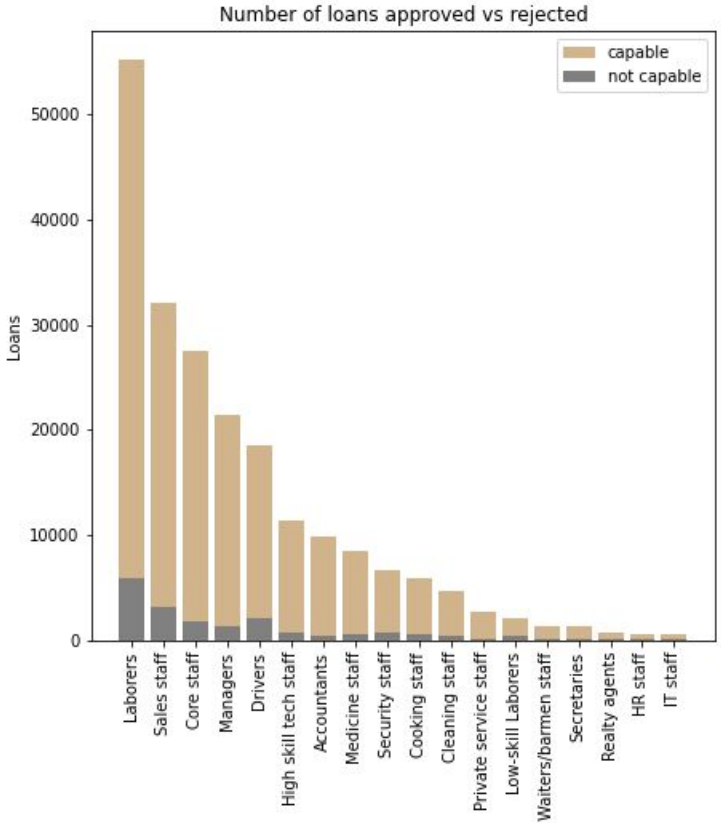


	FLAG_EMAIL	TARGET	total	Avg
0	0	23451	290069	0.081
1	1	1374	17442	0.079

02 Application | Univariate analysis

Kind of occupation clients have

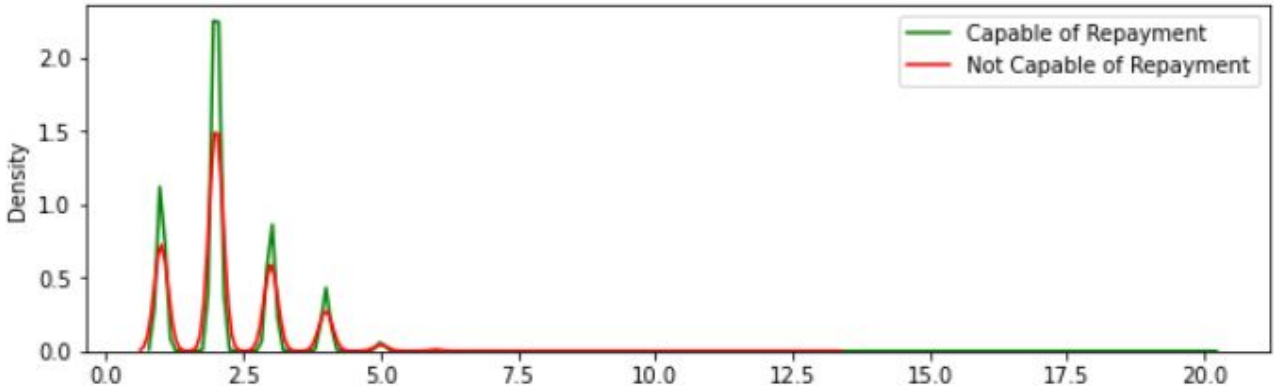
	OCCUPATION_TYPE	TARGET	total	Avg
8	Laborers	5838	55186	0.106
14	Sales staff	3092	32102	0.096
3	Core staff	1738	27570	0.063
10	Managers	1328	21371	0.062
4	Drivers	2107	18603	0.113
6	High skill tech staff	701	11380	0.062
0	Accountants	474	9813	0.048
11	Medicine staff	572	8537	0.067
16	Security staff	722	6721	0.107
2	Cooking staff	621	5946	0.104
1	Cleaning staff	447	4653	0.096
12	Private service staff	175	2652	0.066
9	Low-skill Laborers	359	2093	0.172
17	Waiters/barmen staff	152	1348	0.113
15	Secretaries	92	1305	0.070
13	Realty agents	59	751	0.079
5	HR staff	36	563	0.064
7	IT staff	34	526	0.065



02 Application | Univariate analysis

Family members client have

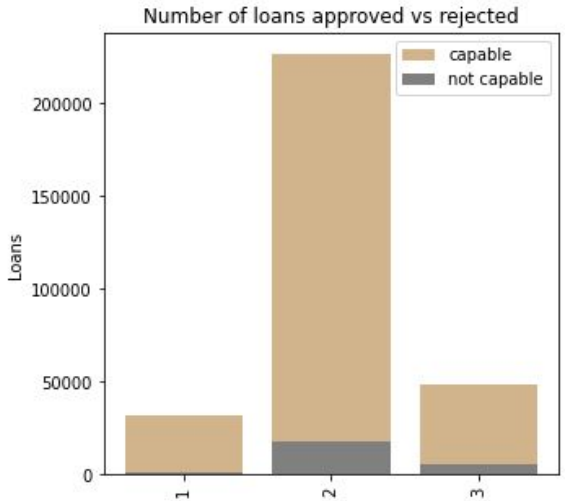
- Most of the applicants have 2 Family Members and there are very few applicants with >5 family members.



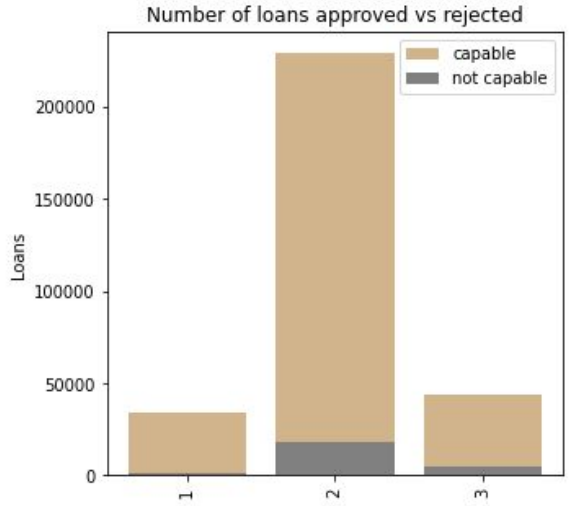
02 Application | Univariate analysis

Rating of the region where client lives

- The rating of region and rating of city are almost the same



	REGION_RATING_CLIENT	TARGET	total	Avg
0	1	1552	32197	0.048
1	2	17907	226984	0.079
2	3	5366	48330	0.111



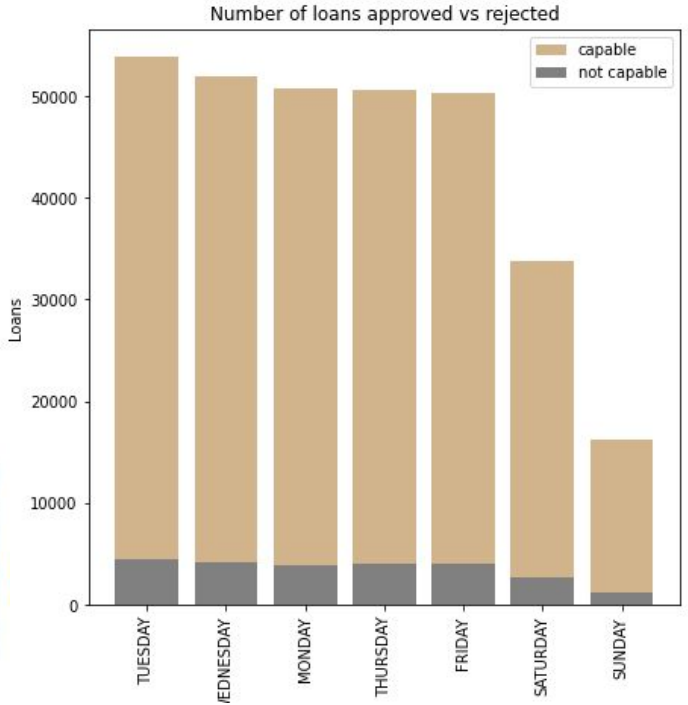
	REGION_RATING_CLIENT_W_CITY	TARGET	total	Avg
0	1	1654	34167	0.048
1	2	18170	229484	0.079
2	3	5001	43860	0.114

02 Application | Univariate analysis

Time clients start application

- This is very interesting because the number of applications are spread almost uniformly throughout the weekdays (Monday–Friday),whereas the number of applications are lower in the weekend.

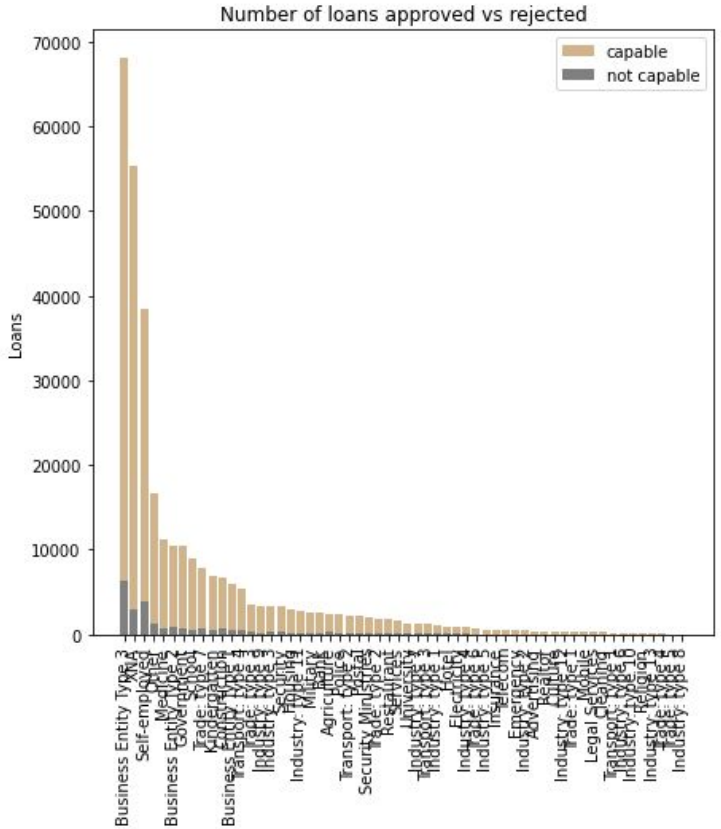
	WEEKDAY_APPR_PROCESS_START	TARGET	total	Avg
5	TUESDAY	4501	53901	0.084
6	WEDNESDAY	4238	51934	0.082
1	MONDAY	3934	50714	0.078
4	THURSDAY	4098	50591	0.081
0	FRIDAY	4101	50338	0.081
2	SATURDAY	2670	33852	0.079
3	SUNDAY	1283	16181	0.079



02 Application | Univariate analysis

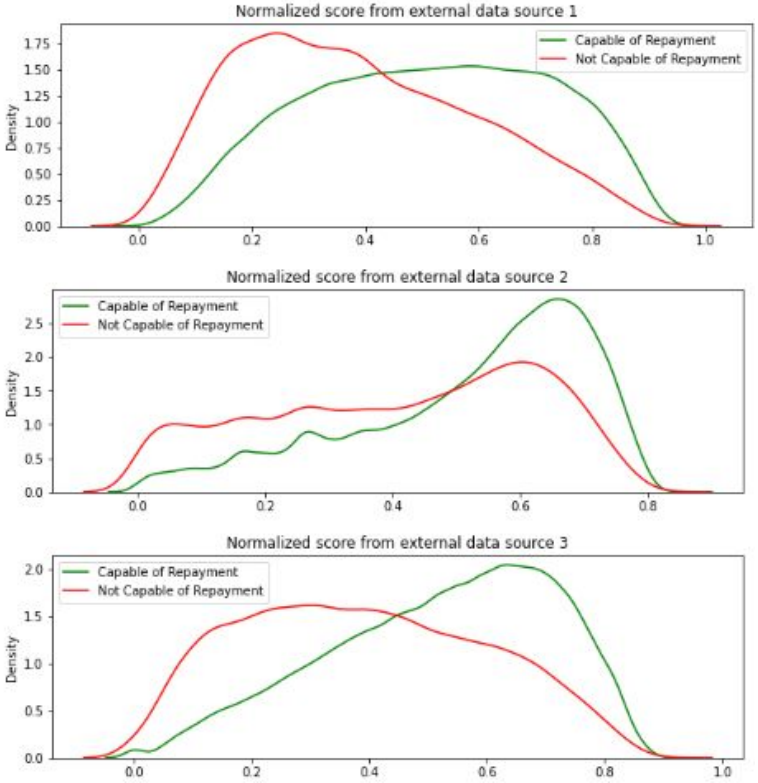
Type of organization where client works

- Business people and XNA (No information provided) are the highest number of applicants.



02 Application | Univariate analysis

Normalized score from external data source



02 Application | Univariate analysis

Documents clients provided

- The percentage of client has submitted the document is very small in most of the cases, which means that the data is highly imbalanced and its presence in the dataset is not going to help us very much.
- However, Flag_Document_3 has a good presence of number of clients provided this document and we can remove all the Flag_Document columns except this one.

	Provided	Not Provide
FLAG_DOCUMENT_2	0.004	99.996
FLAG_DOCUMENT_3	71.002	28.998
FLAG_DOCUMENT_4	0.008	99.992
FLAG_DOCUMENT_5	1.511	98.489
FLAG_DOCUMENT_6	8.806	91.194
FLAG_DOCUMENT_7	0.019	99.981
FLAG_DOCUMENT_8	8.138	91.862
FLAG_DOCUMENT_9	0.390	99.610
FLAG_DOCUMENT_10	0.002	99.998
FLAG_DOCUMENT_11	0.391	99.609
FLAG_DOCUMENT_12	0.001	99.999
FLAG_DOCUMENT_13	0.353	99.647
FLAG_DOCUMENT_14	0.294	99.706
FLAG_DOCUMENT_15	0.121	99.879
FLAG_DOCUMENT_16	0.993	99.007
FLAG_DOCUMENT_17	0.027	99.973
FLAG_DOCUMENT_18	0.813	99.187
FLAG_DOCUMENT_19	0.060	99.940
FLAG_DOCUMENT_20	0.051	99.949
FLAG_DOCUMENT_21	0.033	99.967

02 Application | Fixing Null Values and Outliers

Days_employed

The column 'Days_Employed' basically refers to the number of days before the loan application that the client started his/her first job. Since the values are almost negative, we transform the data into years and convert them to positive values to see the outliers.

```
count   307511.000
mean     63815.046
std     141275.767
min     -17912.000
25%      -2760.000
50%      -1213.000
75%       -289.000
max     365243.000
Name: DAYS_EMPLOYED, dtype: float64
```

02 Application | Fixing Null Values and Outliers

Days_employed

```
application_train.replace(max(application_train['DAYS_EMPLOYED'].values), np.nan, inplace=True)
```

```
(-application_train['DAYS_EMPLOYED'] / 365).describe(percentiles=[0.01 * i for i in range(0, 100, 5)])
```

count	252137.000
mean	6.532
std	6.406
min	-0.000
0%	0.000
5%	0.564
10%	0.912
15%	1.282
20%	1.690
25%	2.101
30%	2.518
35%	2.959
40%	3.425

45%	3.953
50%	4.515
55%	5.126
60%	5.918
65%	6.732
70%	7.649
75%	8.699
80%	10.071
85%	11.978
90%	14.611
95%	19.975
max	49.074
Name:	DAYS_EMPLOYED

02 Application | Fixing Null Values and Outliers

Days_Registration

Like Days_employed, we transform the data to check for outliers.

```
count   307511.000
mean     -4986.120
std       3522.886
min     -24672.000
25%      -7479.500
50%      -4504.000
75%      -2010.000
max          0.000
Name: DAYS_REGISTRATION, dtype: float64
```

02 Application | Fixing Null Values and Outliers

Days_Registration

```
(- application_train['DAYS_REGISTRATION'] / 365).describe()
```

This shows that both the minimum as well as the maximum days of registration are admissible and there are no outliers present in the 'Days_Registration' column.

```
count   307511.000
mean        13.661
std          9.652
min         -0.000
25%          5.507
50%         12.340
75%         20.492
max         67.595
Name: DAYS_REGISTRATION, dtype: float64
```

02 Application | Fixing Null Values and Outliers

Replace 'XNA' value in CODE_GENDER

```
application_train['CODE_GENDER'].replace('XNA', 'M', inplace=True)
```

```
F    202448
M    105063
Name: CODE_GENDER, dtype: int64
```

02 Bureau

- 2 columns 'AMT_ANNUITY', 'AMT_CREDIT_MAX_OVERDUE' have a lot of null value
- Drop these 2 columns

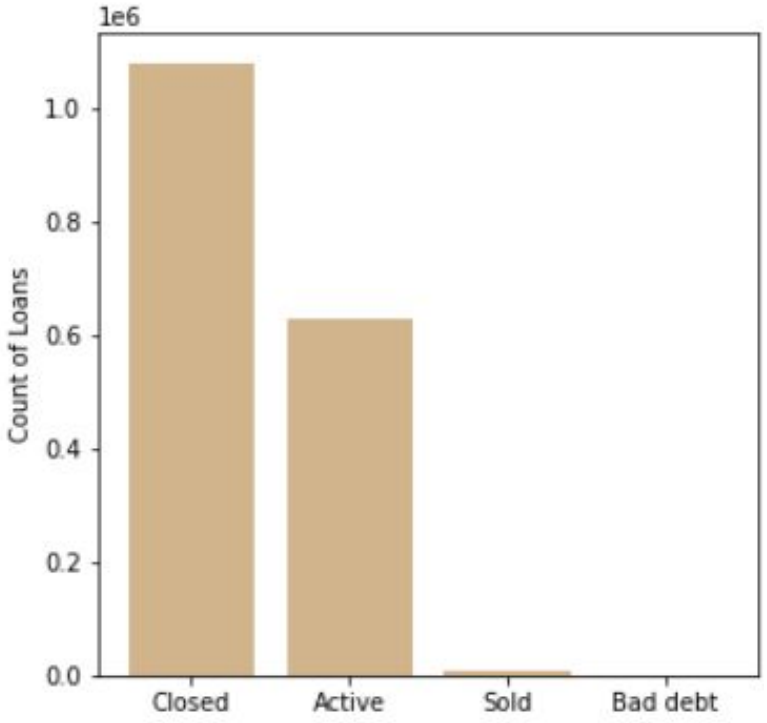
	Total	% of missing values
AMT_ANNUITY	489637	71.473490
AMT_CREDIT_MAX_OVERDUE	591940	65.513264
DAYS_ENDDATE_FACT	1082775	36.916958
AMT_CREDIT_SUM_LIMIT	1124648	34.477415
AMT_CREDIT_SUM_DEBT	1458759	15.011932
DAYS_CREDIT_ENDDATE	1610875	6.149573
AMT_CREDIT_SUM	1716415	0.000757
CREDIT_ACTIVE	1716428	0.000000
CREDIT_CURRENCY	1716428	0.000000
DAYS_CREDIT	1716428	0.000000
CREDIT_DAY_OVERDUE	1716428	0.000000
SK_ID_BUREAU	1716428	0.000000
CNT_CREDIT_PROLONG	1716428	0.000000
AMT_CREDIT_SUM_OVERDUE	1716428	0.000000
CREDIT_TYPE	1716428	0.000000
DAYS_CREDIT_UPDATE	1716428	0.000000
SK_ID_CURR	1716428	0.000000

02 Bureau | Univariate analysis

CREDIT_ACTIVE

- Most of the applications in the Bureau Data are Closed, which is followed by the status being Active.

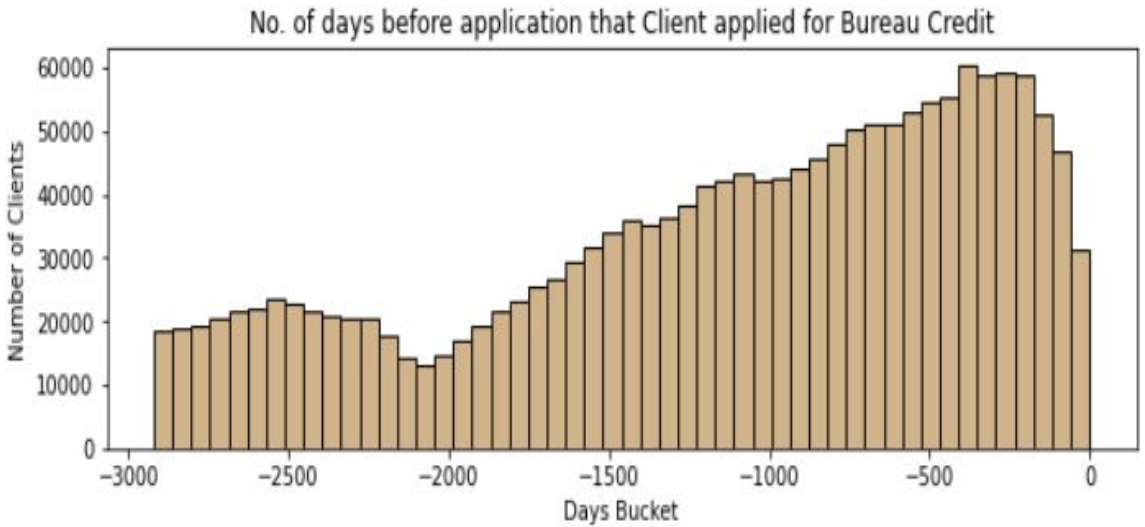
- There are very few loans that are 'Sold' or considered to be 'Bad Debt'.



02 Bureau | Univariate analysis

DAY_CREDIT

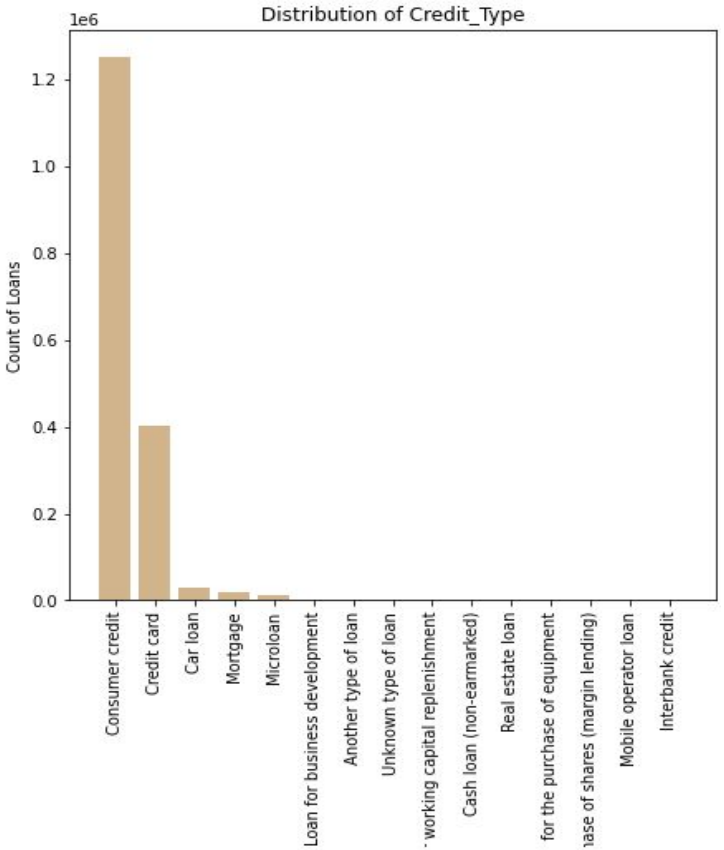
- Most of the clients applied for Bureau Credit less than 500 days before the date of loan application.



02 Bureau | Univariate analysis

CREDIT_TYPE

- Consumer Credit and Credit Cards are the mostly registered credit types in the Credit Bureau.

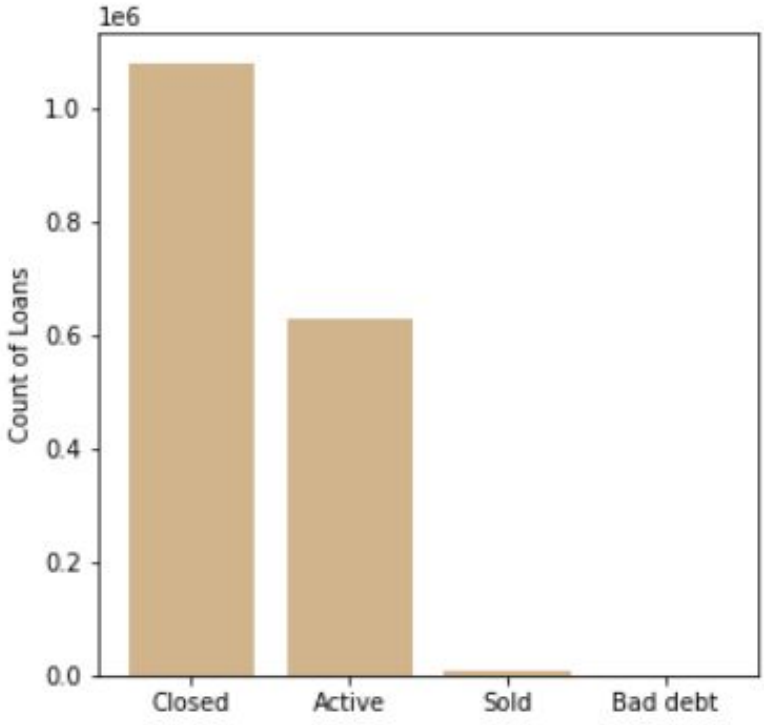


02 Bureau_balance | Univariate analysis

STATUS

- Most of the loans are Closed in the Credit Bureau, which is followed by clients with 0 DPD and then by applicants whose status is unknown.

- We can conclude that there are very few annuity defaulters in the data.



02 Credit_card_balance

- Credit_card_balance table after dropping unnecessary columns –> remain 9 columns

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	CNT_DRAWINGS_CURRENT	CNT_INSTALMENT_MATURE_CUM	NAME_CONTRACT_STATUS	SK_DPD	SK_DPD_DEF
0	2562384	378907	-6	56.970	1	35.0	Active	0	0
1	2582071	363914	-1	63975.555	1	69.0	Active	0	0
2	1740877	371185	-7	31815.225	0	30.0	Active	0	0
3	1389973	337855	-4	236572.110	1	10.0	Active	0	0
4	1891521	126868	-1	453919.455	1	101.0	Active	0	0
...	...	...	...	...	...	...	...	...	...
3840307	1036507	328243	-9	0.000	0	0.0	Active	0	0
3840308	1714892	347207	-9	0.000	0	23.0	Active	0	0
3840309	1302323	215757	-9	275784.975	2	18.0	Active	0	0
3840310	1624872	430337	-10	0.000	0	0.0	Active	0	0
3840311	2411345	236760	-10	0.000	0	21.0	Completed	0	0

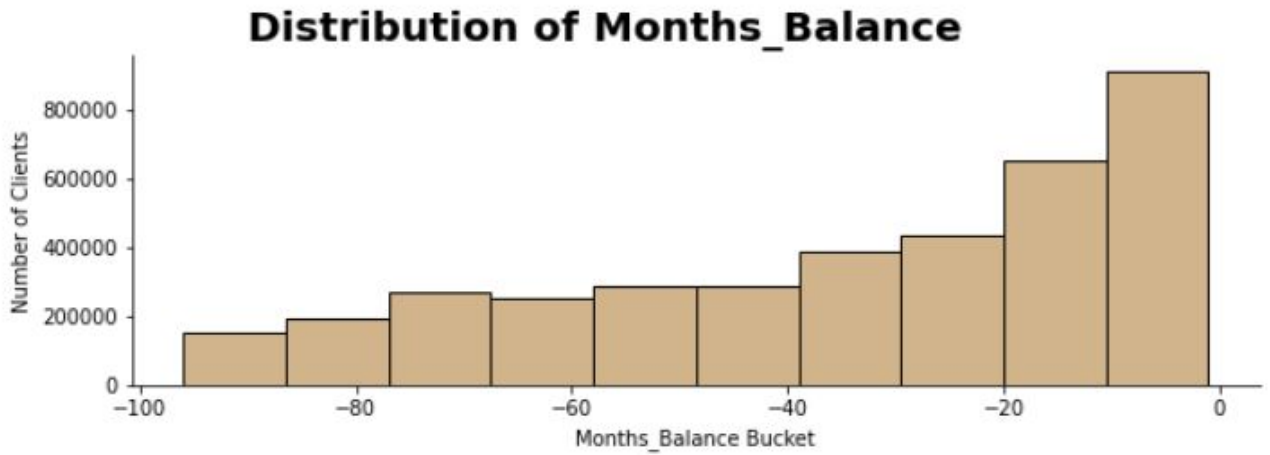
3840312 rows × 9 columns

Missing value	Total	Percent
AMT_PAYMENT_CURRENT	767988	19.998063
AMT_DRAWINGS_ATM_CURRENT	749816	19.524872
CNT_DRAWINGS_POS_CURRENT	749816	19.524872
AMT_DRAWINGS_OTHER_CURRENT	749816	19.524872
AMT_DRAWINGS_POS_CURRENT	749816	19.524872
CNT_DRAWINGS_OTHER_CURRENT	749816	19.524872
CNT_DRAWINGS_ATM_CURRENT	749816	19.524872
CNT_INSTALMENT_MATURE_CUM	305236	7.948208
AMT_INST_MIN_REGULARITY	305236	7.948208
SK_ID_PREV	0	0.000000
AMT_TOTAL_RECEIVABLE	0	0.000000
SK_DPD	0	0.000000
NAME_CONTRACT_STATUS	0	0.000000
CNT_DRAWINGS_CURRENT	0	0.000000
AMT_PAYMENT_TOTAL_CURRENT	0	0.000000
AMT_RECIVABLE	0	0.000000
AMT_RECEIVABLE_PRINCIPAL	0	0.000000
SK_ID_CURR	0	0.000000
AMT_DRAWINGS_CURRENT	0	0.000000
AMT_CREDIT_LIMIT_ACTUAL	0	0.000000
AMT_BALANCE	0	0.000000
MONTHS_BALANCE	0	0.000000
SK_DPD_DEF	0	0.000000

02 Credit_card_balance | Univariate analysis

MONTHS_BALANCE

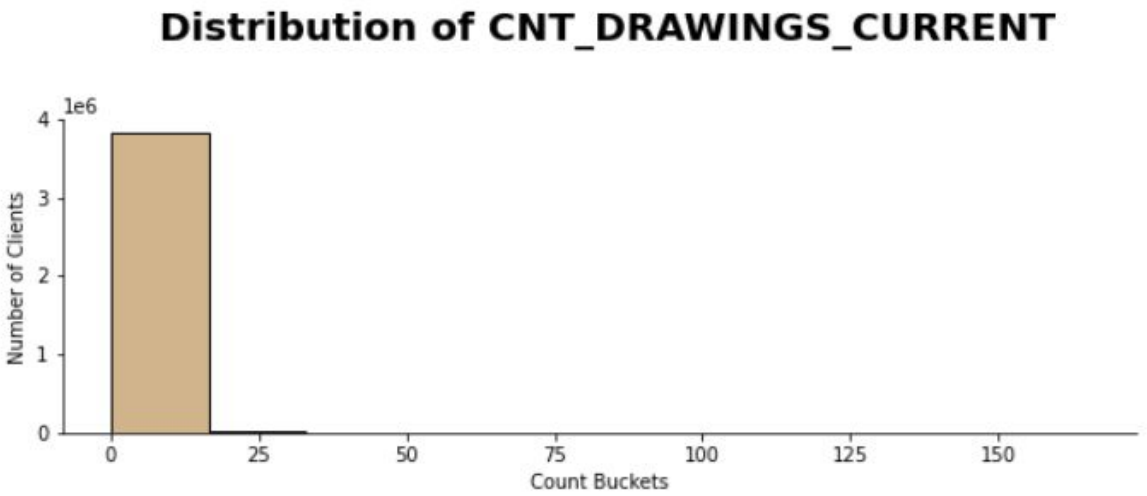
- The majority of clients have Months Balance values of 0 to 10 months before the application date.



02 Credit_card_balance | Univariate analysis

CNT_DRAWINGS_CURRENT

- With the exception of a very tiny number of outliers, the great majority of clients are drawing on their prior credit in the current month for less than 25 months.

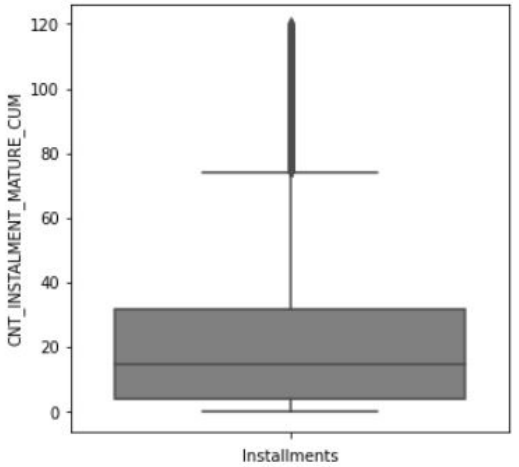


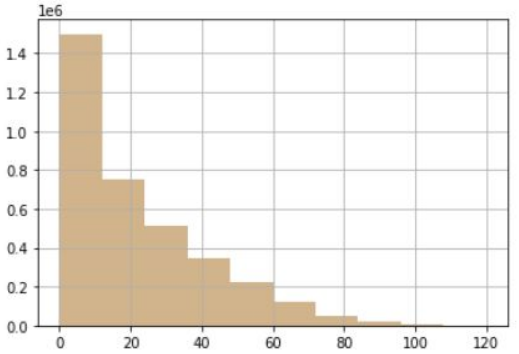
02 Credit_card_balance | Univariate analysis

CNT_INSTALMENT_MATURE_CUM

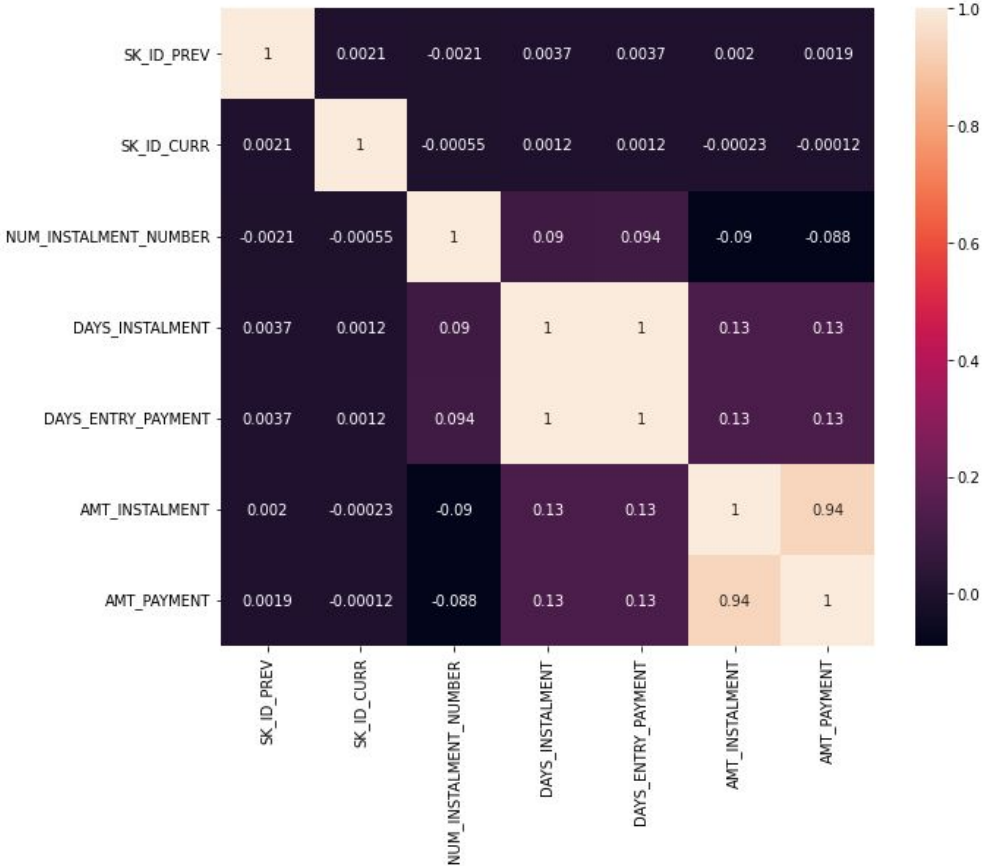
- 'Cnt_Instalment_Mature_Cum' (number of paid instalments on the previous credit), the minimum value is 0 whereas the maximum value is 120.

- 75% of the total values lying are less than 35.





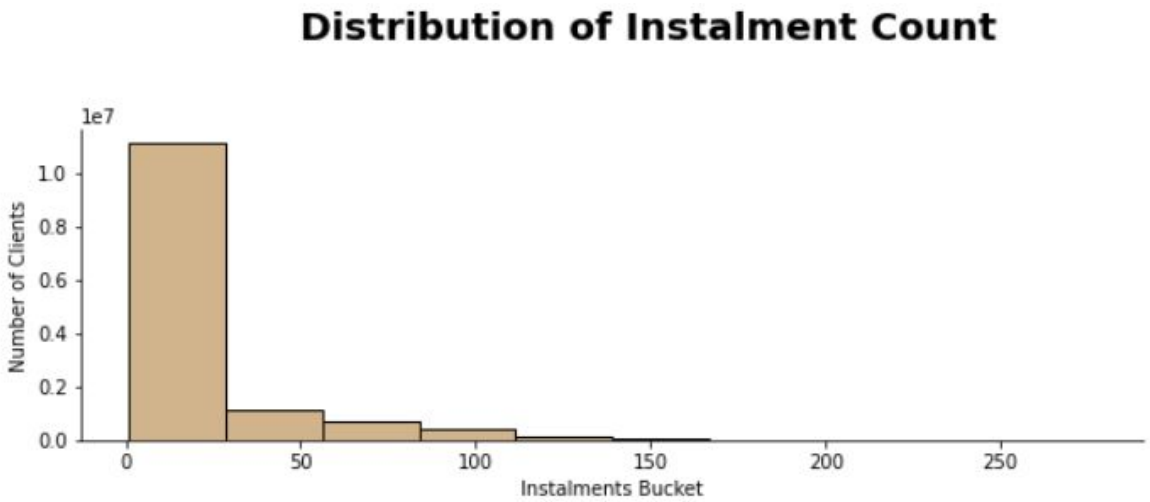
02 Installments_Payments | Correlation



02 Installments_Payments | Univariate analysis

NUM_INSTALMENT_NUMBER

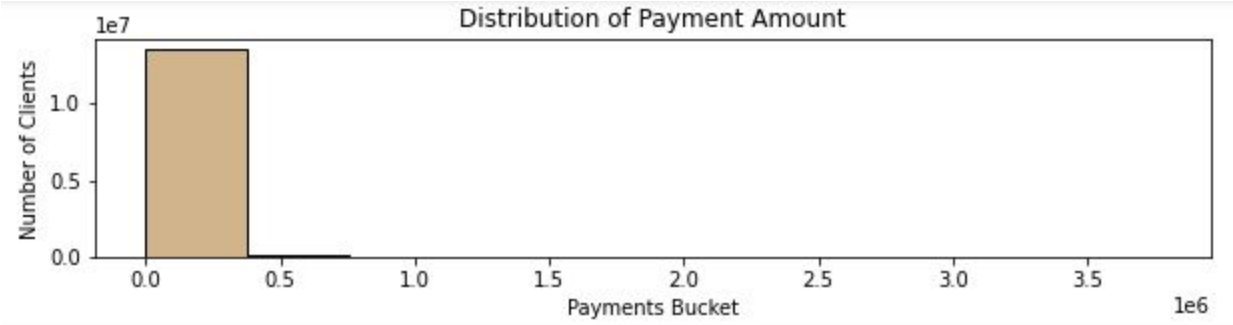
- Most of the clients complete their installment payment before 25 months.



02 Installments_Payments | Univariate analysis

AMT_PAYMENT

- Most of the clients paid less than 50000$ on previous credit on the same installment.



02 Previous_application

- 2 columns 'RATE_INTEREST_PRIVILEGED', 'RATE_INTEREST_PRIMARY' have more than 99% null value

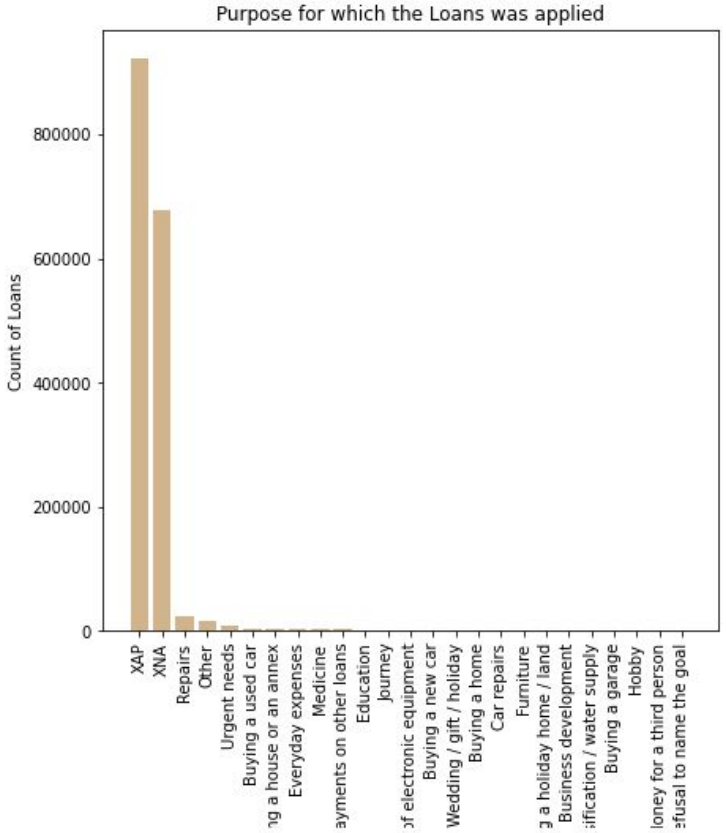
- Drop these 2 columns

	Total	% of missing values
RATE_INTEREST_PRIVILEGED	5951	99.643698
RATE_INTEREST_PRIMARY	5951	99.643698
AMT_DOWN_PAYMENT	774370	53.636480
RATE_DOWN_PAYMENT	774370	53.636480
NAME_TYPE_SUITE	849809	49.119754
NFLAG_INSURED_ON_APPROVAL	997149	40.298129
DAYS_TERMINATION	997149	40.298129
DAYS_LAST_DUE	997149	40.298129
DAYS_LAST_DUE_1ST_VERSION	997149	40.298129
DAYS_FIRST_DUE	997149	40.298129
DAYS_FIRST_DRAWING	997149	40.298129
AMT_GOODS_PRICE	1284699	23.081773
AMT_ANNUITY	1297979	22.286665
CNT_PAYMENT	1297984	22.286366

02 Previous_application | Univariate analysis

Name_Cash_Loan_Purpose

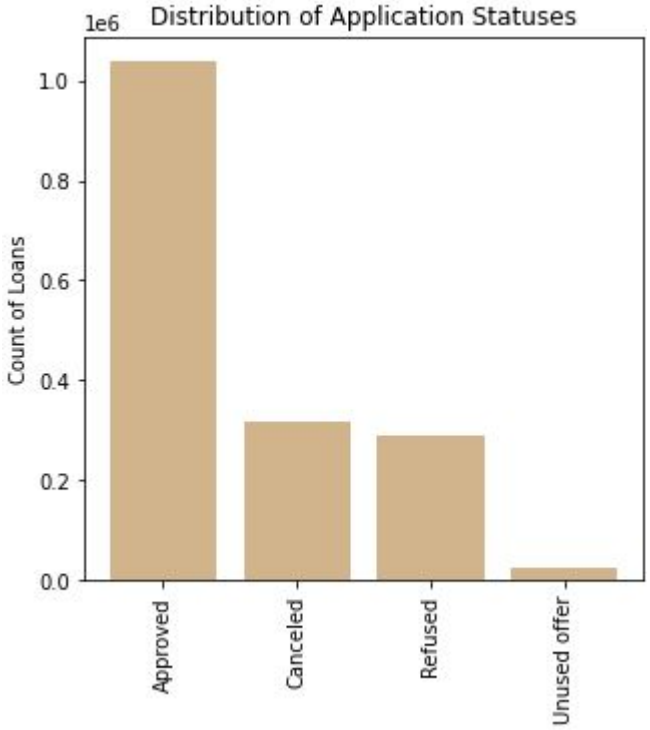
- The purpose for most of the Loan Applications is XAP, then XNA
- This may mean that the loan application purpose was not shared by the applicant, though we cannot be sure.



02 Previous_application | Univariate analysis

Name_Contract_Status

- Most of the previous applications for the clients were approved.
- This is followed by applications that were cancelled and refused.
- There were very few applications that were approved but the loans were unused by the applicant.

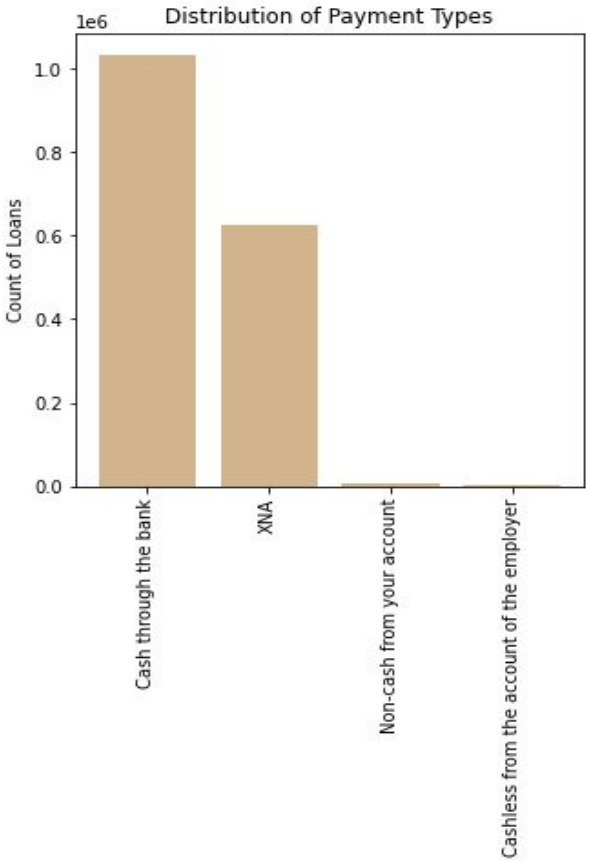


02 Previous_application | Univariate analysis

Name_Payment_Type

- The Payment Type basically refers to the Payment Method that the client chose to pay for the previous application, and as we can see here, most of the clients chose to pay via Cash through the Bank for the same.

- This is followed by people whose payment type is XNA.

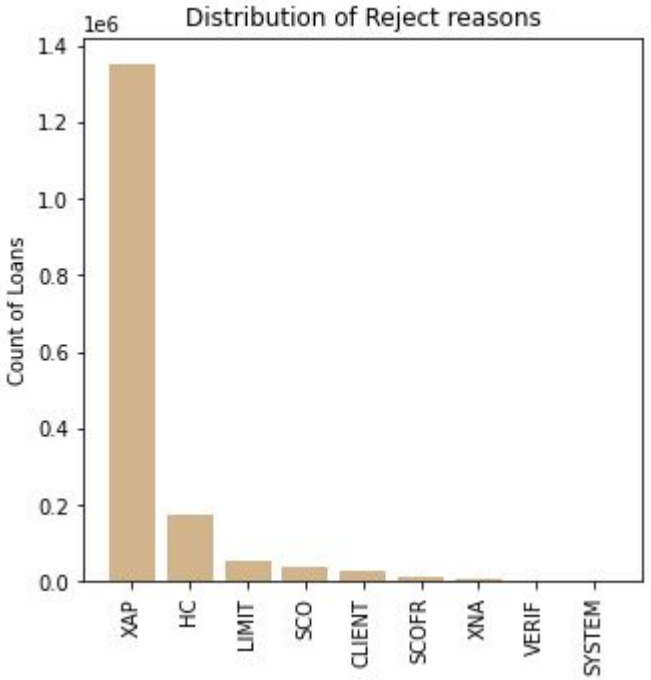


02 Previous_application | Univariate analysis

Code_Reject_Reason

- Code_Reject_Reason basically refers to the reason why the previous loan application of the client was rejected by the bank. As can be seen from here, in most of the cases XAP, was the reason provided.

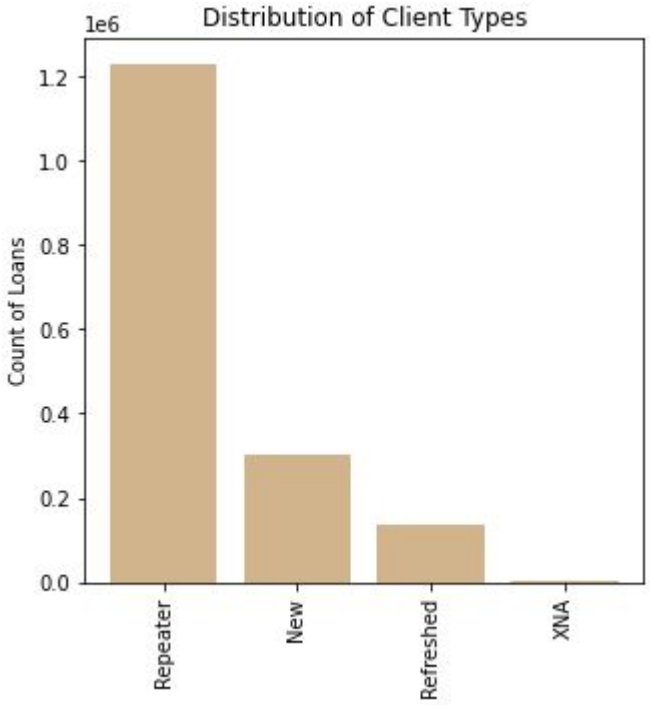
- This is followed by HC as the second most prominent reason.



02 Previous_application | Univariate analysis

Name_Client_Type

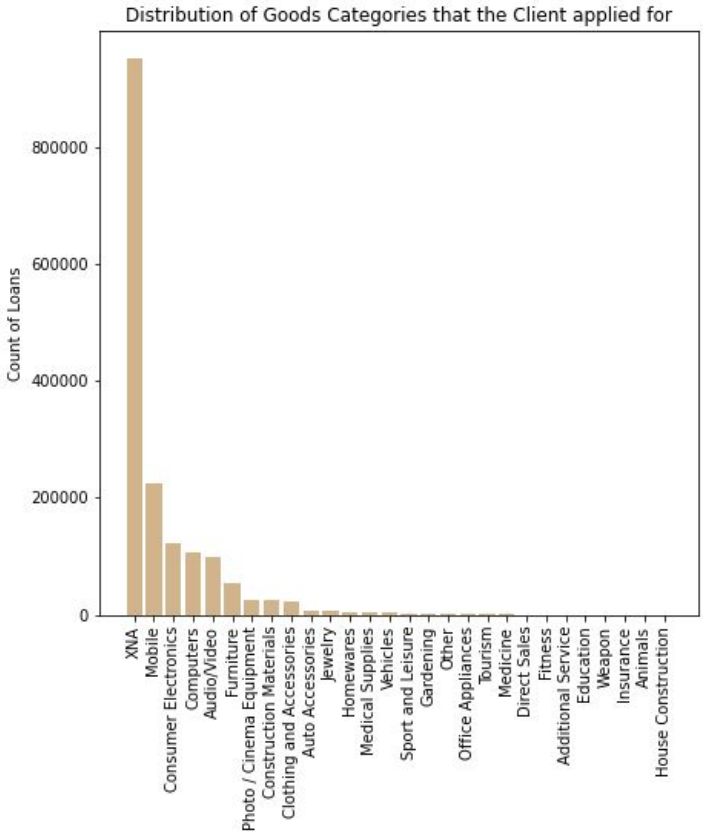
- This particular column defines whether the client was old or new when he/she was applying for the previous application. We can see from here that most of the applicants for the previous application were repeaters and there were very few first time applicants.



02 Previous_application | Univariate analysis

Name_Goods_Category

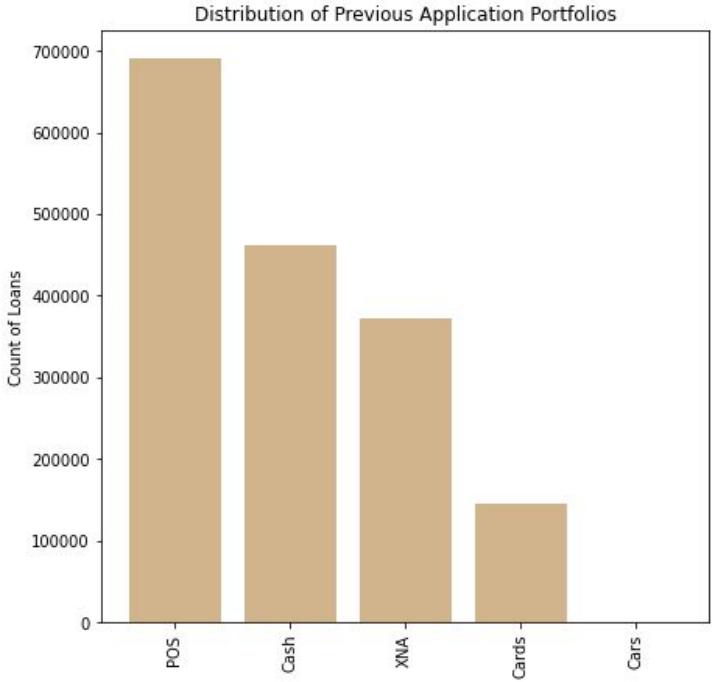
- This defines the kind of goods that the client applied for in the previous application, and as can be seen, XNA is the most popular goods category followed by Mobiles.



02 Previous_application | Univariate analysis

Name_Portfolio

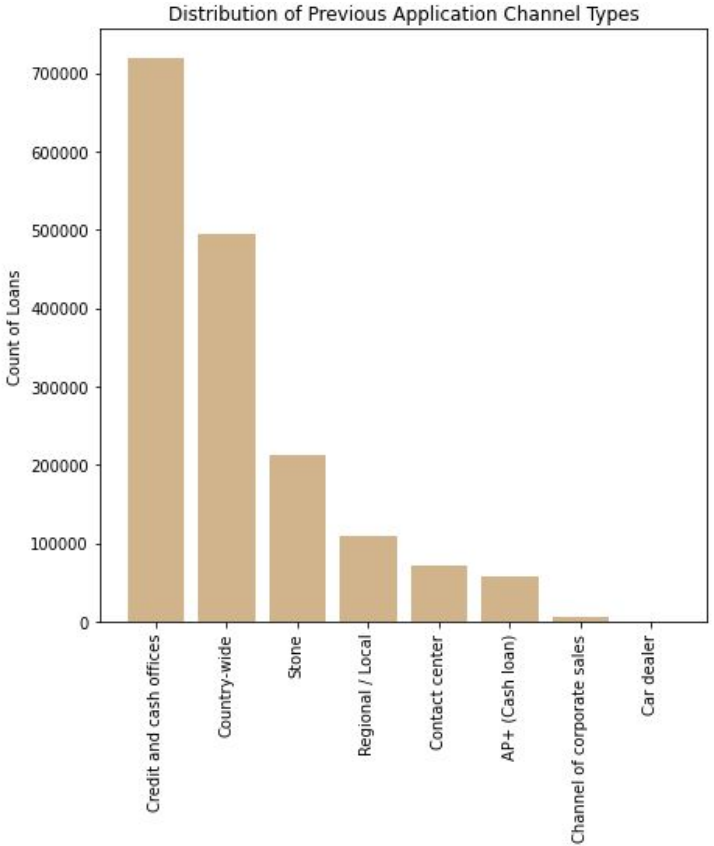
- This shows that most of the previous applications were for POS, which is followed by Cash and XNA.



02 Previous_application | Univariate analysis

Channel_Type

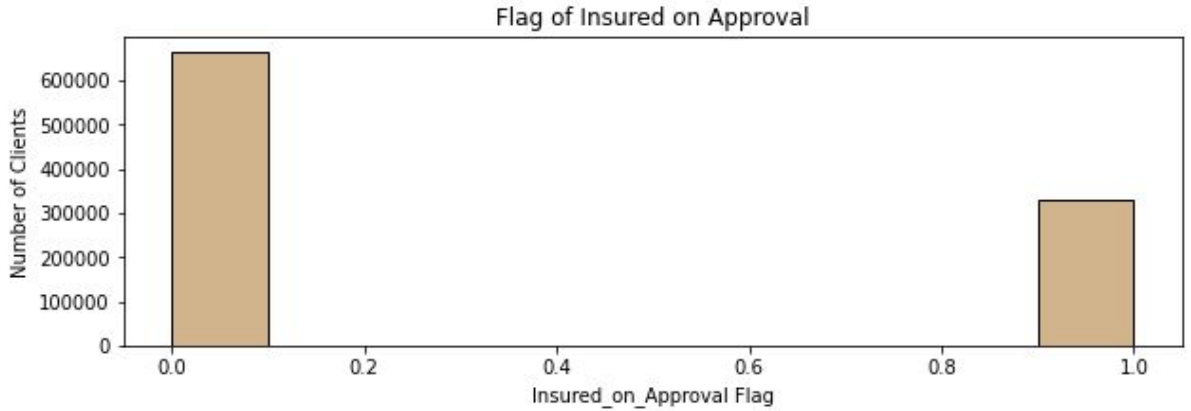
- This shows that the banks obtained most of the clients in their previous application through Credit and Cash offices, which is followed by Country-wide.



02 Previous_application | Univariate analysis

Nflag_Insured_on_Approval

- There are much fewer clients who applied for Insurance in the previous application as compared to the number of clients who did not apply for insurance.

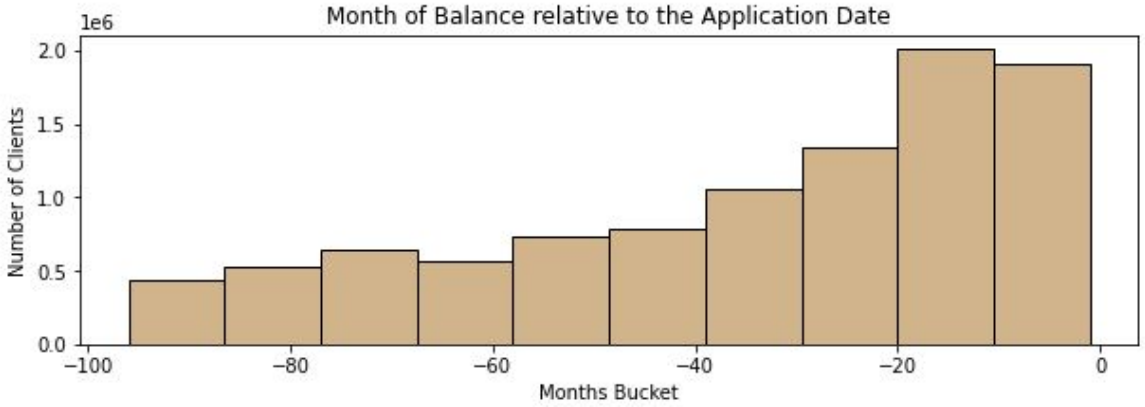


02 POS_CASH_balance | Univariate analysis

Month_Balance

- The Months_Balance for a large number of the clients is between 10 and 20 months before the date of application.

- This is followed by clients with Months_Balance less than 10 months.

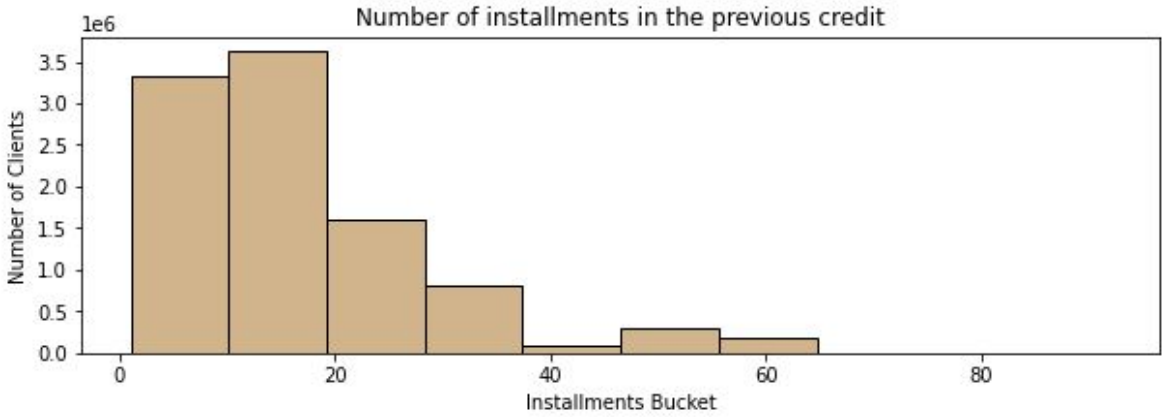


02 POS_CASH_balance | Univariate analysis

Cnt_Instalment

- The number of installments in the previous credit for most clients lies between 10 and 20

- This is followed by clients whose installment count is less than 10 months

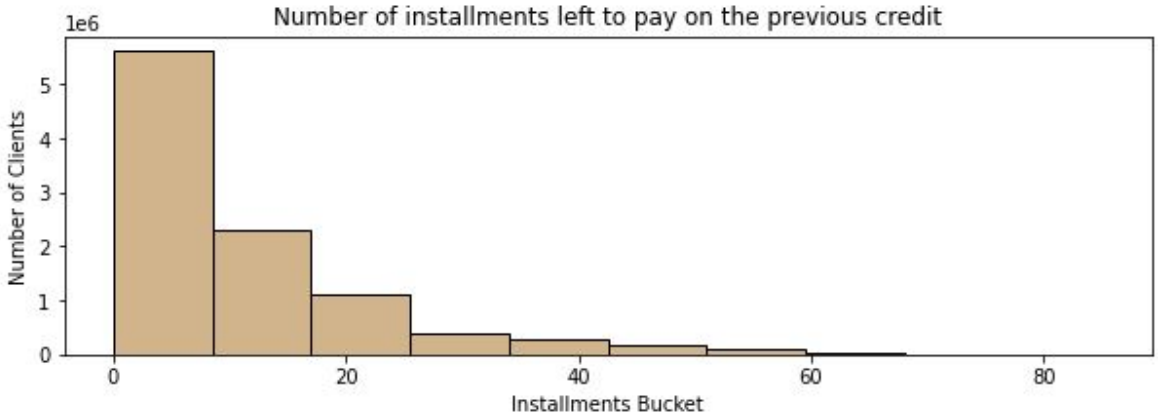


02 POS_CASH_balance | Univariate analysis

Cnt_Installment_Future

- Most of the clients have less than 10 installments left to pay on the previous credit.

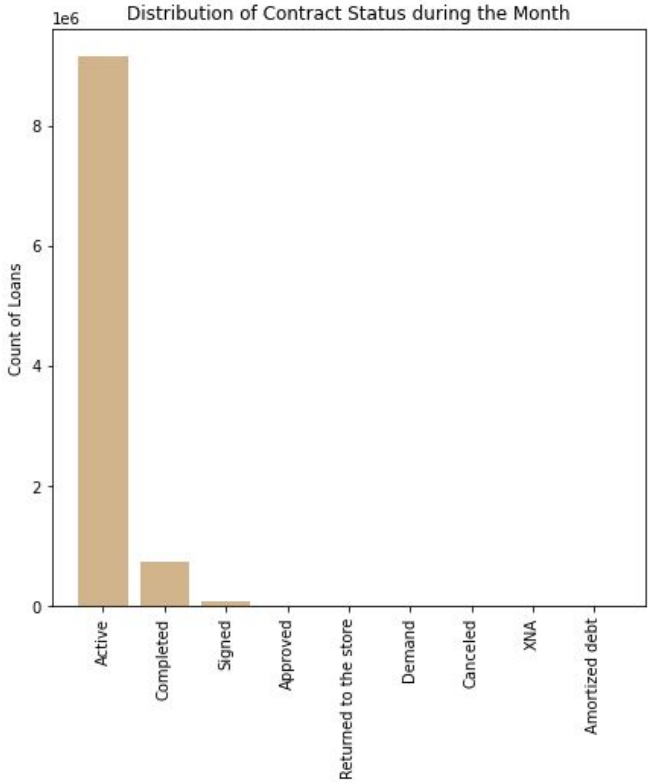
- This is followed by clients whose installment count is between 10 and 20.



02 POS_CASH_balance | Univariate analysis

Name_Contract_Status

- The Name_Contract_Status is Active in the majority of the cases.



03
Feature Engineering

03 Application| Drop columns

```
# Drop columns which unnecessary

application_train.drop(['FLAG_DOCUMENT_2','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'],axis=1, inplace=True)
```

03 Application| Fill NaN in categorical columns

Missing values

SK_ID_CURR	0
TARGET	0
NAME_CONTRACT_TYPE	0
CODE_GENDER	0
AMT_CREDIT	0
AMT_ANNUITY	0
AMT_GOODS_PRICE	0
NAME_TYPE_SUITE	1292
NAME_INCOME_TYPE	0
NAME_EDUCATION_TYPE	0
NAME_FAMILY_STATUS	0
NAME_HOUSING_TYPE	0
REGION_POPULATION_RELATIVE	0
DAYS_BIRTH	0
DAYS_EMPLOYED	0

DAYS_REGISTRATION	0
DAYS_ID_PUBLISH	0
FLAG_MOBIL	0
FLAG_EMP_PHONE	0
FLAG_WORK_PHONE	0
FLAG_CONT_MOBILE	0
FLAG_EMAIL	0
OCCUPATION_TYPE	96391
CNT_FAM_MEMBERS	0
WEEKDAY_APPR_PROCESS_START	0
ORGANIZATION_TYPE	0
EXT_SOURCE_1	0
EXT_SOURCE_2	0
EXT_SOURCE_3	0
FLAG_DOCUMENT_3	0

```
# Fill NaN in 2 categorical features by the most frequent values
application_train['NAME_TYPE_SUITE'].fillna(application_train['NAME_TYPE_SUITE'].value_counts().index[0], inplace=True)
application_train['OCCUPATION_TYPE'].fillna(application_train['OCCUPATION_TYPE'].value_counts().index[0], inplace=True)
```

03 Bureau | Drop columns

```
# drop columns with more than 99% value is 0
bureau = bureau.drop(['CREDIT_DAY_OVERDUE', 'CNT_CREDIT_PROLONG', 'AMT_CREDIT_SUM_OVERDUE'], axis = 1)
```

```
# drop columns with more than 60% null value
bureau = bureau.drop(['AMT_ANNUITY', 'AMT_CREDIT_MAX_OVERDUE'], axis = 1)
```

```
# drop columns which is categoricate
bureau = bureau.drop(['CREDIT_ACTIVE', 'CREDIT_CURRENCY', 'CREDIT_TYPE'], axis = 1)
```

03 Credit_card_balance| Drop columns

- Finding high correlated columns (>0.9)
- Finding high percentage of null value columns

→ Drop

```
to_drop = []

# drop AMT_PAYMENT_CURRENT instead of AMT_PAYMENT_TOTAL_CURRENT
# since percentage of null values of AMT_PAYMENT_CURRENT (19.9%) > AMT_PAYMENT_TOTAL_CURRENT (0%)
to_drop.append('AMT_PAYMENT_CURRENT')

# High correlated feature and can be replaced by AMT_PAYMENT_TOTAL_CURRENT
to_drop.append('AMT_RECEIVABLE_PRINCIPAL')
to_drop.append('AMT_RECIVABLE')
to_drop.append('AMT_BALANCE')
# High percentage of null values and can be represented by CNT_DRAWINGS_CURRENT
to_drop.append('CNT_DRAWINGS_POS_CURRENT')
to_drop.append('CNT_DRAWINGS_OTHER_CURRENT')
to_drop.append('CNT_DRAWINGS_ATM_CURRENT')

# The same reason can be applied to AMT_DRAWINGS_CURRENT
to_drop.append('AMT_DRAWINGS_POS_CURRENT')
to_drop.append('AMT_DRAWINGS_OTHER_CURRENT')
to_drop.append('AMT_DRAWINGS_ATM_CURRENT')
```

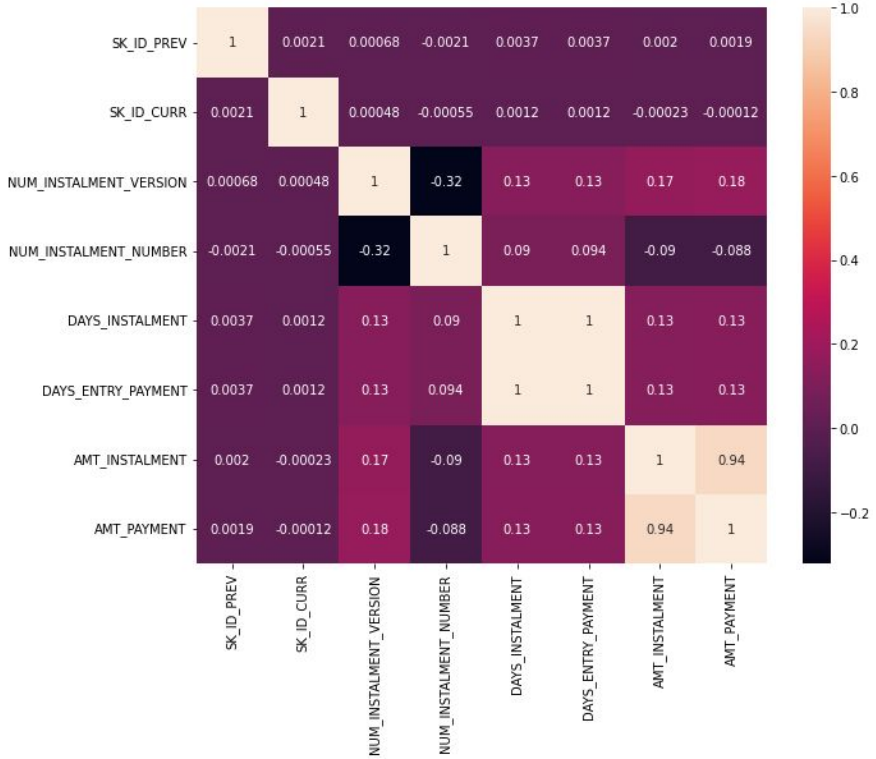
03 Credit_card_balance| Drop columns

- Delete existing column in POS_CASH_balance

```
to_drop.append('NAME_CONTRACT_STATUS')
to_drop.append('SK_DPD')
to_drop.append('SK_DPD_DEF')
```

03 Installments_payments | Drop columns

- Drop high correlated columns :
  'DAYS_ENTRY_PAYMENT',
  ' AMT_PAYMENT'



03 Previous_application | Drop columns

```
#Drop unnecessary columns

drop_list = ['AMT_DOWN_PAYMENT', 'SELLERPLACE_AREA', 'CNT_PAYMENT',
             'PRODUCT_COMBINATION', 'DAYS_FIRST_DRAWING',
             'DAYS_FIRST_DUE','DAYS_LAST_DUE_1ST_VERSION',
             'DAYS_LAST_DUE','DAYS_TERMINATION','NFLAG_INSURED_ON_APPROVAL']
df_prev.drop(drop_list, axis = 1, inplace = True)
```

04

Feature Selection

Merging

Selection

04 Merging

Bureau & Bureau_balance

Merge bureau dataframe with related information in bureau_balance file

Previous_application

These 4 datasets have their own key to for internal mapping – SK_ID_PREV

04 Merging | Bureau & Bureau_balance

Problems

Bureau dataframe comes from the Credit Bureau authority and displays one row for each credit the client from train/test dataset has taken previously. It is matched by SK_ID_CURR with train/test and where in train/test the SK_ID_CURR do not duplicate (1 for 1 client whom we are trying to classify) in most cases bureau dataframe has multiple indices of the same client as he/she had applied to multiple loans previously.

04 Merging | Bureau & Bureau_balance

Problems

In turn bureau_balance even more extends the previous credit information on a greater scale. It contains a separate row for each month of history of every previous credit reported to Credit Bureau (bureau dataframe) and is related to bureau df via SK_ID_BUREAU. So the approach we are going to use is to calculate mean of each statistical column out of these both dataframes to include these mean values as features of our clients whom we are trying to classify.

04 Merging | Bureau & Bureau_balance

Steps

1. Collapse bureau_balance dataframe to mean values grouped by SK_ID_BUREAU
2. Merge this with bureau dataframe
3. Collapse bureau dataframe to mean values grouped by SK_ID_CURR
4. Merge what we've got with our data dataframe

04 Merging | Bureau & Bureau_balance

Step 1 - collapse bureau_balance

```
# first define the formula for grouping rows by ID and calculating mean values
def extract_mean(x):
    y = x.groupby('SK_ID_BUREAU', as_index=False).mean()
    return y
```

```
# apply formula to create bureau_balance dataframe grouped
# by SK_ID_BUREAU with mean values of all numerical columns

bureau_balance_mean = extract_mean(bureau_balance)
```

04 Merging | Bureau & Bureau_balance

Step 2 - merge with Bureau

```
bureau = bureau.merge(bureau_balance_mean, on = 'SK_ID_BUREAU', how = 'left')
bureau.drop('SK_ID_BUREAU', axis = 1, inplace = True) # we don't need this internal ID anymore
```

04 Merging | Bureau & Bureau_balance

Step 3 - collapse bureau

```
def extract_mean(x):
    y = x.groupby('SK_ID_CURR', as_index=False).mean()
    return y

bureau_mean_values = extract_mean(bureau)
```

	SK_ID_CURR	DAYS_CREDIT	DAYS_CREDIT_ENDDATE	DAYS_ENDDATE_FACT	AMT_CREDIT_SUM	AMT_CREDIT_SUM_DEBT	AMT_CREDIT_SUM_LIMIT	DAYS_CREDIT_UPDATE	MONTHS_BALANCE
0	100001	-735.000000	82.428571	-825.500000	207623.571429	85240.928571	0.00000	-93.142857	-11.785714
1	100002	-874.000000	-349.000000	-697.500000	108131.945625	49156.200000	7997.14125	-499.875000	-21.875000
2	100003	-1400.750000	-544.500000	-1097.333333	254350.125000	0.000000	202500.00000	-816.000000	NaN
3	100004	-867.000000	-488.500000	-532.500000	94518.900000	0.000000	0.00000	-532.000000	NaN
4	100005	-190.666667	439.333333	-123.000000	219042.000000	189469.500000	0.00000	-54.333333	-3.000000

04 Merging | Bureau & Bureau_balance

Step 4 - merge Bureau with data

```
data = data.merge(bureau_mean_values, on = 'SK_ID_CURR', how = 'left')
```

04 Merging | Previous_application

Steps

1. Collapse credit_card_balance dataframe to mean values grouped by SK_ID_PREV
2. Merge with previous_application (our 'leading' dataset in this case)
3. Collapse POS_CASH_balance to mean values grouped by SK_ID_PREV
4. Merge with previous_application
5. Collapse installments_payments
6. Merge with previous_application
7. Collapse the resulting previous_application dataset to mean values grouped by SK_ID_CURR
8. Merge our unfolded previous_application statistics with our data

04 Merging | Previous_application

Problems

- We will delete the SK_ID_CURR from the ‘credit_card_balance’ / ‘POS_CASH_balance’ / ‘installment_payments’
- We will group them with our 'leading' dataset previous_application using SK_ID_PREV and our 'leading' dataset has this SK_ID_CURR key to be further mapped with our data

```
credit_card_balance.drop('SK_ID_CURR', axis=1, inplace=True)
installments_payments.drop('SK_ID_CURR', axis=1, inplace=True)
pos_cash_balance.drop('SK_ID_CURR', axis=1, inplace=True)
```

04 Merging | Previous_application

Problems

- Let's extract the number of previous applications of the clients to Home Credit and add this feature to our data

	SK_ID_CURR	PREVIOUS_APPLICATION_COUNT
0	100001	1
1	100002	1
2	100003	3
3	100004	1
4	100005	2

```
# and throw that column in our data
data = data.merge(previous_application_counts, on = 'SK_ID_CURR', how = 'left')
```

04 Merging | Previous_application

Step 1 - collapse credit_card_balance

```
def extract_mean(x):
    y = x.groupby('SK_ID_PREV', as_index=False).mean()
    return y

credit_card_balance_mean = extract_mean(credit_card_balance)
```

04 Merging | Previous_application

Step 2 - merge with previous_application

```
previous_application = previous_application.merge(credit_card_balance_mean,
                                                  on = 'SK_ID_PREV', how = 'left')
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCESS_START
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	17145.0	SATURDAY	15
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	607500.0	THURSDAY	11
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	112500.0	TUESDAY	11
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	450000.0	MONDAY	7
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	337500.0	THURSDAY	9
...	...	...	...	...	...	...	...	...	...
1670209	2300464	352015	Consumer loans	14704.290	267295.5	311400.0	267295.5	WEDNESDAY	12
1670210	2357031	334635	Consumer loans	6622.020	87750.0	64291.5	87750.0	TUESDAY	15
1670211	2659632	249544	Consumer loans	11520.855	105237.0	102523.5	105237.0	MONDAY	12
1670212	2785582	400317	Cash loans	18821.520	180000.0	191880.0	180000.0	WEDNESDAY	9
1670213	2418762	261212	Cash loans	16431.300	360000.0	360000.0	360000.0	SUNDAY	10

1670214 rows × 33 columns

04 Merging | Previous_application

Step 3 - collapse installments_payments

```
def extract_mean(x):
    y = x.groupby('SK_ID_PREV', as_index=False).mean()
    return y

install_pay_mean = extract_mean(installments_payments)
```

	SK_ID_PREV	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYS_INSTALMENT	AMT_INSTALMENT
0	1000001	1.500000	1.500000	-253.000000	34221.712500
1	1000002	1.250000	2.500000	-1555.000000	9308.891250
2	1000003	1.000000	2.000000	-64.000000	4951.350000
3	1000004	1.142857	4.000000	-772.000000	4789.022143
4	1000005	1.000000	5.818182	-1543.454545	14703.210000
...	...	...	...	...	...
997747	2843495	1.142857	4.000000	-349.000000	113932.883571
997748	2843496	0.000000	16.235294	-258.058824	9186.311912
997749	2843497	1.000000	10.500000	-303.000000	9175.185000
997750	2843498	1.333333	3.500000	-1367.666667	69053.572500
997751	2843499	1.100000	5.500000	-1068.000000	57808.224000

997752 rows × 5 columns

04 Merging | Previous_application

Step 4 - merge with previous application

```
previous_application = previous_application.merge(install_pay_mean,
                                                  on = 'SK_ID_PREV', how = 'left')
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCESS_START	F
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	17145.0	SATURDAY	15	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	607500.0	THURSDAY	11	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	112500.0	TUESDAY	11	
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	450000.0	MONDAY	7	
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	337500.0	THURSDAY	9	
...	...	...	...	...	...	...	...	...	...	
1670209	2300464	352015	Consumer loans	14704.290	267295.5	311400.0	267295.5	WEDNESDAY	12	
1670210	2357031	334635	Consumer loans	6622.020	87750.0	64291.5	87750.0	TUESDAY	15	
1670211	2659632	249544	Consumer loans	11520.855	105237.0	102523.5	105237.0	MONDAY	12	
1670212	2785582	400317	Cash loans	18821.520	180000.0	191880.0	180000.0	WEDNESDAY	9	
1670213	2418762	261212	Cash loans	16431.300	360000.0	360000.0	360000.0	SUNDAY	10	

1670214 rows × 37 columns

04 Merging | Previous_application

Step 5 - collapse POS_CASH_balance

```
def extract_mean(x):
    y = x.groupby('SK_ID_PREV', as_index=False).mean()
    return y

pos_mean = extract_mean(pos_cash_balance)
```

	SK_ID_PREV	MONTHS_BALANCE	CNT_INSTALMENT	CNT_INSTALMENT_FUTURE	SK_DPD	SK_DPD_DEF
0	1000001	-9.0	8.666667	7.666667	0.0	0.0
1	1000002	-52.0	5.200000	2.000000	0.0	0.0
2	1000003	-2.5	12.000000	10.500000	0.0	0.0
3	1000004	-25.5	9.625000	6.125000	0.0	0.0
4	1000005	-51.0	10.000000	5.000000	0.0	0.0
...	...	...	...	...	...	...
936320	2843494	-25.0	32.666667	31.666667	0.0	0.0
936321	2843495	-12.5	53.375000	49.875000	0.0	0.0
936322	2843497	-11.0	24.000000	14.000000	0.0	0.0
936323	2843498	-45.0	27.428571	24.285714	0.0	0.0
936324	2843499	-35.0	50.909091	45.636364	0.0	0.0

936325 rows × 6 columns

04 Merging | Previous_application

Step 6 - merge with previous_application

```
previous_application = previous_application.merge(pos_mean,
                                                  on = 'SK_ID_PREV', how = 'left')
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCESS_START
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	17145.0	SATURDAY	15
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	607500.0	THURSDAY	11
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	112500.0	TUESDAY	11
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	450000.0	MONDAY	7
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	337500.0	THURSDAY	9
...	...	...	...	...	...	...	...	...	...
1670209	2300464	352015	Consumer loans	14704.290	267295.5	311400.0	267295.5	WEDNESDAY	12
1670210	2357031	334635	Consumer loans	6622.020	87750.0	64291.5	87750.0	TUESDAY	15
1670211	2659632	249544	Consumer loans	11520.855	105237.0	102523.5	105237.0	MONDAY	12
1670212	2785582	400317	Cash loans	18821.520	180000.0	191880.0	180000.0	WEDNESDAY	9
1670213	2418762	261212	Cash loans	16431.300	360000.0	360000.0	360000.0	SUNDAY	10

1670214 rows × 42 columns

04 Merging | Previous_application

Step 7 - collapse the resulting previous_application dataset to show mean values grouped by SK_ID_CURR

```
def extract_mean(x):
    y = x.groupby('SK_ID_CURR', as_index=False).mean()
    return y

prev_appl_mean = extract_mean(previous_application)
prev_appl_mean = prev_appl_mean.drop('SK_ID_PREV', axis = 1)
```

	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_GOODS_PRICE	HOUR_APPR_PROCESS_START	NFLAG_LAST_APPL_IN_DAY	RATE_DOWN_PAYMENT	DAYS_DECISION	I
0	100001	3951.000000	24835.500	23787.00	24835.500	13.000000	1.0	0.104326	-1740.000	
1	100002	9251.775000	179055.000	179055.00	179055.000	9.000000	1.0	0.000000	-606.000	
2	100003	56553.990000	435436.500	484191.00	435436.500	14.666667	1.0	0.050030	-1305.000	
3	100004	5357.250000	24282.000	20106.00	24282.000	5.000000	1.0	0.212008	-815.000	
4	100005	4813.200000	22308.750	20076.75	44617.500	10.500000	1.0	0.108964	-536.000	
...	...	...	...	...	...	...	...	...	...	
338852	456251	6605.910000	40455.000	40455.00	40455.000	17.000000	1.0	0.000000	-273.000	
338853	456252	10074.465000	57595.500	56821.50	57595.500	10.000000	1.0	0.062443	-2497.000	
338854	456253	4770.405000	24162.750	20625.75	24162.750	11.500000	1.0	0.214316	-2380.000	
338855	456254	10681.132500	121317.750	134439.75	121317.750	15.000000	1.0	0.000000	-299.500	
338856	456255	20775.391875	362770.875	424431.00	362770.875	14.625000	1.0	0.064780	-587.625	

338857 rows × 26 columns

04 Merging | Previous_application

Step 8 - merge what we've got with our data

```
data = data.merge(prev_appl_mean, on = 'SK_ID_CURR', how = 'left')
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	AMT_CREDIT_x	AMT_ANNUITY_x	AMT_GOODS_PRICE_x	NAME_TYPE_SUITE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	...
0	100002.0	1.0	Cash loans	M	406597.5	24700.5	351000.0	Unaccompanied	Working	Secondary / secondary special	...
1	100003.0	0.0	Cash loans	F	1293502.5	35698.5	1129500.0	Family	State servant	Higher education	...
2	100004.0	0.0	Revolving loans	M	135000.0	6750.0	135000.0	Unaccompanied	Working	Secondary / secondary special	...
3	100006.0	0.0	Cash loans	F	312682.5	29686.5	297000.0	Unaccompanied	Working	Secondary / secondary special	...
4	100007.0	0.0	Cash loans	M	513000.0	21865.5	513000.0	Unaccompanied	Working	Secondary / secondary special	...
...	...	...	...	...	...	...	...	...	...	...	...
307506	456251.0	0.0	Cash loans	M	254700.0	27558.0	225000.0	Unaccompanied	Working	Secondary / secondary special	...
307507	456252.0	0.0	Cash loans	F	269550.0	12001.5	225000.0	Unaccompanied	Pensioner	Secondary / secondary special	...
307508	456253.0	0.0	Cash loans	F	677664.0	29979.0	585000.0	Unaccompanied	Working	Higher education	...
307509	456254.0	1.0	Cash loans	F	370107.0	20205.0	319500.0	Unaccompanied	Commercial associate	Secondary / secondary special	...
307510	456255.0	0.0	Cash loans	F	675000.0	49117.5	675000.0	Unaccompanied	Commercial associate	Higher education	...

307511 rows × 65 columns

04 Feature Selection

Steps

1. Drop columns having more than 65% null values
2. Drop 2 columns that have most of the values are 0
3. Drop columns with correlation greater than 0.95
4. Since we have analyzed the columns in application_train before, now we analyze the meaning of the other columns to TARGET column
5. Visualize the distributions of each feature of applicants who are capable of payments and who are not to find the important features

04 Feature Selection

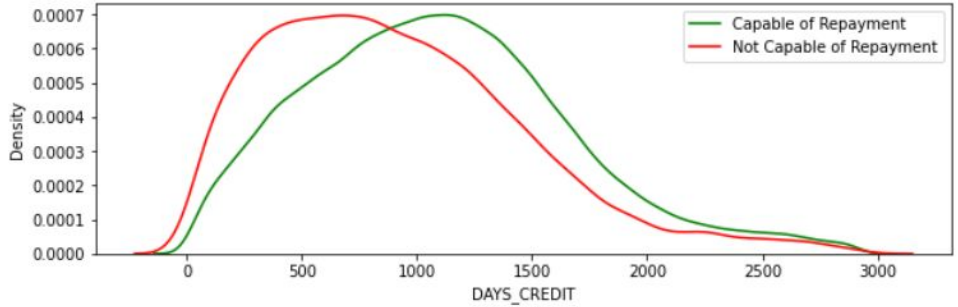
```
# drop columns having more than 65% null values
data.dropna(thresh=data.shape[0]*0.35, axis=1, inplace=True)
```

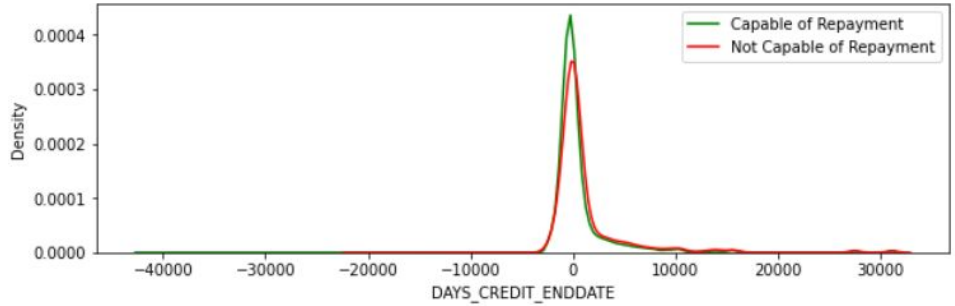
```
# drop 2 columns since most of values are 0
data.drop(['SK_DPD', 'SK_DPD_DEF'], axis=1, inplace=True)
```

```
# Create correlation matrix
corr_matrix = data.corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

# Find features with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]

# Drop features
data.drop(to_drop, axis=1, inplace=True)
```

04 Feature Selection

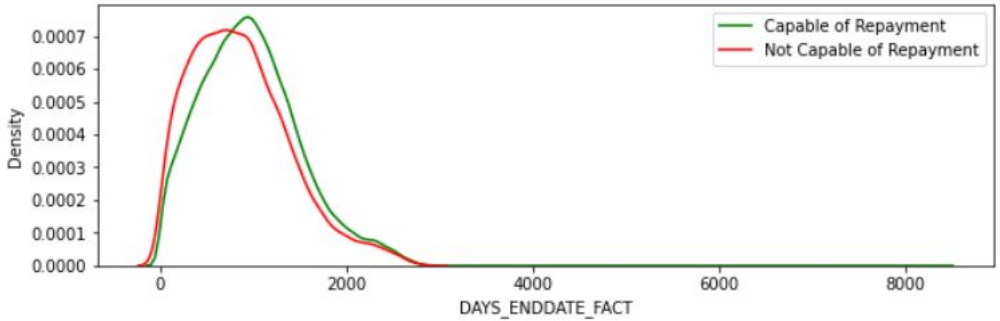
Visualize the distributions of each feature of applicants

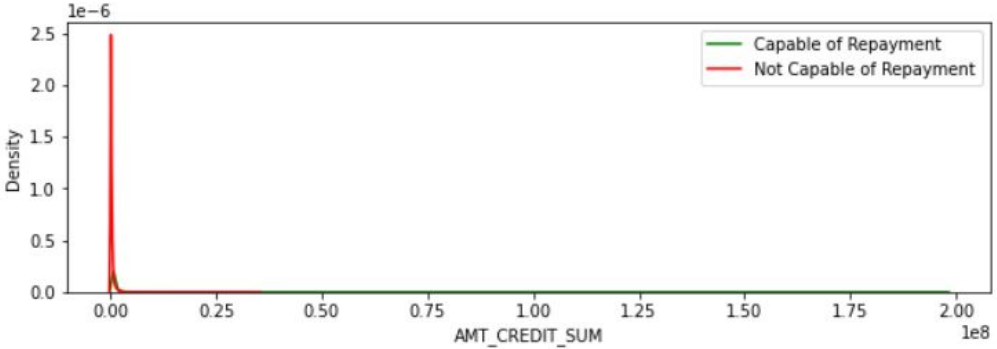




04 Feature Selection

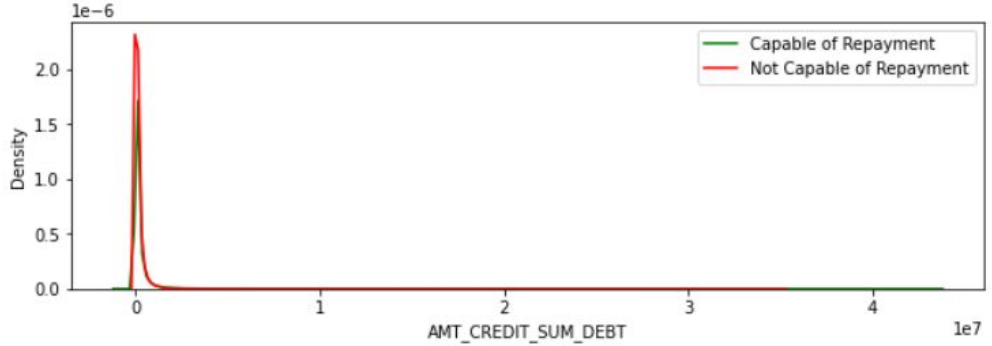
Visualize the distributions of each feature of applicants

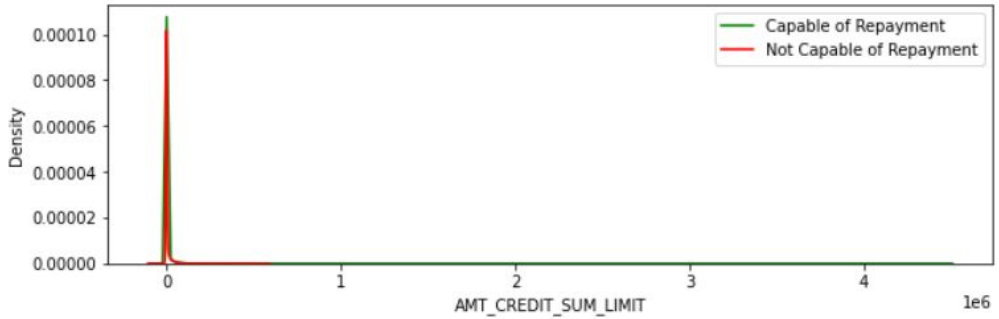




04 Feature Selection

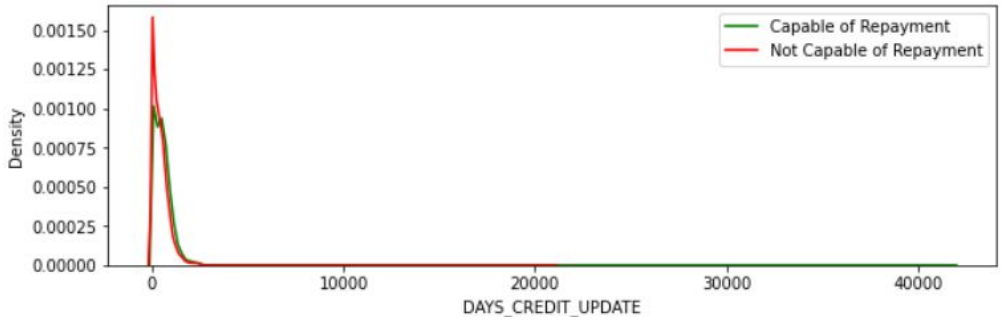
Visualize the distributions of each feature of applicants

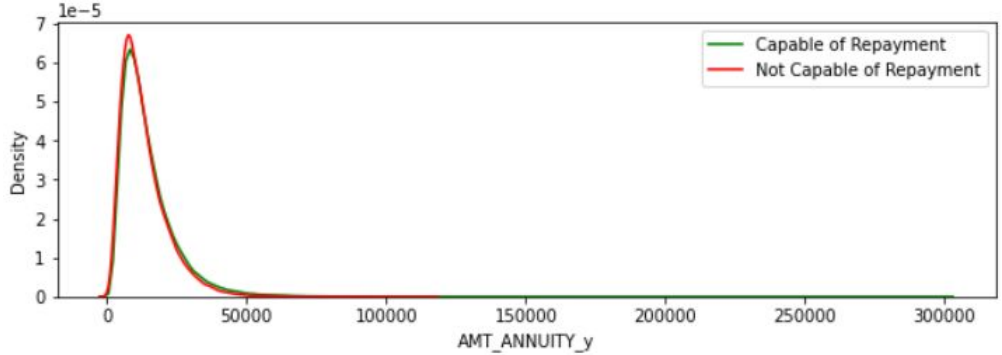




04 Feature Selection

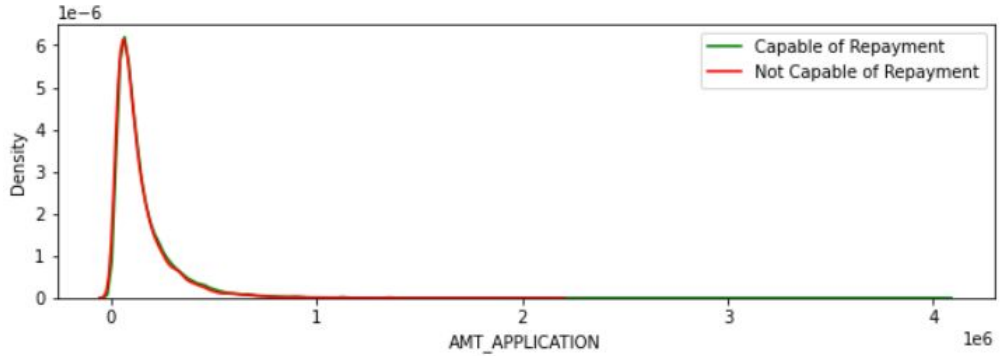
Visualize the distributions of each feature of applicants

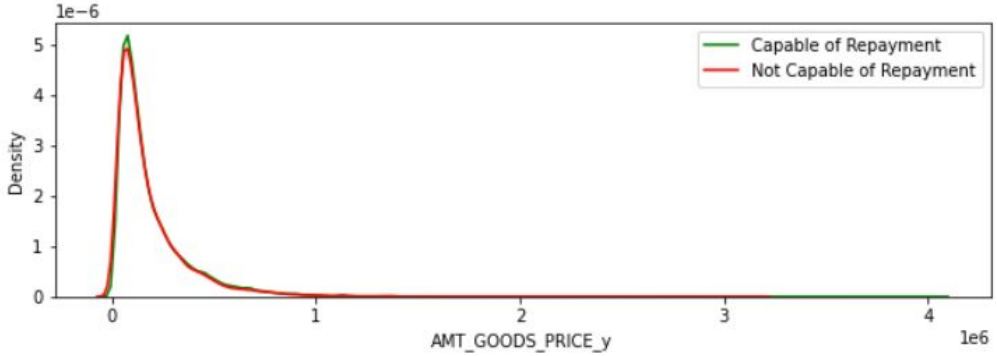




04 Feature Selection

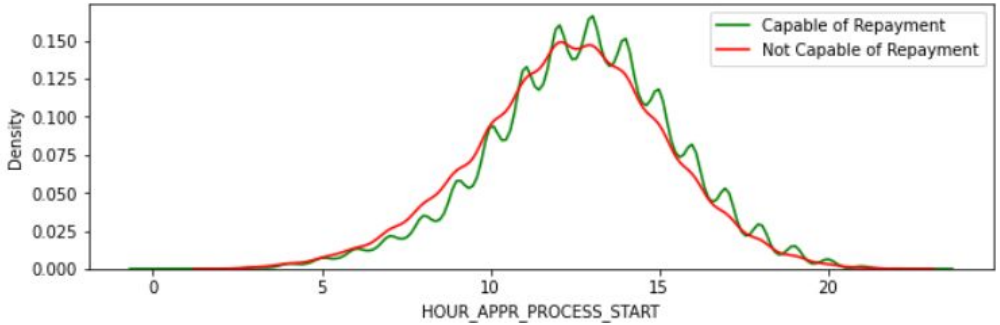
Visualize the distributions of each feature of applicants

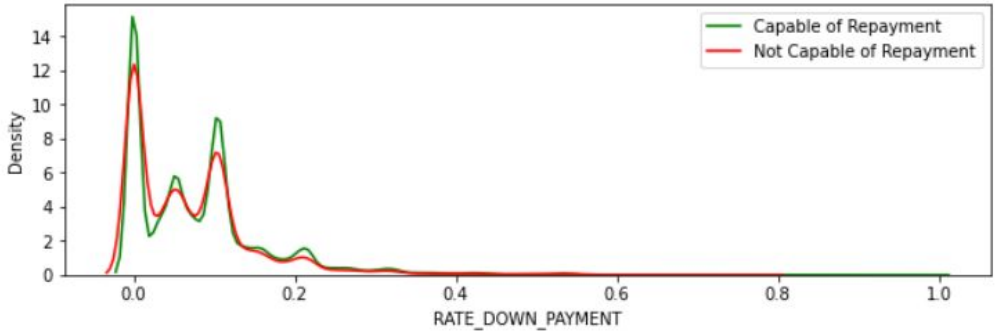




04 Feature Selection

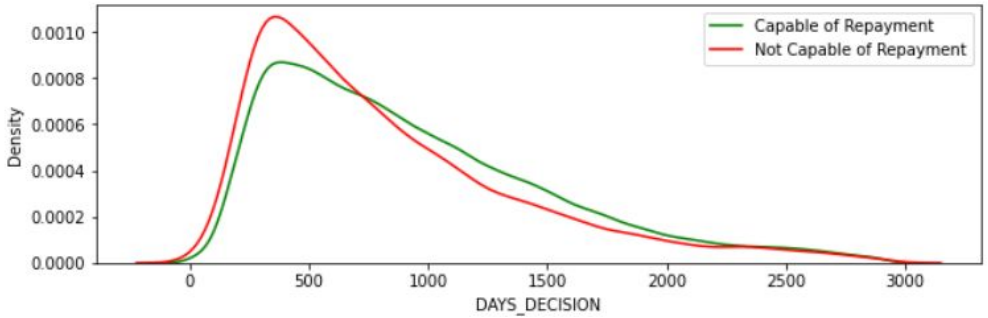
Visualize the distributions of each feature of applicants

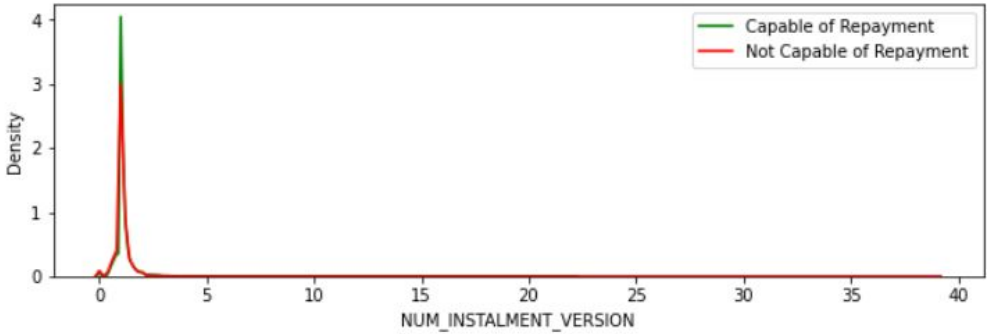




04 Feature Selection

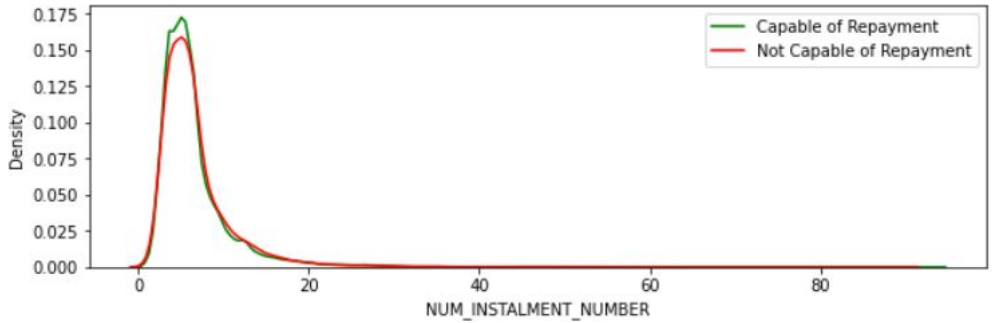
Visualize the distributions of each feature of applicants

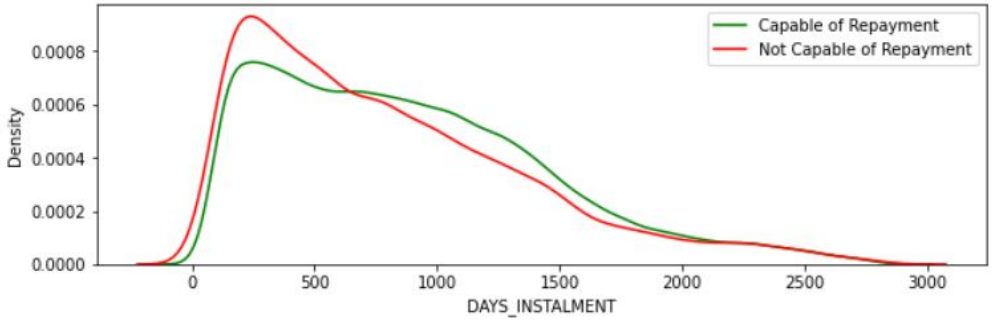




04 Feature Selection

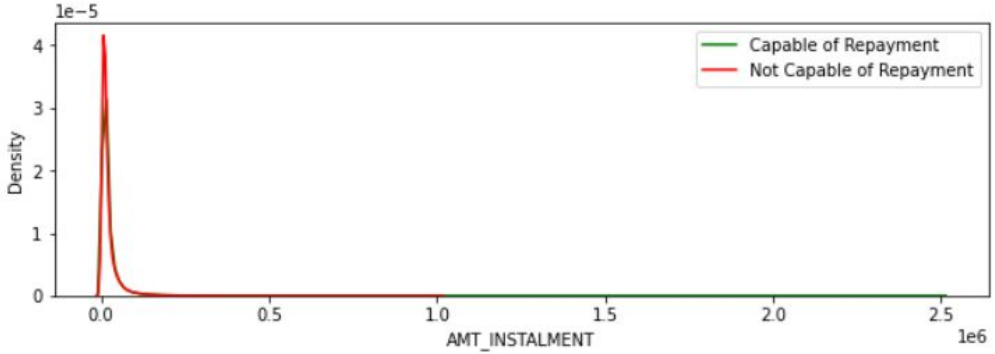
Visualize the distributions of each feature of applicants

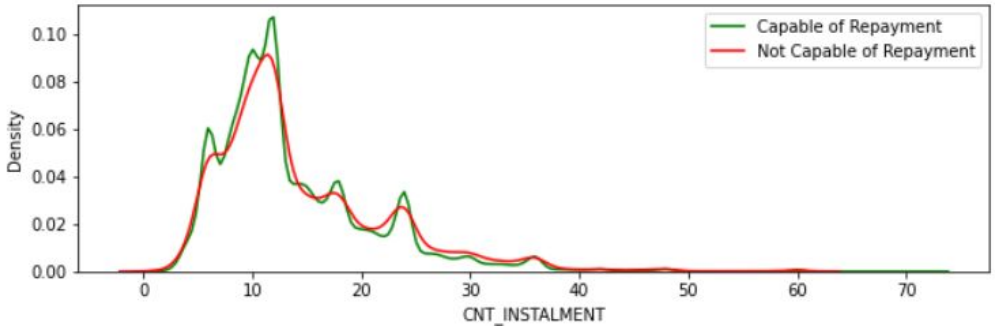




04 Feature Selection

Visualize the distributions of each feature of applicants





The End

Group 8

Thanks for following our project !