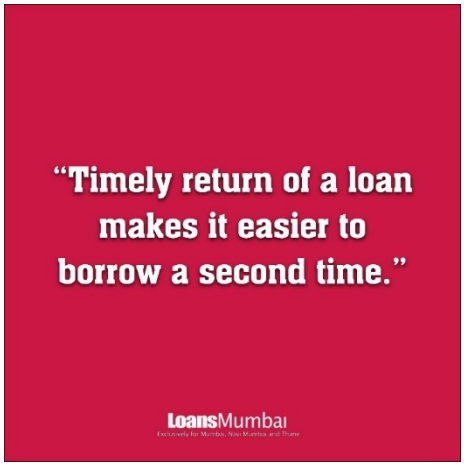


Loans have always been an important part of people's lives for quite some time now. Most of the developing and developed countries are using commodity of loan, it helps to stabilize and grow the economic section of the country as well as individuals. Each individual has different reasons for borrowing a loan. It could be to buy a dream car or a home, to set up a business, or to buy some products. Even wealthy people prefer taking loans overspending their cash so as to get tax benefits and to keep the cash available for future unexpected and unconventional expenses. Even countries take loan from world bank to fund its socio-economic reforms.

Loans are also as important to Lenders as they are for Borrowers. A loan is the major source of income for the banking sector as well as the biggest source of financial risk for banks. Large portions of a bank's assets directly come from the interests earned on loans given.

But here is the catch-

"The only good loan is one that gets paid back."

Its actually just not a quote but the crux of all financial institutions. Once banks make money out of a loan, they will be able to give loans back in market. Now crucial question is "How to pre-assume a loan seeker to be future defaulter.??". So, all around the world,**CREDIT BUREAUs keep track of these loans along with people and their personal details, their financial histories and credit and payback histories.** Some famous credit bureaus are CRIF-HIGHMARLK, CIBIL and EQUIFAX. All financial institutions share them their customer credit and payback histories in weekly or monthly fashion. Whenever a potential customer approach the bank, Bank ask for the all personal, financial and non-financial data from the customer and purpose of the loan. Along with that bank try to gather all other necessary details about customers like his income, assets and liabilities, credit bureau reports etc. So, bank thoroughly check each and every detail about customer before sanction a loan. It is used by a domain expert and it's a precise work because we don't want to lose a potential customer as well as we need to be much more confident as the loan shouldn't be a faulty loan at the end or the customer should be worthy of getting the loan and he can repay as well.

The Lending Organizations are faced with the tough task of analyzing the risk associated with each client. Therefore, it is important to identify the risky behaviors of clients and make educated decisions. The two most critical questions in the banking industry are

- (i) How risky is the borrower?
- (ii) Given the borrower's risk, should we lend him/her?

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders. Here we face issue of CREDIT RISK.

Default Credit risk is the chance that companies or individuals will be unable to make the required payments on their debt obligations. In other words, credit default risk is the probability that if you lend money, there is a chance that they won't be able to give the money back on time. Lenders and investors are exposed to default risk in virtually all forms of credit extensions. To mitigate the impact of default risk, lenders often impose charges that correspond to the debtor's level of default risk. A higher level of risk leads to a higher required return.

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful. So, Lending Organizations are approached usually by TWO types of customers:

-
    a. People who do not particularly have a prior credit history, for example students' loans, small businessmen, etc. who need credits, be it for studies, or for setting up some sort of businesses. Without adequate credit history, the lending organizations find it difficult to lend credits to such people, as these loans could be associated with high risks. In these kinds of situations, some lending organizations even tend to exploit the borrowers by asking for too high of an interest rate.
-
    a. People, who do have prior credit history, which could be with the same organization or some other organizations. However, going through that historical data could be very time consuming and redundant. This would scale up even further as the number of applicants increases. For such cases, if there could be a way through which the lending organization could predict or estimate the borrower's repayment capability, the process could be streamlined and be made effective for both the lender and the borrower. It could save resources both in terms of humans and time.

Home Credit is an international consumer finance provider that operates in 9 countries. It provides point of sales loans, cash loans, and revolving loans to underserved borrowers. They believe that credit history should not be a barrier for the borrowers to fulfil their dreams. Over 22 years of track record, they have accumulated a large number of

borrowers' behavioural data which they leverage to provide financial assistance to such customers. They have built predictive models that help them to efficiently analyse the risk associated with a given client and to also estimate the safe credit amount to be lent to customers, even with no credit history.

To solve this problem of **CREDIT RISK FOR PEOPLE WHO MAY HAVE OR NOT HAVING PRIOR CREDIT HISTORY**, we can take leverage of various data analysis tools like Statical analysis, graphs/charts and map it as a machine learning problem, So we can automate the process of loan approval with the lowering Credit default risk with help of different type of data available for a customer and we can create a benchmark to accept /reject a loan application. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

- Objective: The objective of this project is to use historical loan application data to predict whether or not an applicant will be able to repay a loan. This is a standard supervised classification model

- Supervised: The labels are included in the training data and the goal is to train a model to learn to predict the labels from the features

- Classification: The label is a binary variable, 0 (will repay loan on time), 1 (will have difficulty repaying loan)

  Here we are trying to create a Binary classifier which can predict with given feature(data) about a customer that whether a customers will face

  - Difficulty in payment: 1
  - No difficulty in payment: 0

- No strict low-latency requirement. It may take an hour /day to accept/reject a customer loan application.

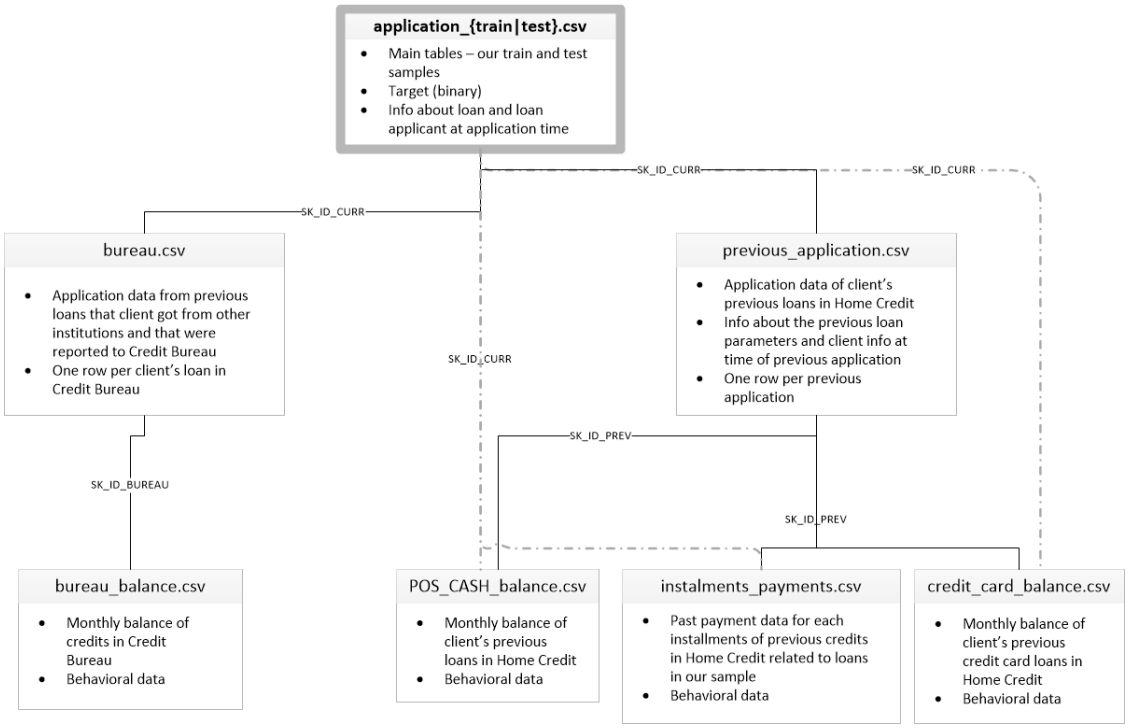
- Probability of prediction is useful to accept/reject a customer loan application.

- We need to be quite susceptible about False Positive (Offering loan to a potential defaulter). Because it be come with financial penalties so we will avoid in any case.

- Interpretability is somewhat important.

Data description and column analysis

The dataset is provided by **Home Credit Service** is an international non-bank financial institution founded in 1997 in the Czech Republic and headquartered in Netherlands. The company operates in 9 countries and focuses on installment lending primarily to people with little or no credit history. We have provided with data with the below Relational Diagram



Data Source : https://www.kaggle.com/c/home-credit-default-risk/data

application_{train|test}.csv:

- 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.

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 our sample, there are as many rows as number of credits the client had in 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 history of every previous credit reported to Credit Bureau – i.e the table has (#loans in sample * # of relative previous credits * # of months where we have some history observable for the previous credits) rows.

POS_CASH_balance.csv:

- Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit.

- This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous credits * # of months in which we have some history observable for the previous credits) rows.

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 history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous credit cards * # of months where we have some history observable for the previous credit card) rows.

previous_application.csv:

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

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 payment.
- One row is equivalent to one payment of one instalment OR one instalment corresponding to one payment of one previous Home Credit related to loans in our sample.

HomeCredit_columns_description.csv:

- This file contains descriptions for the columns in the various data files.
- Our interest of data is application_{train|test}.csv
- We can get column description of each column in application_train.csv from HomeCredit_columns_description.csv.

Let's see some important aspect of train data:

1. We have total 122 column, where TARGET is the label column and other 121 are feature columns.

2. Each feature consists of a specific financial or non-financial data about the customers.

3. We have 307511 customer's details in train data. There are

- 16 text feature which are just labels (categorical Column)

- Rest 105 are numerical features.

1. Each row has unique id ‘SK_ID_CURR’ and the output label is in the ‘TARGET’ column.
2. Our label column “Target” is a binary value (0,1).
3. This data is pretty imbalanced though with respect to “TARGET”. We need to be care about such kind of imbalanced data.
4. There are lots of missing values in each column.

We will dive deep in EDA part later and get more information about data.

```
import od

od.download("https://www.kaggle.com/c/home-credit-default-risk/data")
```

```
# Importing required libraries for overall analysis
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
pd.set_option('display.max_rows', None)
```

Exploratory Data Analysis (EDA)

```
### So we have total 6 files for use

import os

os.listdir("home-credit-default-risk/")
```

```
['application_test.csv',
 'application_train.csv',
 'bureau.csv',
 'bureau_balance.csv',
 'credit_card_balance.csv',
 'HomeCredit_columns_description.csv',
 'installments_payments.csv',
 'POS_CASH_balance.csv',
 'previous_application.csv',
 'sample_submission.csv']
```

```
# lets us read HomeCredit_columns_description.csv

col_descriptions=
pd.read_csv('HomeCredit_columns_description.csv',index_col='Unnamed:
0')
```

```
app_data_col_Details=col_descriptions[col_descriptions['Table']=='appl
ication_{train|test}.csv']

#So here we have details of all column in application.csv file
app_data_col_Details=app_data_col_Details[['Row','Description']]

## loading application_train.csv file to a dataframe
df_train=pd.read_csv(r'home-credit-default-risk/application_train.csv'
)

## loading application_train.csv file to a dataframe
df_test=pd.read_csv(r'home-credit-default-risk/application_test.csv')

## checking how the dataframe looks

df_train.head()
```

```
   SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
0      100002       1         Cash loans           M            N
1      100003       0         Cash loans           F            N
2      100004       0    Revolving loans           M            Y
3      100006       0         Cash loans           F            N
4      100007       0         Cash loans           M            N

  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT
AMT_ANNUITY  \
0               Y             0          202500.0    406597.5
24700.5
1               N             0          270000.0   1293502.5
35698.5
2               Y             0           67500.0    135000.0
6750.0
3               Y             0          135000.0    312682.5
29686.5
4               Y             0          121500.0    513000.0
21865.5

   ...  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20
FLAG_DOCUMENT_21  \
0  ...                 0                0                0
0
1  ...                 0                0                0
0
2  ...                 0                0                0
0
3  ...                 0                0                0
0
4  ...                 0                0                0
0

  AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
```

```
0                         0.0                         0.0
1                         0.0                         0.0
2                         0.0                         0.0
3                         NaN                         NaN
4                         0.0                         0.0

   AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                         0.0                        0.0
1                         0.0                        0.0
2                         0.0                        0.0
3                         NaN                        NaN
4                         0.0                        0.0

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                         1.0
1                        0.0                         0.0
2                        0.0                         0.0
3                        NaN                         NaN
4                        0.0                         0.0

[5 rows x 122 columns]
```

```
#dimension of data
df_train.shape
```

```
(307511, 122)
```

```
#Lets see how many columns are categorical column
len(df_train.select_dtypes(include=[object]).columns.to_list())
```

```
16
```

```
## getting the information about the dataframe created above
#df_train.info(verbose='all')
```

```
# checking the data types of all columns
df_train.dtypes.value_counts()
```

```
float64    65
int64      41
object     16
dtype: int64
```

So we have

- total 65 columns of type float64,
- total 41 columns of type int64(including TARGET column)
- total 16 columns of type object

```
## describing the dataframe
df_train.describe()
```

```
          SK_ID_CURR         TARGET   CNT_CHILDREN
AMT_INCOME_TOTAL  \
count  307511.000000  307511.000000  307511.000000      3.075110e+05

mean   278180.518577       0.080729       0.417052      1.687979e+05

std    102790.175348       0.272419       0.722121      2.371231e+05

min    100002.000000       0.000000       0.000000      2.565000e+04

25%    189145.500000       0.000000       0.000000      1.125000e+05

50%    278202.000000       0.000000       0.000000      1.471500e+05

75%    367142.500000       0.000000       1.000000      2.025000e+05

max    456255.000000       1.000000      19.000000      1.170000e+08


         AMT_CREDIT    AMT_ANNUITY  AMT_GOODS_PRICE  \
count  3.075110e+05  307499.000000     3.072330e+05
mean   5.990260e+05   27108.573909     5.383962e+05
std    4.024908e+05   14493.737315     3.694465e+05
min    4.500000e+04    1615.500000     4.050000e+04
25%    2.700000e+05   16524.000000     2.385000e+05
50%    5.135310e+05   24903.000000     4.500000e+05
75%    8.086500e+05   34596.000000     6.795000e+05
max    4.050000e+06  258025.500000     4.050000e+06

       REGION_POPULATION_RELATIVE     DAYS_BIRTH
DAYS_EMPLOYED  ...  \
count               307511.000000  307511.000000  307511.000000  ...

mean                     0.020868  -16036.995067   63815.045904  ...

std                      0.013831    4363.988632  141275.766519  ...

min                      0.000290  -25229.000000  -17912.000000  ...

25%                      0.010006  -19682.000000   -2760.000000  ...

50%                      0.018850  -15750.000000   -1213.000000  ...

75%                      0.028663  -12413.000000    -289.000000  ...

max                      0.072508   -7489.000000  365243.000000  ...
```

```
       FLAG_DOCUMENT_18  FLAG_DOCUMENT_19  FLAG_DOCUMENT_20
FLAG_DOCUMENT_21  \
count     307511.000000     307511.000000     307511.000000
307511.000000
mean           0.008130          0.000595          0.000507
0.000335
std            0.089798          0.024387          0.022518
0.018299
min            0.000000          0.000000          0.000000
0.000000
25%            0.000000          0.000000          0.000000
0.000000
50%            0.000000          0.000000          0.000000
0.000000
75%            0.000000          0.000000          0.000000
0.000000
max            1.000000          1.000000          1.000000
1.000000

       AMT_REQ_CREDIT_BUREAU_HOUR  AMT_REQ_CREDIT_BUREAU_DAY  \
count               265992.000000              265992.000000
mean                     0.006402                   0.007000
std                      0.083849                   0.110757
min                      0.000000                   0.000000
25%                      0.000000                   0.000000
50%                      0.000000                   0.000000
75%                      0.000000                   0.000000
max                      4.000000                   9.000000

       AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
count               265992.000000              265992.000000
mean                     0.034362                   0.267395
std                      0.204685                   0.916002
min                      0.000000                   0.000000
25%                      0.000000                   0.000000
50%                      0.000000                   0.000000
75%                      0.000000                   0.000000
max                      8.000000                  27.000000

       AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
count              265992.000000               265992.000000
mean                   0.265474                    1.899974
std                    0.794056                    1.869295
min                    0.000000                    0.000000
25%                    0.000000                    0.000000
50%                    0.000000                    1.000000
75%                    0.000000                    3.000000
max                  261.000000                   25.000000

[8 rows x 106 columns]
```

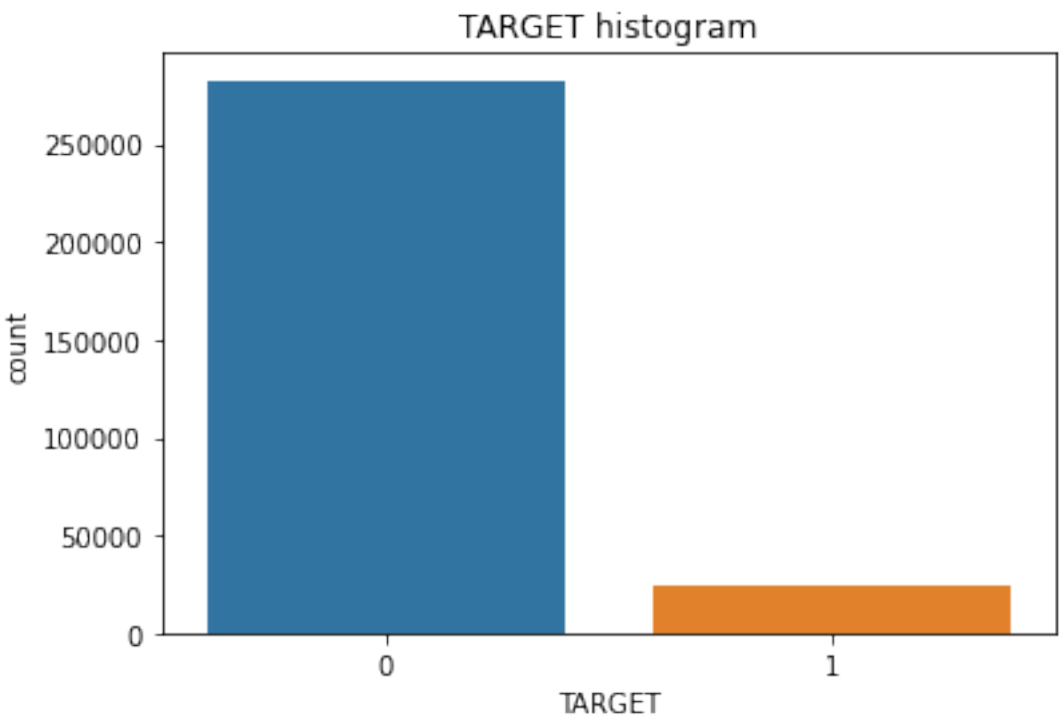
"TARGET" is the binary label column

```
#lets check if the dataset is balanced or not??

df_train.TARGET.value_counts()
```

```
0    282686
1     24825
Name: TARGET, dtype: int64
```

```
ax=sns.countplot(x=df_train.TARGET)
ax.set_title("TARGET histogram")
```

```
Text(0.5, 1.0, 'TARGET histogram')
```



So we can say our train dataet is higly imbalanced

lets check which are the column which are higly correlated with TARGET column

- There is a catch actually ,I think becaause Here 1 is difficulty in payment,So
- Higher the positive correlation means ,Higher chances for facing problem in repayment of loan in future
- Higher the negative correlation means ,Lower chances for facing problem in repayment of loan in future

```
df_train.corr('spearman')['TARGET'].sort_values(ascending=True)[::6]
```

```
EXT_SOURCE_3                  -0.166330
LIVINGAREA_AVG                -0.041631
ELEVATORS_MODE                -0.036845
```

```
AMT_GOODS_PRICE                -0.031493
FLAG_DOCUMENT_6                -0.028602
FLAG_PHONE                     -0.023806
BASEMENTAREA_MEDI              -0.022691
AMT_CREDIT                     -0.017537
NONLIVINGAREA_MODE             -0.012511
FLAG_DOCUMENT_8                -0.008040
FLAG_DOCUMENT_9                -0.004352
FLAG_EMAIL                     -0.001758
AMT_ANNUITY                    -0.000073
LIVE_REGION_NOT_WORK_REGION     0.002819
CNT_FAM_MEMBERS                 0.007006
FLAG_WORK_PHONE                 0.028524
REG_CITY_NOT_LIVE_CITY          0.044395
REGION_RATING_CLIENT            0.058909
Name: TARGET, dtype: float64
```

```
df_train.corr('pearson')['TARGET'].sort_values(ascending=True)[::6]
```

```
EXT_SOURCE_3                    -0.178919
FLOORSMAX_MODE                  -0.043226
FLOORSMIN_MEDI                  -0.033394
LIVINGAREA_MODE                 -0.030685
LIVINGAPARTMENTS_AVG            -0.025031
YEARS_BUILD_MEDI                -0.022326
ENTRANCES_MEDI                  -0.019025
NONLIVINGAREA_MEDI              -0.013337
LANDAREA_MEDI                   -0.011256
YEARS_BEGINEXPLUATATION_MODE    -0.009036
AMT_INCOME_TOTAL                -0.003982
AMT_REQ_CREDIT_BUREAU_QRT       -0.002022
FLAG_DOCUMENT_12                -0.000756
AMT_REQ_CREDIT_BUREAU_HOUR       0.000930
REG_REGION_NOT_WORK_REGION       0.006942
FLAG_WORK_PHONE                  0.028524
FLAG_DOCUMENT_3                  0.044346
REGION_RATING_CLIENT             0.058899
Name: TARGET, dtype: float64
```

```
df_train.corr('kendall')['TARGET'].sort_values(ascending=True)[::6]
```

```
EXT_SOURCE_3                   -0.135980
ELEVATORS_AVG                  -0.035479
LIVINGAREA_MODE                -0.031902
FLOORSMIN_MODE                 -0.027381
LIVINGAPARTMENTS_MODE          -0.023997
ENTRANCES_MEDI                 -0.020837
ENTRANCES_MODE                 -0.018855
AMT_CREDIT                     -0.014368
LANDAREA_AVG                   -0.011258
FLAG_DOCUMENT_8                -0.008040
```

```
FLAG_DOCUMENT_9               -0.004352
SK_ID_CURR                    -0.001724
AMT_ANNUITY                   -0.000060
LIVE_REGION_NOT_WORK_REGION    0.002819
REG_REGION_NOT_WORK_REGION     0.006942
FLAG_WORK_PHONE                0.028524
OWN_CAR_AGE                    0.044123
REGION_RATING_CLIENT           0.057178
Name: TARGET, dtype: float64
```

These correlation doesnt justify very well itself because if we see the column details and target,The correlation values seems to be insignificant

```
#get a correlation matrix

df_corr= df_train.corr()
```

Checking Null values

```
#Checking the number of columns with null values
#df_train.isnull().sum()
df_train[list(df_train.columns[df_train.isna().any()])].isnull().sum()
```

```
AMT_ANNUITY                        12
AMT_GOODS_PRICE                   278
NAME_TYPE_SUITE                  1292
OWN_CAR_AGE                    202929
OCCUPATION_TYPE                 96391
CNT_FAM_MEMBERS                     2
EXT_SOURCE_1                   173378
EXT_SOURCE_2                      660
EXT_SOURCE_3                    60965
APARTMENTS_AVG                 156061
BASEMENTAREA_AVG               179943
YEARS_BEGINEXPLUATATION_AVG    150007
YEARS_BUILD_AVG                204488
COMMONAREA_AVG                 214865
ELEVATORS_AVG                  163891
ENTRANCES_AVG                  154828
FLOORSMAX_AVG                  153020
FLOORSMIN_AVG                  208642
LANDAREA_AVG                   182590
LIVINGAPARTMENTS_AVG           210199
LIVINGAREA_AVG                 154350
NONLIVINGAPARTMENTS_AVG        213514
NONLIVINGAREA_AVG              169682
APARTMENTS_MODE                156061
BASEMENTAREA_MODE              179943
YEARS_BEGINEXPLUATATION_MODE   150007
YEARS_BUILD_MODE               204488
COMMONAREA_MODE                214865
ELEVATORS_MODE                 163891
```

```
ENTRANCES_MODE                  154828
FLOORSMAX_MODE                  153020
FLOORSMIN_MODE                  208642
LANDAREA_MODE                   182590
LIVINGAPARTMENTS_MODE           210199
LIVINGAREA_MODE                 154350
NONLIVINGAPARTMENTS_MODE        213514
NONLIVINGAREA_MODE              169682
APARTMENTS_MEDI                 156061
BASEMENTAREA_MEDI               179943
YEARS_BEGINEXPLUATATION_MEDI    150007
YEARS_BUILD_MEDI                204488
COMMONAREA_MEDI                 214865
ELEVATORS_MEDI                  163891
ENTRANCES_MEDI                  154828
FLOORSMAX_MEDI                  153020
FLOORSMIN_MEDI                  208642
LANDAREA_MEDI                   182590
LIVINGAPARTMENTS_MEDI           210199
LIVINGAREA_MEDI                 154350
NONLIVINGAPARTMENTS_MEDI        213514
NONLIVINGAREA_MEDI              169682
FONDKAPREMONT_MODE              210295
HOUSETYPE_MODE                  154297
TOTALAREA_MODE                  148431
WALLSMATERIAL_MODE              156341
EMERGENCYSTATE_MODE             145755
OBS_30_CNT_SOCIAL_CIRCLE          1021
DEF_30_CNT_SOCIAL_CIRCLE          1021
OBS_60_CNT_SOCIAL_CIRCLE          1021
DEF_60_CNT_SOCIAL_CIRCLE          1021
DAYS_LAST_PHONE_CHANGE               1
AMT_REQ_CREDIT_BUREAU_HOUR       41519
AMT_REQ_CREDIT_BUREAU_DAY        41519
AMT_REQ_CREDIT_BUREAU_WEEK       41519
AMT_REQ_CREDIT_BUREAU_MON        41519
AMT_REQ_CREDIT_BUREAU_QRT        41519
AMT_REQ_CREDIT_BUREAU_YEAR       41519
dtype: int64
```

```
#we  can see many of columns are having nulll ,lets filter those
columns

#df_train.columns[df_train.isna().any()].to_list()

#Total numbers of columns which have null values
len(df_train.columns[df_train.isna().any()].to_list())
```

```
67
```

```
#lets check if this is same for test data
len(df_test.columns[df_test.isna().any()].to_list())
```

```
64
```

```
#So lets check which columns are not in test data data as NULL

set(df_train.columns[df_train.isna().any()].to_list())-
set(df_test.columns[df_test.isna().any()].to_list())
```

```
{'AMT_GOODS_PRICE', 'CNT_FAM_MEMBERS', 'DAYS_LAST_PHONE_CHANGE'}
```

```
#So not a problem,let see The distribution of NULL values in train
data

percent_missing = df_train.isnull().sum() * 100 / len(df_train)

percent_missing[percent_missing>0]
```

```
AMT_ANNUITY                      0.003902
AMT_GOODS_PRICE                  0.090403
NAME_TYPE_SUITE                  0.420148
OWN_CAR_AGE                     65.990810
OCCUPATION_TYPE                 31.345545
CNT_FAM_MEMBERS                  0.000650
EXT_SOURCE_1                    56.381073
EXT_SOURCE_2                     0.214626
EXT_SOURCE_3                    19.825307
APARTMENTS_AVG                  50.749729
BASEMENTAREA_AVG                58.515956
YEARS_BEGINEXPLUATATION_AVG     48.781019
YEARS_BUILD_AVG                 66.497784
COMMONAREA_AVG                  69.872297
ELEVATORS_AVG                   53.295980
ENTRANCES_AVG                   50.348768
FLOORSMAX_AVG                   49.760822
FLOORSMIN_AVG                   67.848630
LANDAREA_AVG                    59.376738
LIVINGAPARTMENTS_AVG            68.354953
LIVINGAREA_AVG                  50.193326
NONLIVINGAPARTMENTS_AVG         69.432963
NONLIVINGAREA_AVG               55.179164
APARTMENTS_MODE                 50.749729
BASEMENTAREA_MODE               58.515956
YEARS_BEGINEXPLUATATION_MODE    48.781019
YEARS_BUILD_MODE                66.497784
COMMONAREA_MODE                 69.872297
ELEVATORS_MODE                  53.295980
ENTRANCES_MODE                  50.348768
FLOORSMAX_MODE                  49.760822
FLOORSMIN_MODE                  67.848630
```

```
LANDAREA_MODE                   59.376738
LIVINGAPARTMENTS_MODE           68.354953
LIVINGAREA_MODE                 50.193326
NONLIVINGAPARTMENTS_MODE        69.432963
NONLIVINGAREA_MODE              55.179164
APARTMENTS_MEDI                 50.749729
BASEMENTAREA_MEDI               58.515956
YEARS_BEGINEXPLUATATION_MEDI    48.781019
YEARS_BUILD_MEDI                66.497784
COMMONAREA_MEDI                 69.872297
ELEVATORS_MEDI                  53.295980
ENTRANCES_MEDI                  50.348768
FLOORSMAX_MEDI                  49.760822
FLOORSMIN_MEDI                  67.848630
LANDAREA_MEDI                   59.376738
LIVINGAPARTMENTS_MEDI           68.354953
LIVINGAREA_MEDI                 50.193326
NONLIVINGAPARTMENTS_MEDI        69.432963
NONLIVINGAREA_MEDI              55.179164
FONDKAPREMONT_MODE              68.386172
HOUSETYPE_MODE                  50.176091
TOTALAREA_MODE                  48.268517
WALLSMATERIAL_MODE              50.840783
EMERGENCYSTATE_MODE             47.398304
OBS_30_CNT_SOCIAL_CIRCLE         0.332021
DEF_30_CNT_SOCIAL_CIRCLE         0.332021
OBS_60_CNT_SOCIAL_CIRCLE         0.332021
DEF_60_CNT_SOCIAL_CIRCLE         0.332021
DAYS_LAST_PHONE_CHANGE           0.000325
AMT_REQ_CREDIT_BUREAU_HOUR      13.501631
AMT_REQ_CREDIT_BUREAU_DAY       13.501631
AMT_REQ_CREDIT_BUREAU_WEEK      13.501631
AMT_REQ_CREDIT_BUREAU_MON       13.501631
AMT_REQ_CREDIT_BUREAU_QRT       13.501631
AMT_REQ_CREDIT_BUREAU_YEAR      13.501631
dtype: float64
```

```
missing_value_df_train = pd.DataFrame({'column_name':
df_train.columns,
                                       'percent_missing':
percent_missing.values},index=None)
```

```
missing_value_df_train
```

```
                      column_name  percent_missing
0                      SK_ID_CURR         0.000000
1                          TARGET         0.000000
2              NAME_CONTRACT_TYPE         0.000000
3                     CODE_GENDER         0.000000
4                    FLAG_OWN_CAR         0.000000
5                 FLAG_OWN_REALTY         0.000000
```

6	CNT_CHILDREN	0.000000
7	AMT_INCOME_TOTAL	0.000000
8	AMT_CREDIT	0.000000
9	AMT_ANNUITY	0.003902
10	AMT_GOODS_PRICE	0.090403
11	NAME_TYPE_SUITE	0.420148
12	NAME_INCOME_TYPE	0.000000
13	NAME_EDUCATION_TYPE	0.000000
14	NAME_FAMILY_STATUS	0.000000
15	NAME_HOUSING_TYPE	0.000000
16	REGION_POPULATION_RELATIVE	0.000000
17	DAYS_BIRTH	0.000000
18	DAYS_EMPLOYED	0.000000
19	DAYS_REGISTRATION	0.000000
20	DAYS_ID_PUBLISH	0.000000
21	OWN_CAR_AGE	65.990810
22	FLAG_MOBIL	0.000000
23	FLAG_EMP_PHONE	0.000000
24	FLAG_WORK_PHONE	0.000000
25	FLAG_CONT_MOBILE	0.000000
26	FLAG_PHONE	0.000000
27	FLAG_EMAIL	0.000000
28	OCCUPATION_TYPE	31.345545
29	CNT_FAM_MEMBERS	0.000650
30	REGION_RATING_CLIENT	0.000000
31	REGION_RATING_CLIENT_W_CITY	0.000000
32	WEEKDAY_APPR_PROCESS_START	0.000000
33	HOUR_APPR_PROCESS_START	0.000000
34	REG_REGION_NOT_LIVE_REGION	0.000000
35	REG_REGION_NOT_WORK_REGION	0.000000
36	LIVE_REGION_NOT_WORK_REGION	0.000000
37	REG_CITY_NOT_LIVE_CITY	0.000000
38	REG_CITY_NOT_WORK_CITY	0.000000
39	LIVE_CITY_NOT_WORK_CITY	0.000000
40	ORGANIZATION_TYPE	0.000000
41	EXT_SOURCE_1	56.381073
42	EXT_SOURCE_2	0.214626
43	EXT_SOURCE_3	19.825307
44	APARTMENTS_AVG	50.749729
45	BASEMENTAREA_AVG	58.515956
46	YEARS_BEGINEXPLUATATION_AVG	48.781019
47	YEARS_BUILD_AVG	66.497784
48	COMMONAREA_AVG	69.872297
49	ELEVATORS_AVG	53.295980
50	ENTRANCES_AVG	50.348768
51	FLOORSMAX_AVG	49.760822
52	FLOORSMIN_AVG	67.848630
53	LANDAREA_AVG	59.376738
54	LIVINGAPARTMENTS_AVG	68.354953
55	LIVINGAREA_AVG	50.193326

56	NONLIVINGAPARTMENTS_AVG	69.432963
57	NONLIVINGAREA_AVG	55.179164
58	APARTMENTS_MODE	50.749729
59	BASEMENTAREA_MODE	58.515956
60	YEARS_BEGINEXPLUATATION_MODE	48.781019
61	YEARS_BUILD_MODE	66.497784
62	COMMONAREA_MODE	69.872297
63	ELEVATORS_MODE	53.295980
64	ENTRANCES_MODE	50.348768
65	FLOORSMAX_MODE	49.760822
66	FLOORSMIN_MODE	67.848630
67	LANDAREA_MODE	59.376738
68	LIVINGAPARTMENTS_MODE	68.354953
69	LIVINGAREA_MODE	50.193326
70	NONLIVINGAPARTMENTS_MODE	69.432963
71	NONLIVINGAREA_MODE	55.179164
72	APARTMENTS_MEDI	50.749729
73	BASEMENTAREA_MEDI	58.515956
74	YEARS_BEGINEXPLUATATION_MEDI	48.781019
75	YEARS_BUILD_MEDI	66.497784
76	COMMONAREA_MEDI	69.872297
77	ELEVATORS_MEDI	53.295980
78	ENTRANCES_MEDI	50.348768
79	FLOORSMAX_MEDI	49.760822
80	FLOORSMIN_MEDI	67.848630
81	LANDAREA_MEDI	59.376738
82	LIVINGAPARTMENTS_MEDI	68.354953
83	LIVINGAREA_MEDI	50.193326
84	NONLIVINGAPARTMENTS_MEDI	69.432963
85	NONLIVINGAREA_MEDI	55.179164
86	FONDKAPREMONT_MODE	68.386172
87	HOUSETYPE_MODE	50.176091
88	TOTALAREA_MODE	48.268517
89	WALLSMATERIAL_MODE	50.840783
90	EMERGENCYSTATE_MODE	47.398304
91	OBS_30_CNT_SOCIAL_CIRCLE	0.332021
92	DEF_30_CNT_SOCIAL_CIRCLE	0.332021
93	OBS_60_CNT_SOCIAL_CIRCLE	0.332021
94	DEF_60_CNT_SOCIAL_CIRCLE	0.332021
95	DAYS_LAST_PHONE_CHANGE	0.000325
96	FLAG_DOCUMENT_2	0.000000
97	FLAG_DOCUMENT_3	0.000000
98	FLAG_DOCUMENT_4	0.000000
99	FLAG_DOCUMENT_5	0.000000
100	FLAG_DOCUMENT_6	0.000000
101	FLAG_DOCUMENT_7	0.000000
102	FLAG_DOCUMENT_8	0.000000
103	FLAG_DOCUMENT_9	0.000000
104	FLAG_DOCUMENT_10	0.000000
105	FLAG_DOCUMENT_11	0.000000

```
106              FLAG_DOCUMENT_12         0.000000
107              FLAG_DOCUMENT_13         0.000000
108              FLAG_DOCUMENT_14         0.000000
109              FLAG_DOCUMENT_15         0.000000
110              FLAG_DOCUMENT_16         0.000000
111              FLAG_DOCUMENT_17         0.000000
112              FLAG_DOCUMENT_18         0.000000
113              FLAG_DOCUMENT_19         0.000000
114              FLAG_DOCUMENT_20         0.000000
115              FLAG_DOCUMENT_21         0.000000
116    AMT_REQ_CREDIT_BUREAU_HOUR        13.501631
117     AMT_REQ_CREDIT_BUREAU_DAY        13.501631
118    AMT_REQ_CREDIT_BUREAU_WEEK        13.501631
119     AMT_REQ_CREDIT_BUREAU_MON        13.501631
120     AMT_REQ_CREDIT_BUREAU_QRT        13.501631
121    AMT_REQ_CREDIT_BUREAU_YEAR        13.501631
```

```
missing_value_df_train.sort_values('percent_missing',ascending=False,i
nplace=True)
```

```
missing_value_df_train=missing_value_df_train[missing_value_df_train['
percent_missing']>0]
```

```
missing_value_df_train.reset_index(inplace=True,drop=True)
```

```
missing_value_df_train
```

```
                     column_name  percent_missing
0                COMMONAREA_MEDI        69.872297
1                 COMMONAREA_AVG        69.872297
2                COMMONAREA_MODE        69.872297
3       NONLIVINGAPARTMENTS_MODE        69.432963
4        NONLIVINGAPARTMENTS_AVG        69.432963
5       NONLIVINGAPARTMENTS_MEDI        69.432963
6             FONDKAPREMONT_MODE        68.386172
7          LIVINGAPARTMENTS_MODE        68.354953
8           LIVINGAPARTMENTS_AVG        68.354953
9          LIVINGAPARTMENTS_MEDI        68.354953
10                 FLOORSMIN_AVG        67.848630
11                FLOORSMIN_MODE        67.848630
12                FLOORSMIN_MEDI        67.848630
13              YEARS_BUILD_MEDI        66.497784
14              YEARS_BUILD_MODE        66.497784
15               YEARS_BUILD_AVG        66.497784
16                   OWN_CAR_AGE        65.990810
17                 LANDAREA_MEDI        59.376738
18                 LANDAREA_MODE        59.376738
19                  LANDAREA_AVG        59.376738
20             BASEMENTAREA_MEDI        58.515956
21              BASEMENTAREA_AVG        58.515956
22             BASEMENTAREA_MODE        58.515956
```

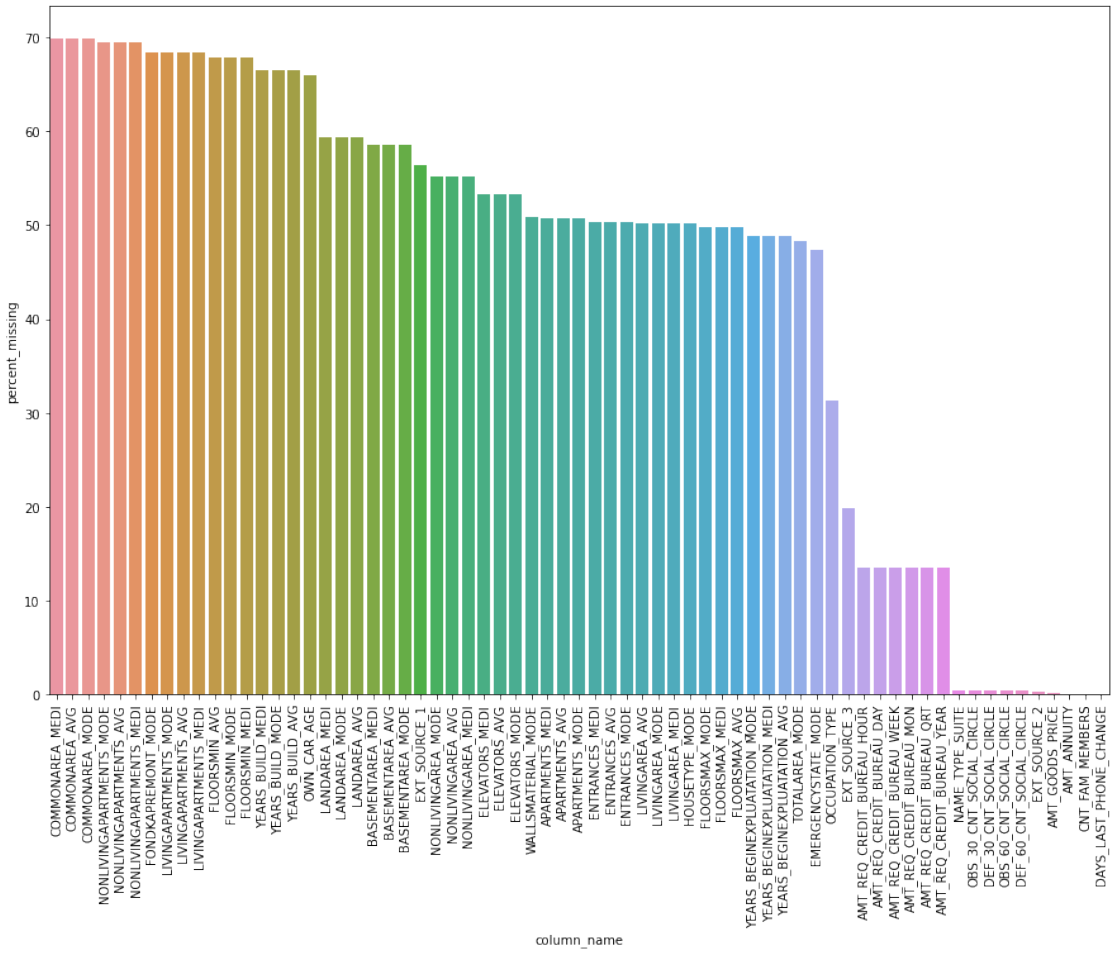
```
23                  EXT_SOURCE_1        56.381073
24            NONLIVINGAREA_MODE        55.179164
25             NONLIVINGAREA_AVG        55.179164
26            NONLIVINGAREA_MEDI        55.179164
27                ELEVATORS_MEDI        53.295980
28                 ELEVATORS_AVG        53.295980
29                ELEVATORS_MODE        53.295980
30            WALLSMATERIAL_MODE        50.840783
31               APARTMENTS_MEDI        50.749729
32                APARTMENTS_AVG        50.749729
33               APARTMENTS_MODE        50.749729
34                ENTRANCES_MEDI        50.348768
35                 ENTRANCES_AVG        50.348768
36                ENTRANCES_MODE        50.348768
37                LIVINGAREA_AVG        50.193326
38               LIVINGAREA_MODE        50.193326
39               LIVINGAREA_MEDI        50.193326
40                HOUSETYPE_MODE        50.176091
41                FLOORSMAX_MODE        49.760822
42                FLOORSMAX_MEDI        49.760822
43                 FLOORSMAX_AVG        49.760822
44  YEARS_BEGINEXPLUATATION_MODE        48.781019
45  YEARS_BEGINEXPLUATATION_MEDI        48.781019
46   YEARS_BEGINEXPLUATATION_AVG        48.781019
47                TOTALAREA_MODE        48.268517
48           EMERGENCYSTATE_MODE        47.398304
49               OCCUPATION_TYPE        31.345545
50                  EXT_SOURCE_3        19.825307
51    AMT_REQ_CREDIT_BUREAU_HOUR        13.501631
52     AMT_REQ_CREDIT_BUREAU_DAY        13.501631
53    AMT_REQ_CREDIT_BUREAU_WEEK        13.501631
54     AMT_REQ_CREDIT_BUREAU_MON        13.501631
55     AMT_REQ_CREDIT_BUREAU_QRT        13.501631
56    AMT_REQ_CREDIT_BUREAU_YEAR        13.501631
57               NAME_TYPE_SUITE         0.420148
58      OBS_30_CNT_SOCIAL_CIRCLE         0.332021
59      DEF_30_CNT_SOCIAL_CIRCLE         0.332021
60      OBS_60_CNT_SOCIAL_CIRCLE         0.332021
61      DEF_60_CNT_SOCIAL_CIRCLE         0.332021
62                  EXT_SOURCE_2         0.214626
63               AMT_GOODS_PRICE         0.090403
64                   AMT_ANNUITY         0.003902
65               CNT_FAM_MEMBERS         0.000650
66        DAYS_LAST_PHONE_CHANGE         0.000325
```

```
#Lets plot it
plt.figure(figsize=(15,10))
g=sns.barplot(x='column_name',y='percent_missing',data=missing_value_d
f_train)
```

```
labels=missing_value_df_train.column_name.to_list()
g.set_xticklabels(labels,rotation=90)
```

```
[Text(0, 0, 'COMMONAREA_MEDI'),
 Text(1, 0, 'COMMONAREA_AVG'),
 Text(2, 0, 'COMMONAREA_MODE'),
 Text(3, 0, 'NONLIVINGAPARTMENTS_MODE'),
 Text(4, 0, 'NONLIVINGAPARTMENTS_AVG'),
 Text(5, 0, 'NONLIVINGAPARTMENTS_MEDI'),
 Text(6, 0, 'FONDKAPREMONT_MODE'),
 Text(7, 0, 'LIVINGAPARTMENTS_MODE'),
 Text(8, 0, 'LIVINGAPARTMENTS_AVG'),
 Text(9, 0, 'LIVINGAPARTMENTS_MEDI'),
 Text(10, 0, 'FLOORSMIN_AVG'),
 Text(11, 0, 'FLOORSMIN_MODE'),
 Text(12, 0, 'FLOORSMIN_MEDI'),
 Text(13, 0, 'YEARS_BUILD_MEDI'),
 Text(14, 0, 'YEARS_BUILD_MODE'),
 Text(15, 0, 'YEARS_BUILD_AVG'),
 Text(16, 0, 'OWN_CAR_AGE'),
 Text(17, 0, 'LANDAREA_MEDI'),
 Text(18, 0, 'LANDAREA_MODE'),
 Text(19, 0, 'LANDAREA_AVG'),
 Text(20, 0, 'BASEMENTAREA_MEDI'),
 Text(21, 0, 'BASEMENTAREA_AVG'),
 Text(22, 0, 'BASEMENTAREA_MODE'),
 Text(23, 0, 'EXT_SOURCE_1'),
 Text(24, 0, 'NONLIVINGAREA_MODE'),
 Text(25, 0, 'NONLIVINGAREA_AVG'),
 Text(26, 0, 'NONLIVINGAREA_MEDI'),
 Text(27, 0, 'ELEVATORS_MEDI'),
 Text(28, 0, 'ELEVATORS_AVG'),
 Text(29, 0, 'ELEVATORS_MODE'),
 Text(30, 0, 'WALLSMATERIAL_MODE'),
 Text(31, 0, 'APARTMENTS_MEDI'),
 Text(32, 0, 'APARTMENTS_AVG'),
 Text(33, 0, 'APARTMENTS_MODE'),
 Text(34, 0, 'ENTRANCES_MEDI'),
 Text(35, 0, 'ENTRANCES_AVG'),
 Text(36, 0, 'ENTRANCES_MODE'),
 Text(37, 0, 'LIVINGAREA_AVG'),
 Text(38, 0, 'LIVINGAREA_MODE'),
 Text(39, 0, 'LIVINGAREA_MEDI'),
 Text(40, 0, 'HOUSETYPE_MODE'),
 Text(41, 0, 'FLOORSMAX_MODE'),
 Text(42, 0, 'FLOORSMAX_MEDI'),
 Text(43, 0, 'FLOORSMAX_AVG'),
 Text(44, 0, 'YEARS_BEGINEXPLUATATION_MODE'),
 Text(45, 0, 'YEARS_BEGINEXPLUATATION_MEDI'),
 Text(46, 0, 'YEARS_BEGINEXPLUATATION_AVG'),
```

```
Text(47, 0, 'TOTALAREA_MODE'),
Text(48, 0, 'EMERGENCYSTATE_MODE'),
Text(49, 0, 'OCCUPATION_TYPE'),
Text(50, 0, 'EXT_SOURCE_3'),
Text(51, 0, 'AMT_REQ_CREDIT_BUREAU_HOUR'),
Text(52, 0, 'AMT_REQ_CREDIT_BUREAU_DAY'),
Text(53, 0, 'AMT_REQ_CREDIT_BUREAU_WEEK'),
Text(54, 0, 'AMT_REQ_CREDIT_BUREAU_MON'),
Text(55, 0, 'AMT_REQ_CREDIT_BUREAU_QRT'),
Text(56, 0, 'AMT_REQ_CREDIT_BUREAU_YEAR'),
Text(57, 0, 'NAME_TYPE_SUITE'),
Text(58, 0, 'OBS_30_CNT_SOCIAL_CIRCLE'),
Text(59, 0, 'DEF_30_CNT_SOCIAL_CIRCLE'),
Text(60, 0, 'OBS_60_CNT_SOCIAL_CIRCLE'),
Text(61, 0, 'DEF_60_CNT_SOCIAL_CIRCLE'),
Text(62, 0, 'EXT_SOURCE_2'),
Text(63, 0, 'AMT_GOODS_PRICE'),
Text(64, 0, 'AMT_ANNUITY'),
Text(65, 0, 'CNT_FAM_MEMBERS'),
Text(66, 0, 'DAYS_LAST_PHONE_CHANGE')]
```



- We can see top NaN values are belong the columns which are refering to the Accmodtaion building
- Next comes the Occupation_type,3rdExt_IncomeSource, Credit Bureau details ,observation of client's social surroundings defaulted in day past dues
- Rest are very very less NAN value
- These details being blank is normal because these are the extra details which banks collect if these details actually exist for the customer

```
# we will handle it step by step
```

Lets see categorical variables first

```
categorical_features=df_train.select_dtypes(include=[object]).columns.
to_list()
categorical_features
```

```
['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']
```

```
#So we have 16 categorical features ,we need to Numerically encode
it,Lets go column by column

#To see The defaulters statistics we will create a sepearte defaulters
dataframe

df_defaulter = df_train[df_train['TARGET']==1]

#To see The non defaulters statistics we will create a sepearte
defaulters dataframe

df_non_defaulter = df_train[df_train['TARGET']==0]
```

NAME_CONTRACT_TYPE

Identification if loan is cash or revolving

```
df_train['NAME_CONTRACT_TYPE'].value_counts()
```

```
Cash loans         278232
Revolving loans     29279
Name: NAME_CONTRACT_TYPE, dtype: int64
```

```
df_test['NAME_CONTRACT_TYPE'].value_counts()
```

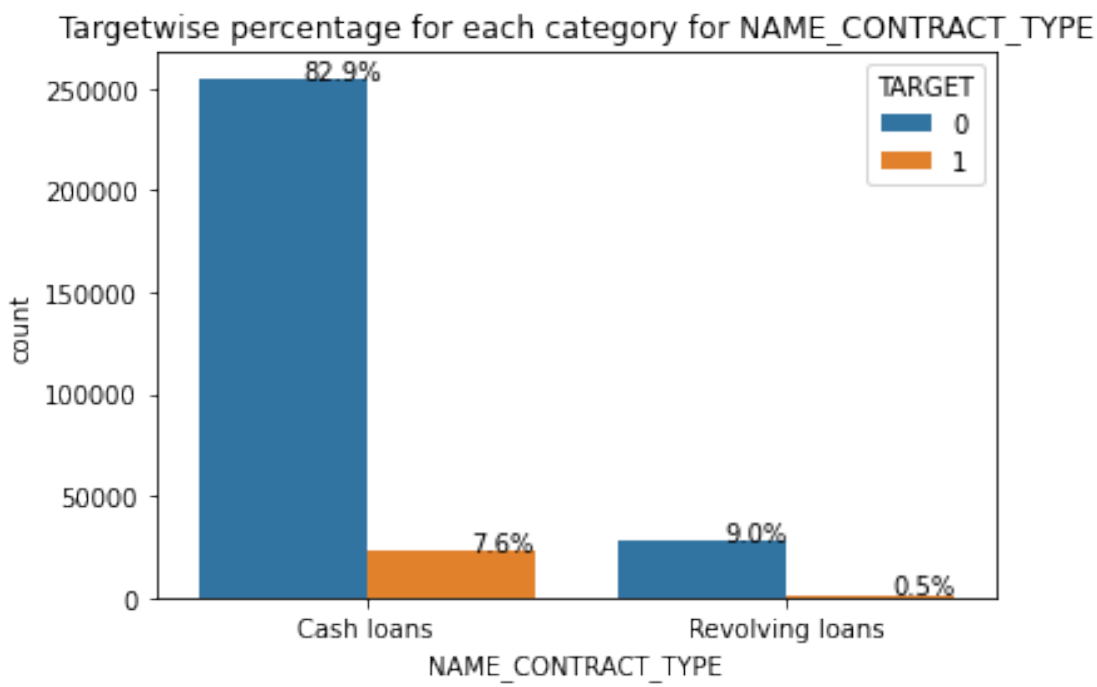
```
Cash loans         48305
Revolving loans      439
Name: NAME_CONTRACT_TYPE, dtype: int64
```

```
#Any null values??
df_train['NAME_CONTRACT_TYPE'].isnull().sum()
```

```
0
```

```
#lets see its distribution wrt TARGET
ax=sns.countplot(x='NAME_CONTRACT_TYPE',hue='TARGET',data=df_train)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))


ax.set_title("Targetwise percentage for each category for
NAME_CONTRACT_TYPE")
```

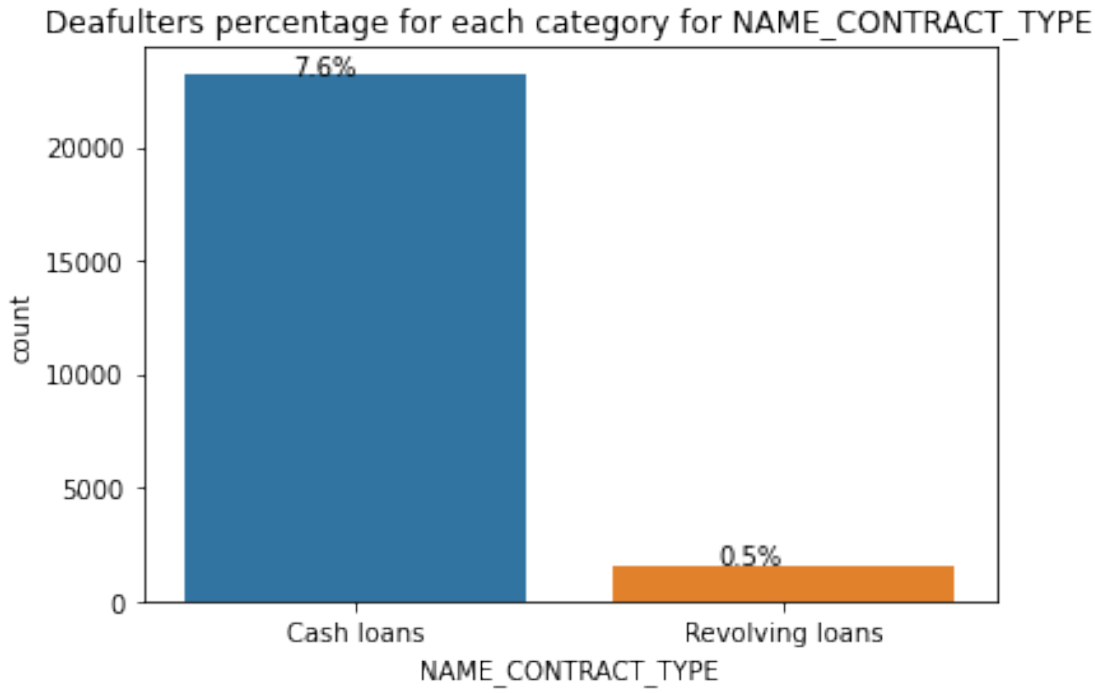
```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NAME_CONTRACT_TYPE')
```




```
#lets see defaulters distribution
ax=sns.countplot(x='NAME_CONTRACT_TYPE',data=df_defaulter)
```

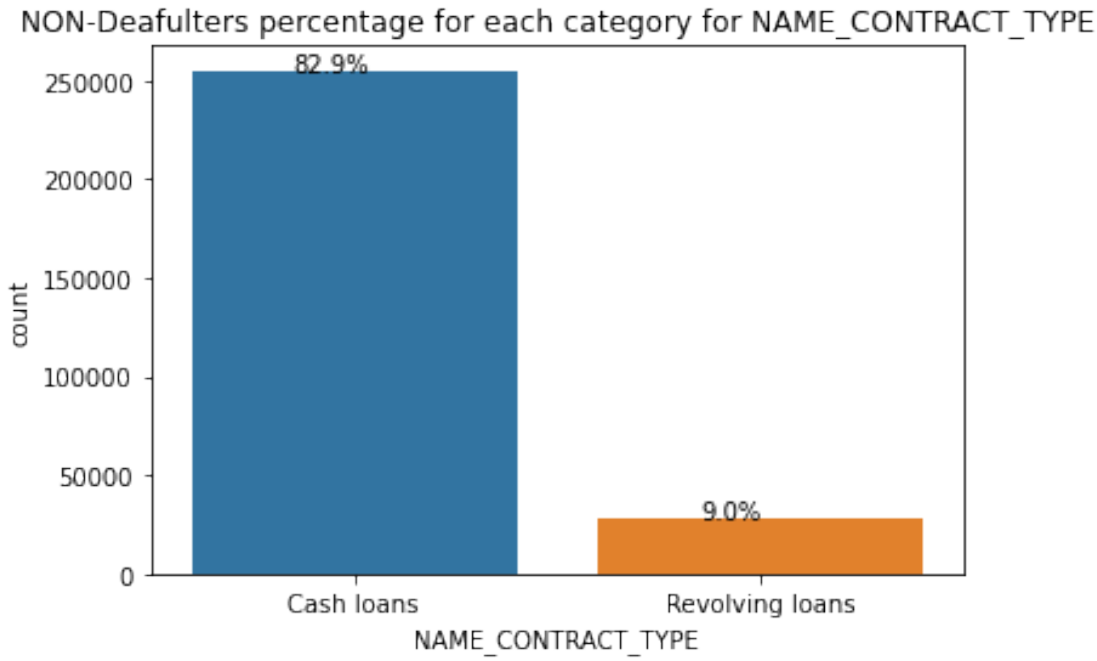
```
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Deafulters percentage for each category for
NAME_CONTRACT_TYPE")
```

```
Text(0.5, 1.0, 'Deafulters percentage for each category for
NAME_CONTRACT_TYPE')
```




```
#so we can see cash loans are seems to be more having defaulter

#lets see defaulters distribution wrt TARGET
ax=sns.countplot(x='NAME_CONTRACT_TYPE',data=df_non_defaulter)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("NON-Deafulters percentage for each category for
NAME_CONTRACT_TYPE")
```

```
Text(0.5, 1.0, 'NON-Deafulters percentage for each category for
NAME_CONTRACT_TYPE')
```

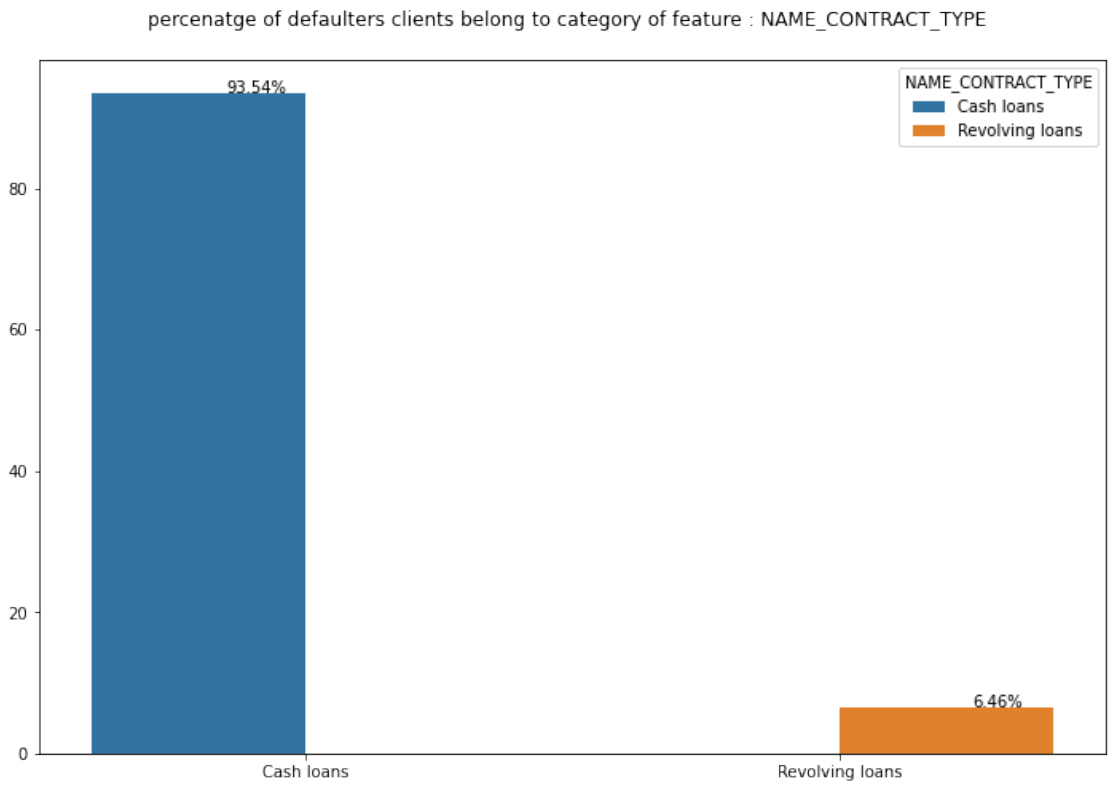



```
# we can also among Non default loans also cash loan seems to be more
easily paid

#lets see one more way,we need to find percentrage of deafaulters in
each category

df_train['NAME_CONTRACT_TYPE'].value_counts().index.to_list()
```

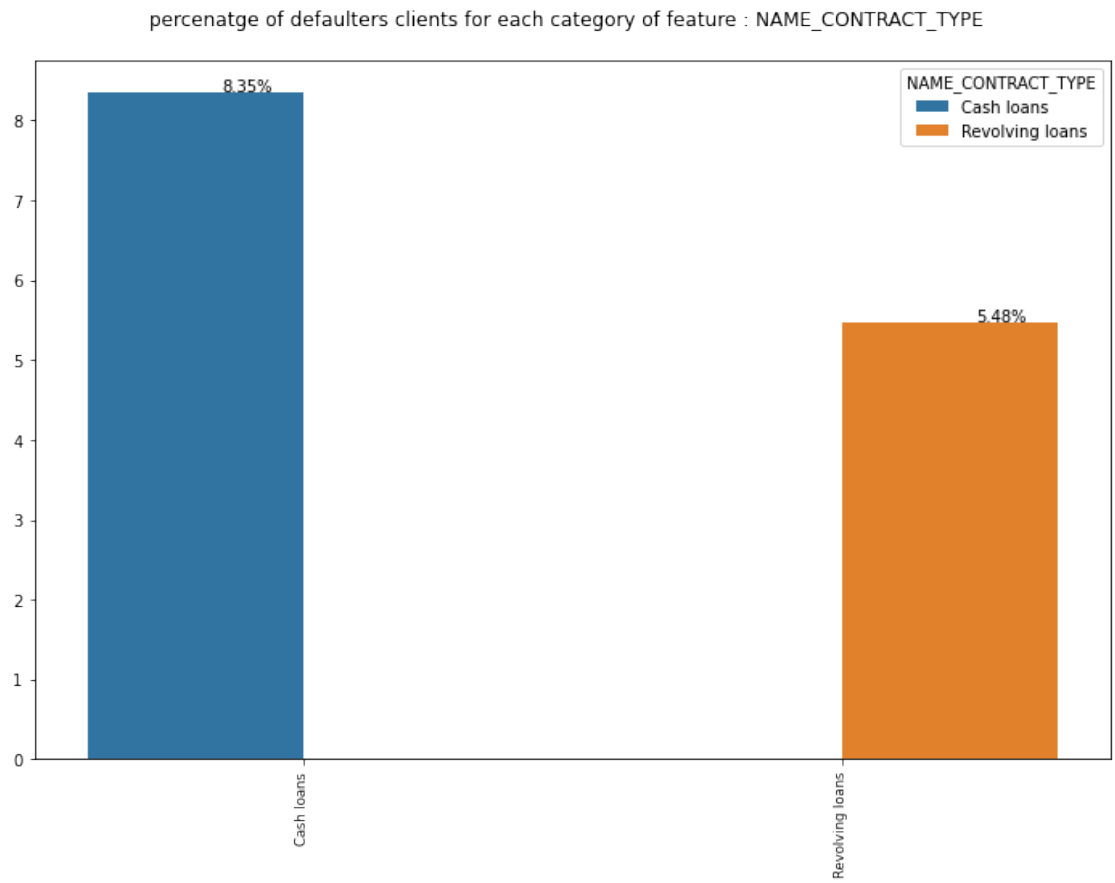
```
['Cash loans', 'Revolving loans']
```

```
def plot_for_defaulters_categorywise(df,feature):
    """
    Here plot will show in any category of feature what are % of
clients which are deafulters
    """
    categories= df[feature].value_counts().index.to_list()
    category_defaulter_dict=dict()
    for i in categories:
        count_defaulter=df_train[(df[feature]==i) &
(df['TARGET']==1)].shape[0]
        count_total=df_train[(df[feature]==i)].shape[0]
        #count_total=df_train[df_train[]].shape[0]
        percetage_defaulter= float(count_defaulter*100/count_total)
        category_defaulter_dict[i]=percetage_defaulter
    #return category_defaulter_dict
    keys = list(category_defaulter_dict.keys())
    # get values in the same order as keys, and parse percentage
values
    vals = [float(category_defaulter_dict[k]) for k in keys]
    plt.figure(figsize=(12,8))
```

```
    ax=sns.barplot(x=keys, y=vals,hue=keys)
    ax.legend(title=feature,loc='best')
    plt.xticks(rotation=90,fontsize='small')
    for p in ax.patches:
        ax.annotate('{:.2f}%'.format(p.get_height()), (p.get_x()+0.25,
p.get_height()+0.01))
    plt.title(f"percenatge of defaulters clients for each category of
feature : {feature}  \n ")

def plot_for_categorical_defaulters(df,feature):
    """
    Here plot will show out of all defaulters what are percentage of
defaulter who belong to any feature's category

    """
    categories= df[feature].value_counts().index.to_list()
    category_defaulter_dict=dict()
    for i in categories:
        count_defaulter=df_train[(df[feature]==i) &
(df['TARGET']==1)].shape[0]
        count_total=df_train[(df['TARGET']==1)].shape[0]
        #count_total=df_train[df_train[]].shape[0]
        percetage_defaulter= float(count_defaulter*100/count_total)
        category_defaulter_dict[i]=percetage_defaulter
    #return category_defaulter_dict
    keys = list(category_defaulter_dict.keys())
    # get values in the same order as keys, and parse percentage
values
    vals = [float(category_defaulter_dict[k]) for k in keys]
    plt.figure(figsize=(12,8))
    ax=sns.barplot(x=keys, y=vals,hue=keys)
    ax.legend(title=feature,loc='best')
    for p in ax.patches:
        ax.annotate('{:.2f}%'.format(p.get_height()), (p.get_x()+0.25,
p.get_height()+0.01))
    plt.title(f"percenatge of defaulters clients belong to category of
feature : {feature}  \n ")

#in any category of feature what are % of clients which are deafulters
plot_for_categorical_defaulters(df_train,'NAME_CONTRACT_TYPE')
```




```
#out of all defaulters what are percentage of defaulter who belong to
any features category
plot_for_defaulters_categorywise(df_train,'NAME_CONTRACT_TYPE')
```



- So we can see cash loans are in higher qunatity so It has more signifinact defaulters and non defaulters
- Revolving credit is a type of credit that does not have a fixed number of payments, in contrast to installment credit. Credit cards are an example of revolving credit used by consumers. Corporate revolving credit facilities are typically used to provide liquidity for a company's day-to-day operations

```
# we can encode cash loan as 1 and revolving loan as zero

#Target guided Ordinal categoris

# map_dict_target=df_train.groupby('NAME_CONTRACT_TYPE')
['TARGET'].mean().to_dict()
# df_train['NAME_CONTRACT_TYPE_fencoded']=
df_train['NAME_CONTRACT_TYPE'].map(map_dict_target)

# map_dict_target

# df_train['NAME_CONTRACT_TYPE_fencoded'].value_counts()

# df_test['NAME_CONTRACT_TYPE'].value_counts()

# df_test['NAME_CONTRACT_TYPE_fencoded']=
df_test['NAME_CONTRACT_TYPE'].map(map_dict_target)
```

```
# df_test['NAME_CONTRACT_TYPE_fencoded'].value_counts()

# map_label_encode= {'Cash loans': 1, 'Revolving loans': 0}

# df_train['NAME_CONTRACT_TYPE_Lencoded']=
df_train['NAME_CONTRACT_TYPE'].map(map_label_encode)

# df_test['NAME_CONTRACT_TYPE_Lencoded']=
df_test['NAME_CONTRACT_TYPE'].map(map_label_encode)

#Lets see Highest correlated column with this column


# df_train.corr()
['NAME_CONTRACT_TYPE_Lencoded'].sort_values(ascending=False)[::6]

# df_train.corr()
['NAME_CONTRACT_TYPE_fencoded'].sort_values(ascending=False)[::6]

#we can see it doesnt mattered here kind of label encoi=ding we are
doing ,SO we will just keep The label Encoding
# df_train.drop("NAME_CONTRACT_TYPE_fencoded",axis=1,inplace=True)
# df_test.drop("NAME_CONTRACT_TYPE_fencoded",axis=1,inplace=True)

#its look little high correlated with FLAG_DOCUMENT_6:Did client
provide document 6
# This could be a document which is submitted in case of cash loans

## DOne
```

CODE_GENDER

Gender of the client

```
df_train['CODE_GENDER'].value_counts()
```

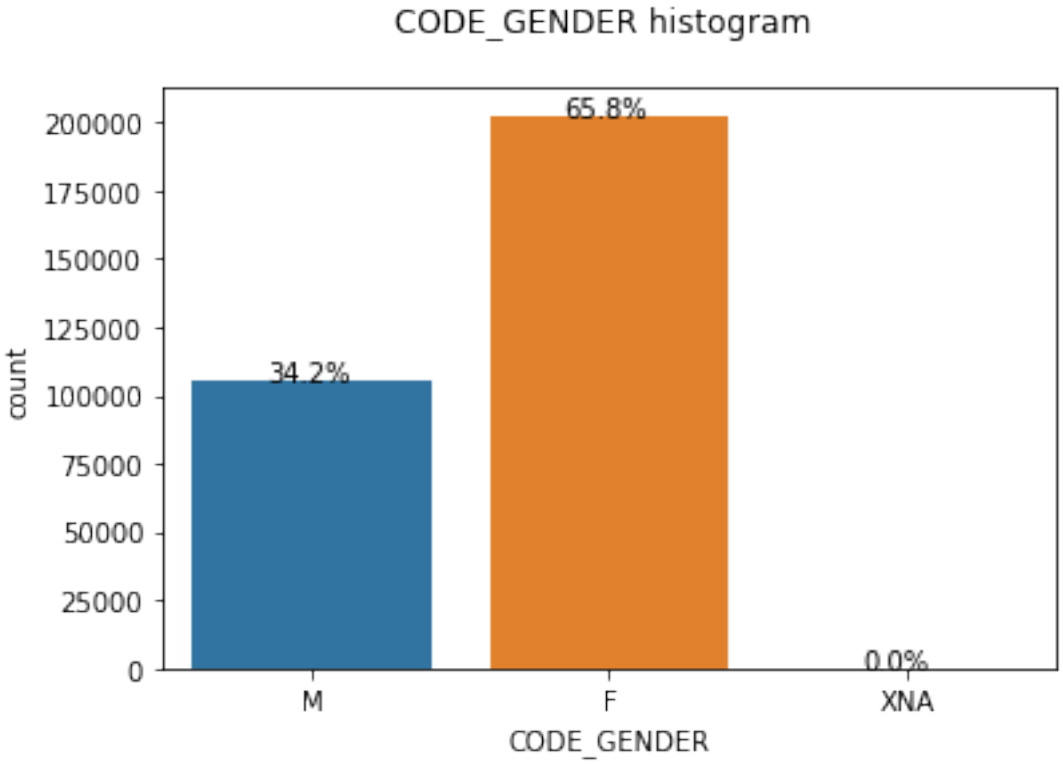
```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

```
#Any null values??
df_train['CODE_GENDER'].isna().sum()
```

```
0
```

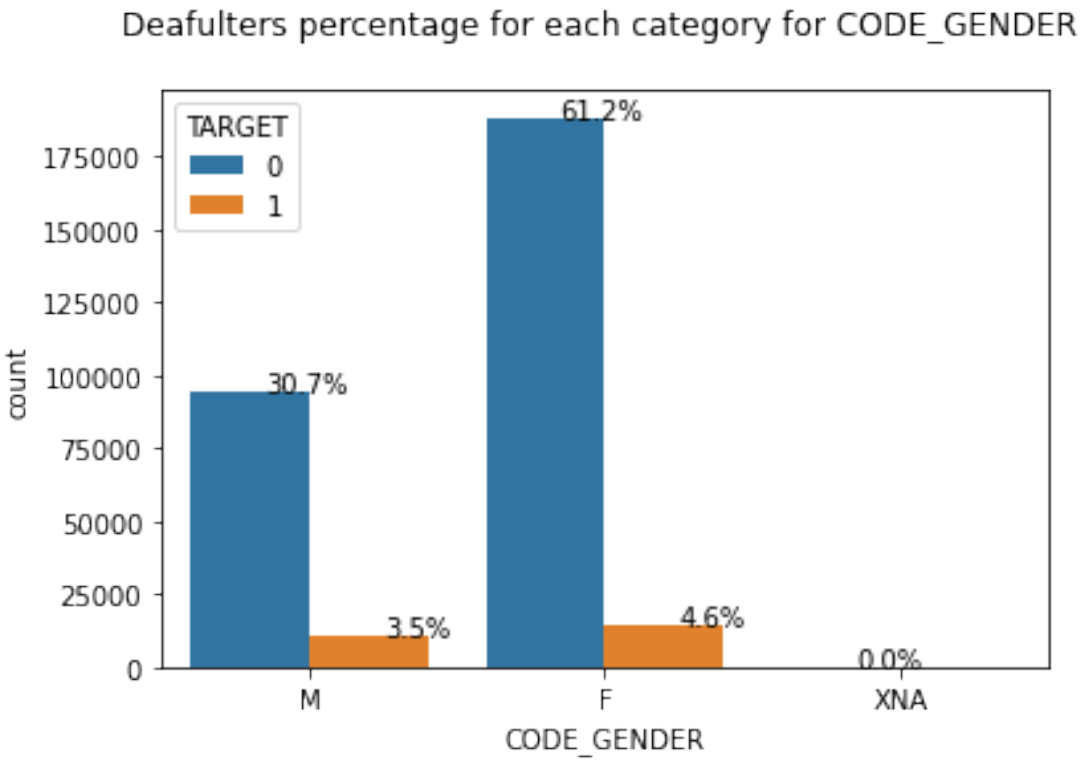
```
#lets see its distribution
ax=sns.countplot(x='CODE_GENDER',data=df_train)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("CODE_GENDER histogram \n")
```

```
Text(0.5, 1.0, 'CODE_GENDER histogram \n')
```




```
#lets see its distribution WRT Target
ax=sns.countplot(x='CODE_GENDER',hue='TARGET',data=df_train)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Deafulters percentage for each category for CODE_GENDER
\n")
```

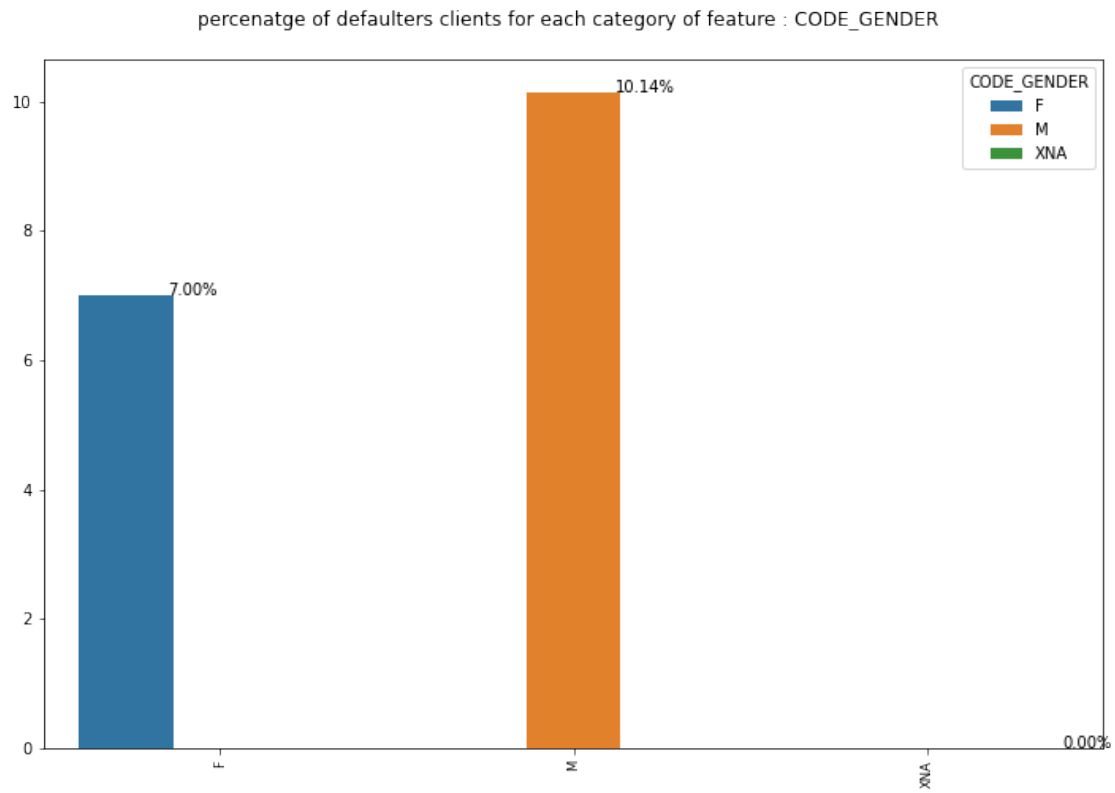
```
Text(0.5, 1.0, 'Deafulters percentage for each category for
CODE_GENDER \n')
```



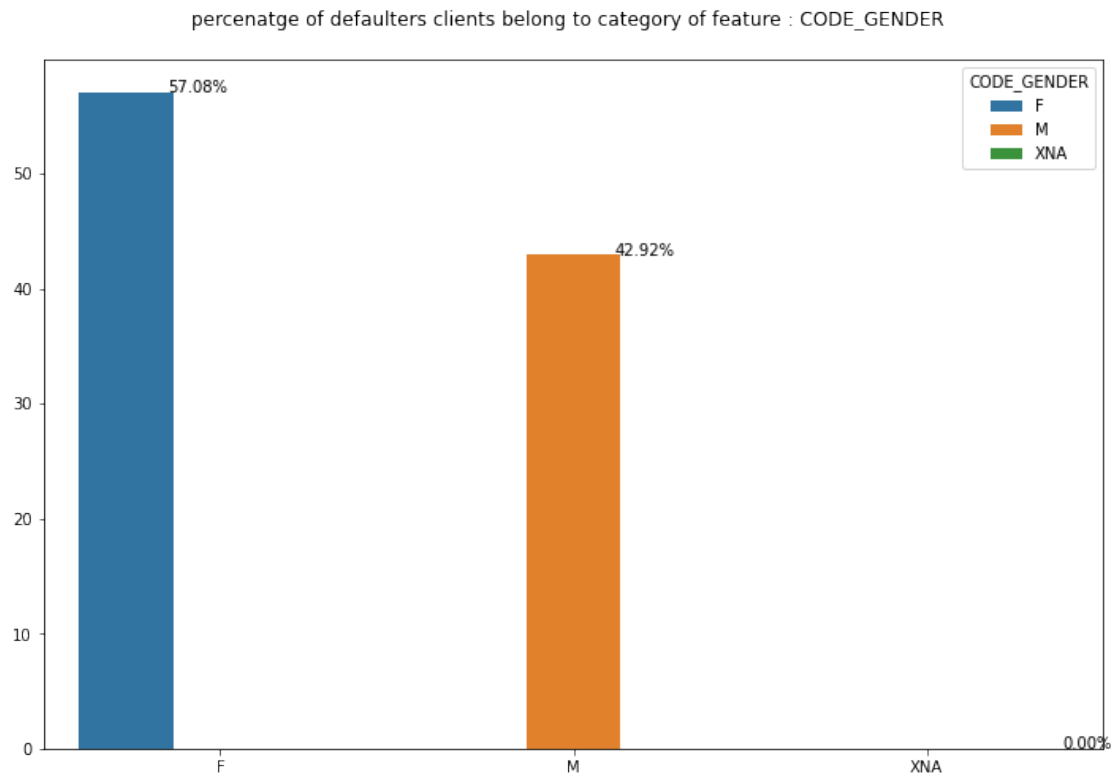
We understood that

- Females tends to apply more for loans
- Men seems to be more defaulter than females

```
#% of  defaulters categorywise
plot_for_defaulters_categorywise(df_train,'CODE_GENDER')
```




```
# % of defaulters in each category

plot_for_categorical_defaulters(df_train,'CODE_GENDER')
```




```
# def encode_CODE_GENDER(df):
#     df['CODE_GENDER']= df['CODE_GENDER'].map({'F':1,'M':0,'XNA':2})

# encode_CODE_GENDER(df_train)
# encode_CODE_GENDER(df_test)
```

- This flag document 8 seems to be any form which have more submiited by female and XNA genders

FLAG_OWN_CAR

```
df_train['FLAG_OWN_CAR'].value_counts()
```

```
N    202924
Y    104587
Name: FLAG_OWN_CAR, dtype: int64
```

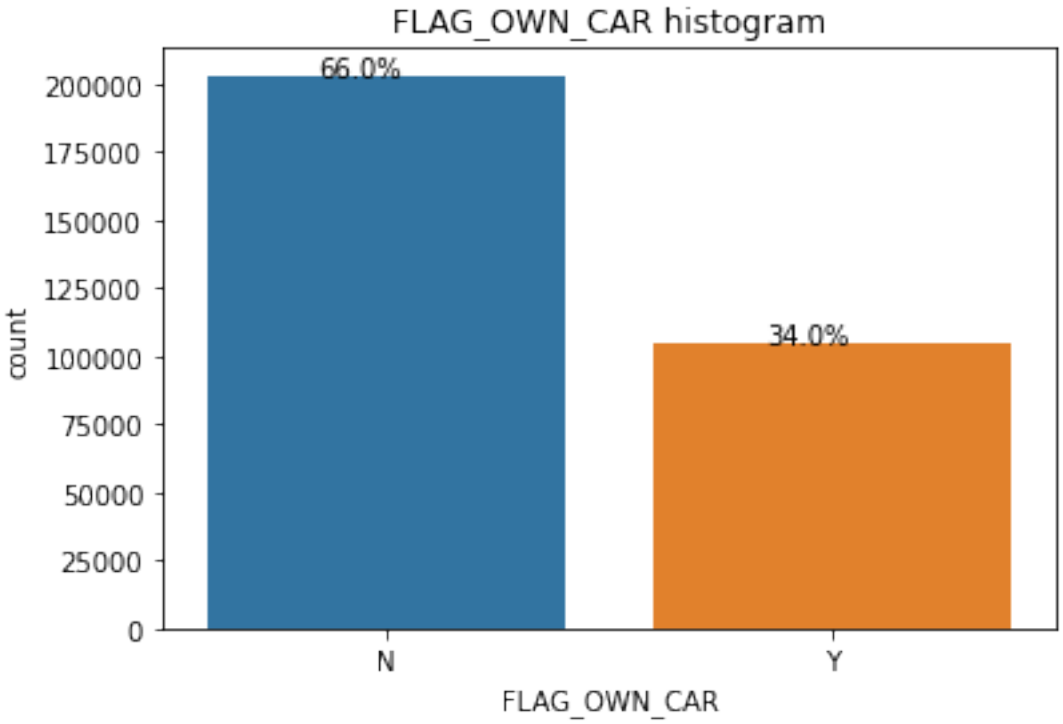
```
#Any null values??
df_train['FLAG_OWN_CAR'].isnull().sum()
```

```
0
```

```
#lets see its distribution
ax=sns.countplot(x='FLAG_OWN_CAR',data=df_train)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
```

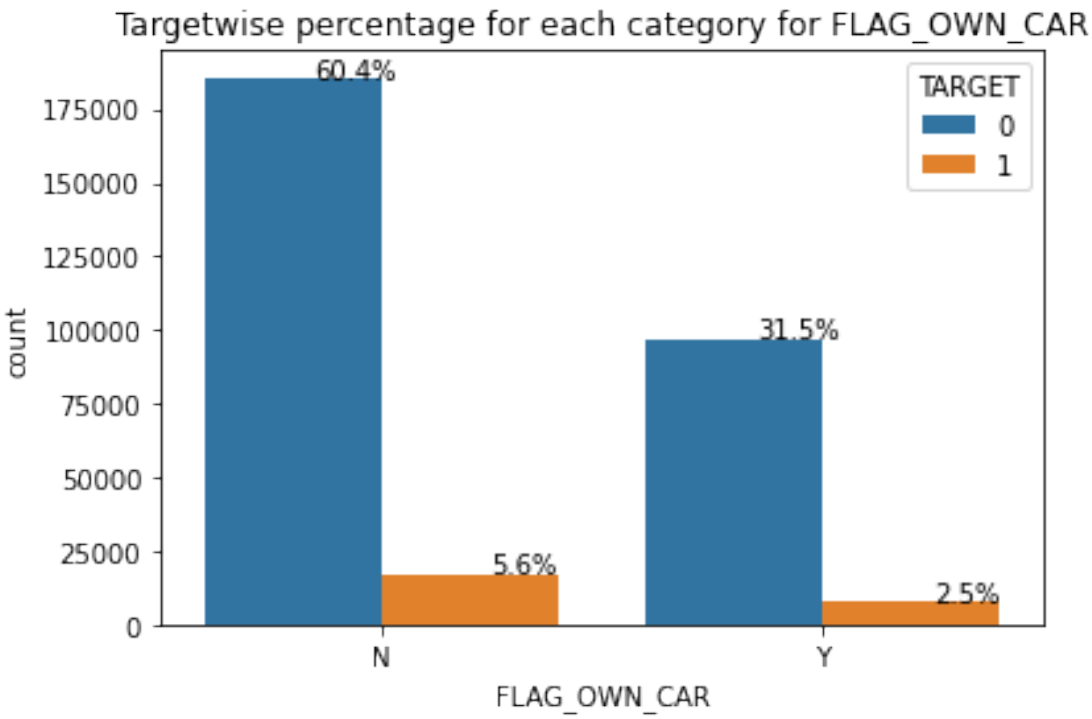
```
ax.set_title("FLAG_OWN_CAR histogram")
```

```
Text(0.5, 1.0, 'FLAG_OWN_CAR histogram')
```

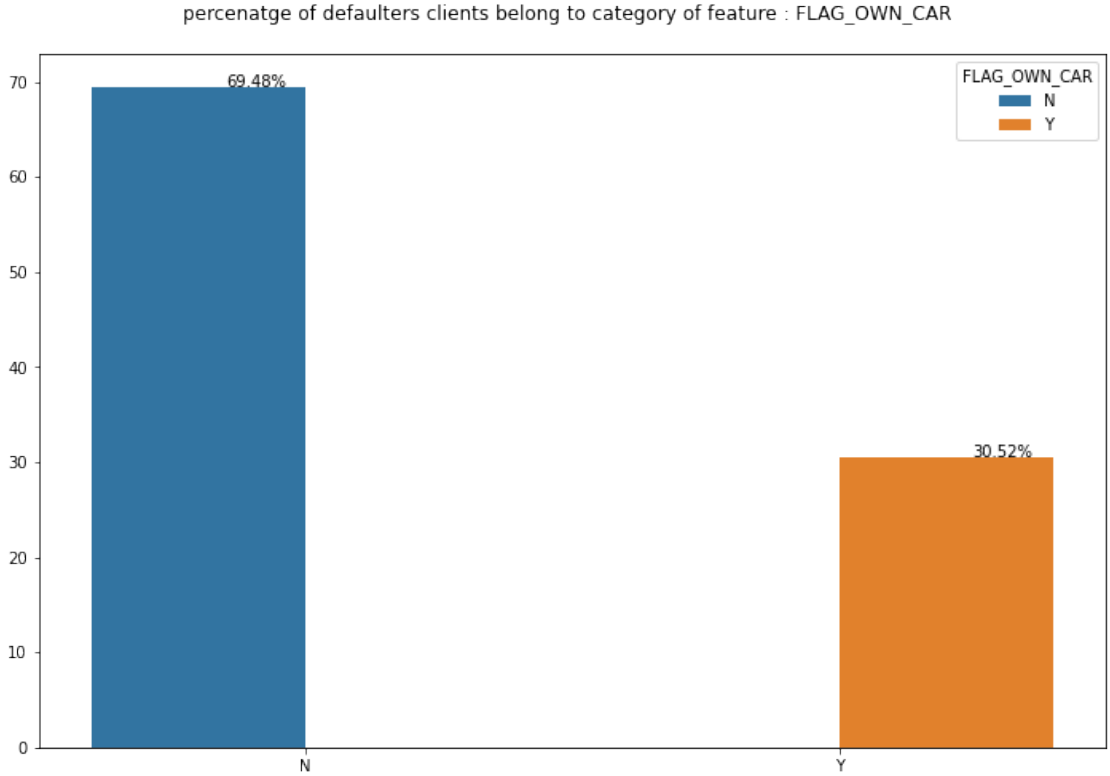



```
#lets see its distribution with respect to target
ax=sns.countplot(x='FLAG_OWN_CAR',hue='TARGET',data=df_train)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
FLAG_OWN_CAR")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FLAG_OWN_CAR')
```

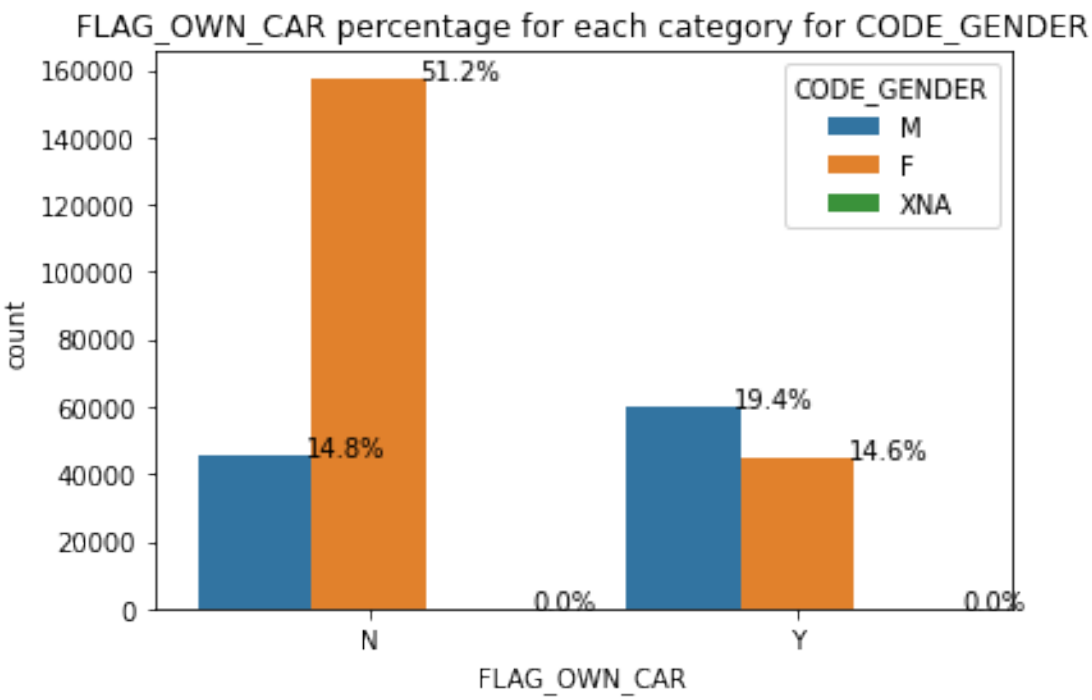



```
#lets see its distribution for defaulters
plot_for_categorical_defaulters(df_train,'FLAG_OWN_CAR')
```



```
#So people not having cars are facing more difficulties for paying the
loan back

# def encode_FLAG_OWN_CAR(df):
#     df['FLAG_OWN_CAR']= df['FLAG_OWN_CAR'].map({'N':1,'Y':0})

# encode_FLAG_OWN_CAR(df_train)
# encode_FLAG_OWN_CAR(df_test)

#lets see its distribution wrt CODE_Gender
ax=sns.countplot(x='FLAG_OWN_CAR',hue='CODE_GENDER',data=df_train)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("FLAG_OWN_CAR percentage for each category for
CODE_GENDER")
```

```
Text(0.5, 1.0, 'FLAG_OWN_CAR percentage for each category for
CODE_GENDER')
```




```
#Its shows Men tend have Car more than women
```

FLAG_OWN_REALTY

Flag if client owns a house or flat

```
df_train['FLAG_OWN_REALTY'].value_counts()
```

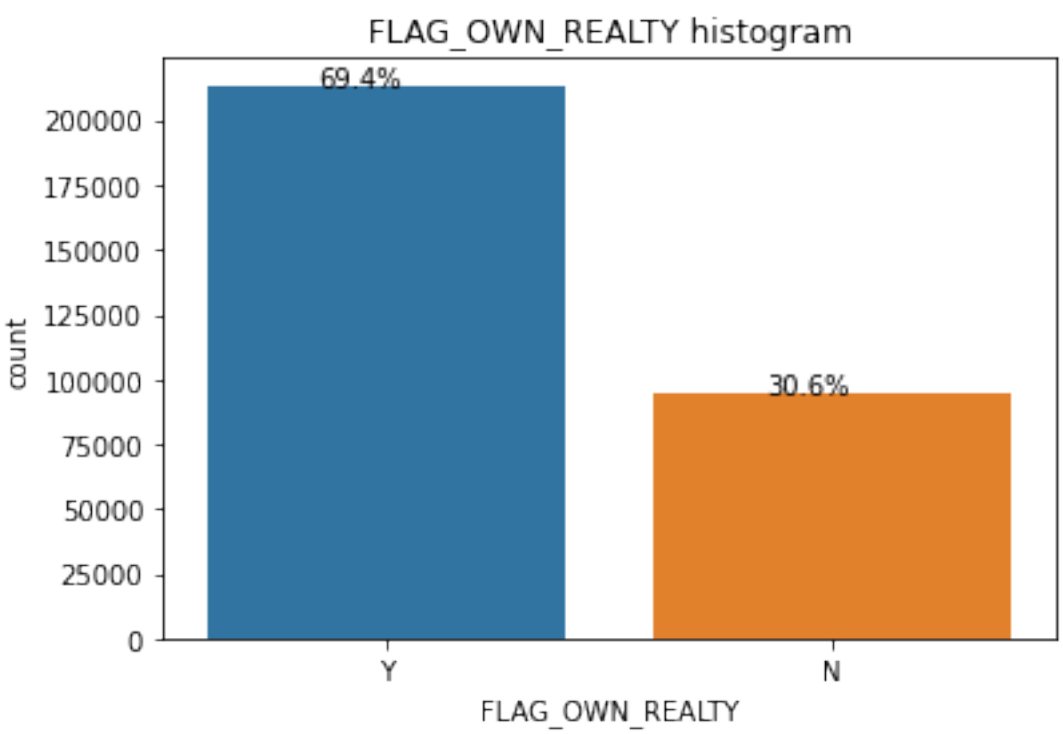
```
Y    213312
N     94199
Name: FLAG_OWN_REALTY, dtype: int64
```

```
#Any null values??
df_train['FLAG_OWN_REALTY'].isnull().sum()
```

```
0
```

```
#lets see its distribution
ax=sns.countplot(x='FLAG_OWN_REALTY',data=df_train)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("FLAG_OWN_REALTY histogram")
```

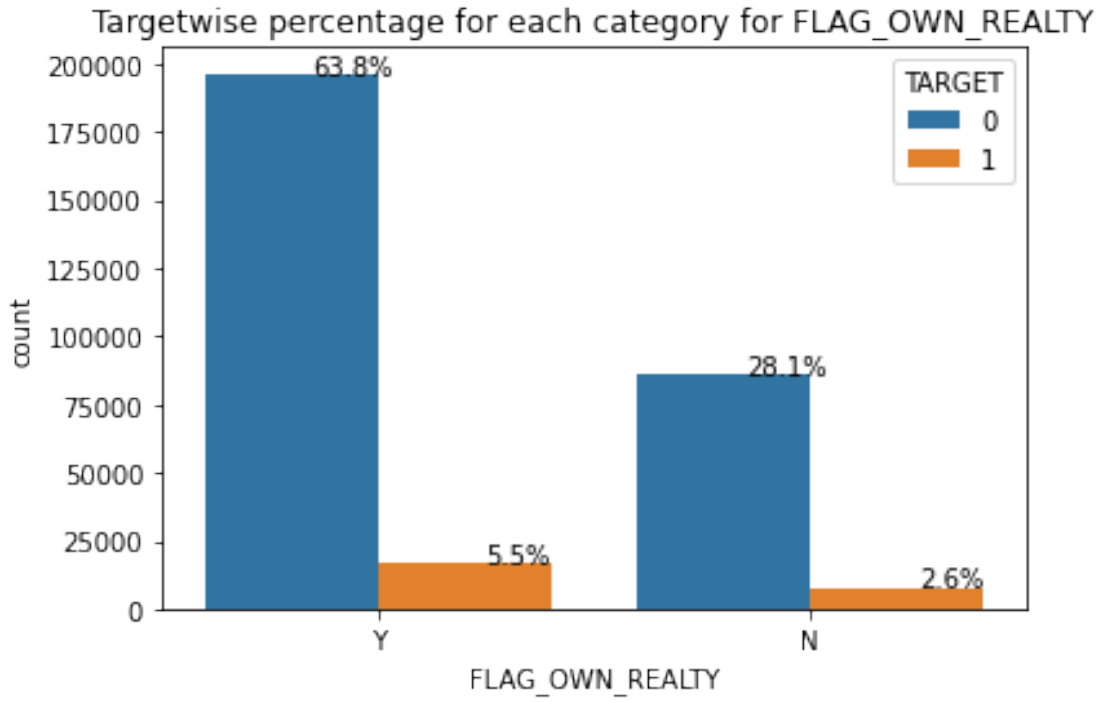
```
Text(0.5, 1.0, 'FLAG_OWN_REALTY histogram')
```



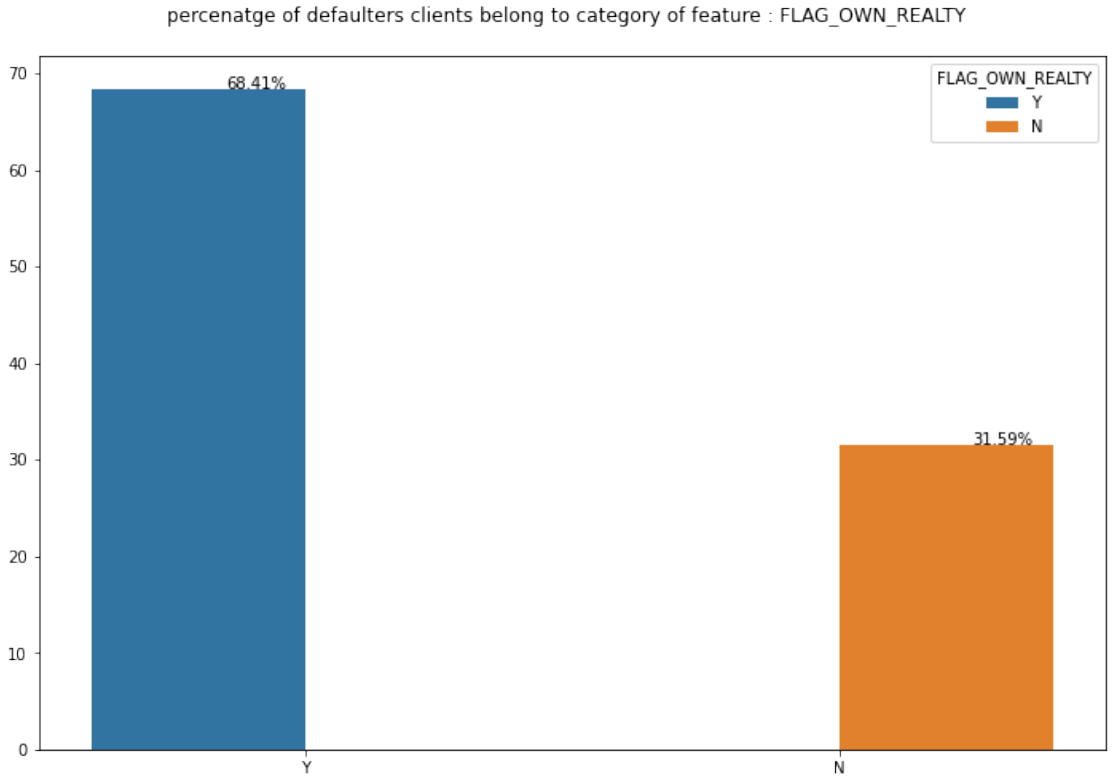

```
#more people applying for laon have their own homes
```

```
#lets see its distribution with resepect to TARGET
ax=sns.countplot(x='FLAG_OWN_REALTY',hue='TARGET',data=df_train)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Targetwise percentage for each category for
FLAG_OWN_REALTY")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FLAG_OWN_REALTY')
```



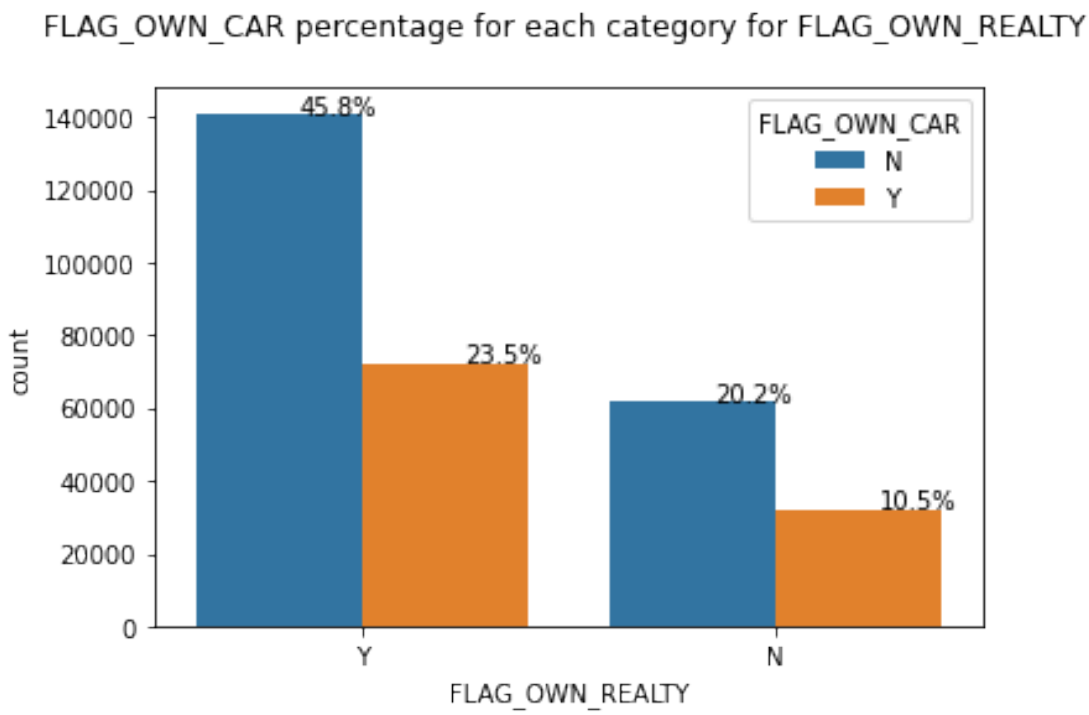

```
plot_for_categorical_defaulters(df_train,'FLAG_OWN_REALTY')
```



- It shows customer having home shows less irregularity in payment than customers who dont have,
- This happens because poor people who dont have home may face issue in payments

```
# we see which are most 5 correlated columns for this column
#we can use labelEncoding
from sklearn.preprocessing import LabelEncoder

# le_FLAG_OWN_REALTY=LabelEncoder()
# le_FLAG_OWN_REALTY.fit(df_train['FLAG_OWN_REALTY'])
#
df_train['FLAG_OWN_REALTY']=le_FLAG_OWN_REALTY.transform(df_train['FLA
G_OWN_REALTY'])
#
df_test['FLAG_OWN_REALTY']=le_FLAG_OWN_REALTY.transform(df_test['FLAG_
OWN_REALTY'])

#df_train['FLAG_OWN_REALTY'].value_counts()

#1: Have home
# 0: dont have home

#So FLAG_OWN_REALTY shows higher corelations with

#EXT_SOURCE_3  :Normalized score from external data source
#ENTRANCES_AVG  :  Normalized information about building where the
client lives,number of entrances Avg
#LANDAREA_AVG  : Normalized information about building where the
client lives,LandArea Avg
#APARTMENTS_MEDI   :   apartment size Median

# This is just building properties

#lets see How many people having Home have Car also
ax=sns.countplot(x='FLAG_OWN_REALTY',hue='FLAG_OWN_CAR',data=df_train)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))


ax.set_title("FLAG_OWN_CAR percentage for each category for
FLAG_OWN_REALTY \n")
```

```
Text(0.5, 1.0, 'FLAG_OWN_CAR percentage for each category for
FLAG_OWN_REALTY \n')
```



- There are more peple who have car and home also

NAME_TYPE_SUITE

Who was accompanying client when he was applying for the loan

```
df_train['NAME_TYPE_SUITE'].value_counts()
```

```
Unaccompanied      248526
Family              40149
Spouse, partner     11370
Children             3267
Other_B              1770
Other_A               866
Group of people       271
Name: NAME_TYPE_SUITE, dtype: int64
```

```
df_train['NAME_TYPE_SUITE'].value_counts().index.to_list()
```

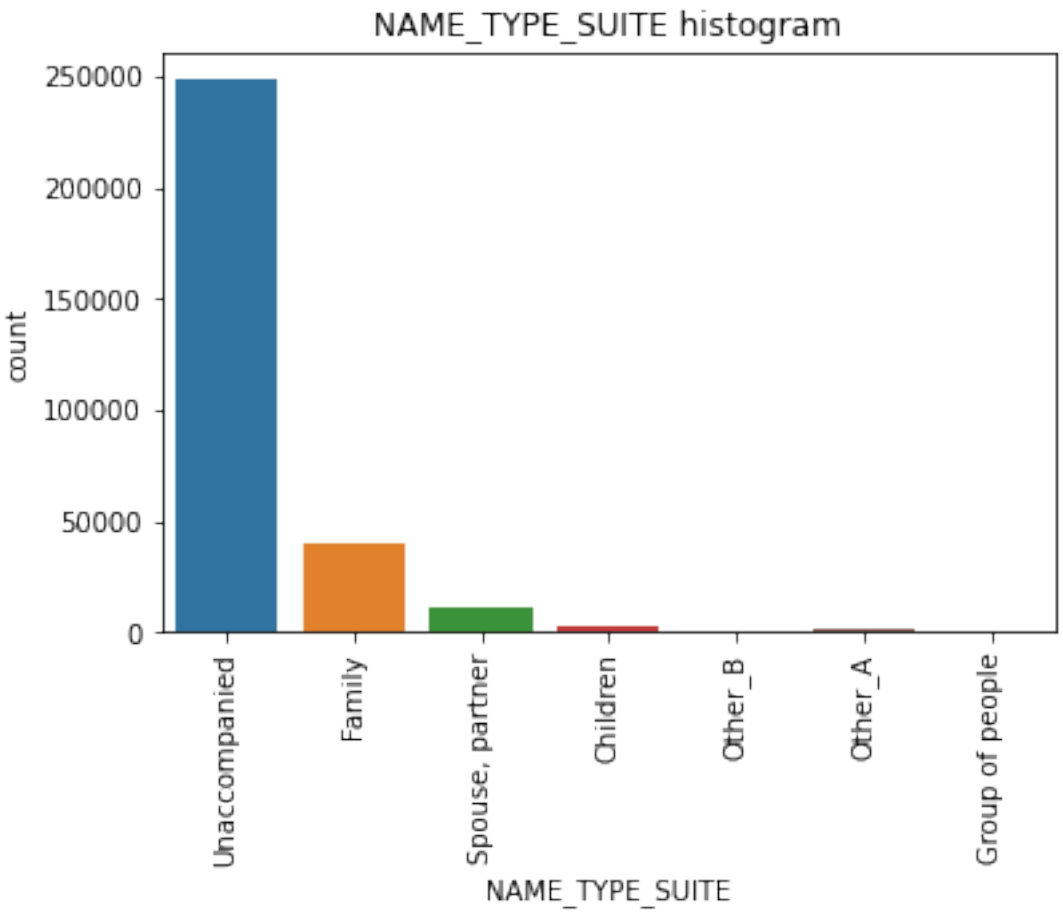
```
['Unaccompanied',
 'Family',
 'Spouse, partner',
 'Children',
 'Other_B',
 'Other_A',
 'Group of people']
```

```
#Any null values??
df_train['NAME_TYPE_SUITE'].isnull().sum()
```

```
1292
```

```
#So this conatins NULL
# I think We can fill NA values with "Unaccompanied" which should be
more obvious

ax=sns.countplot(x='NAME_TYPE_SUITE',data=df_train)
ax.set_xticklabels(df_train['NAME_TYPE_SUITE'].value_counts().index.to
_list(),rotation=90)

ax.set_title("NAME_TYPE_SUITE histogram")
```

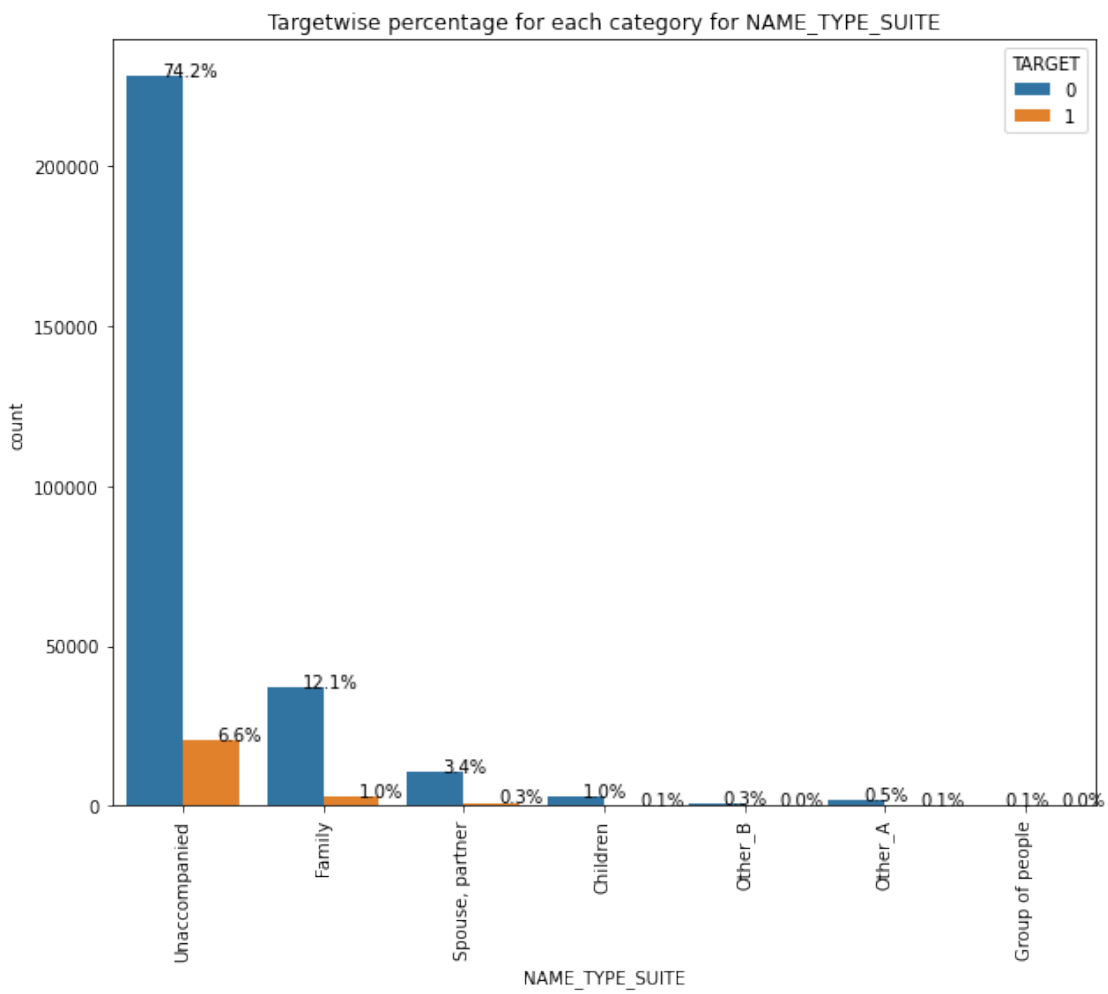
```
Text(0.5, 1.0, 'NAME_TYPE_SUITE histogram')
```




```
#lets see its distribution
plt.figure(figsize=(10,8))
ax=sns.countplot(x='NAME_TYPE_SUITE',hue='TARGET',data=df_train)
ax.set_xticklabels(df_train['NAME_TYPE_SUITE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
```

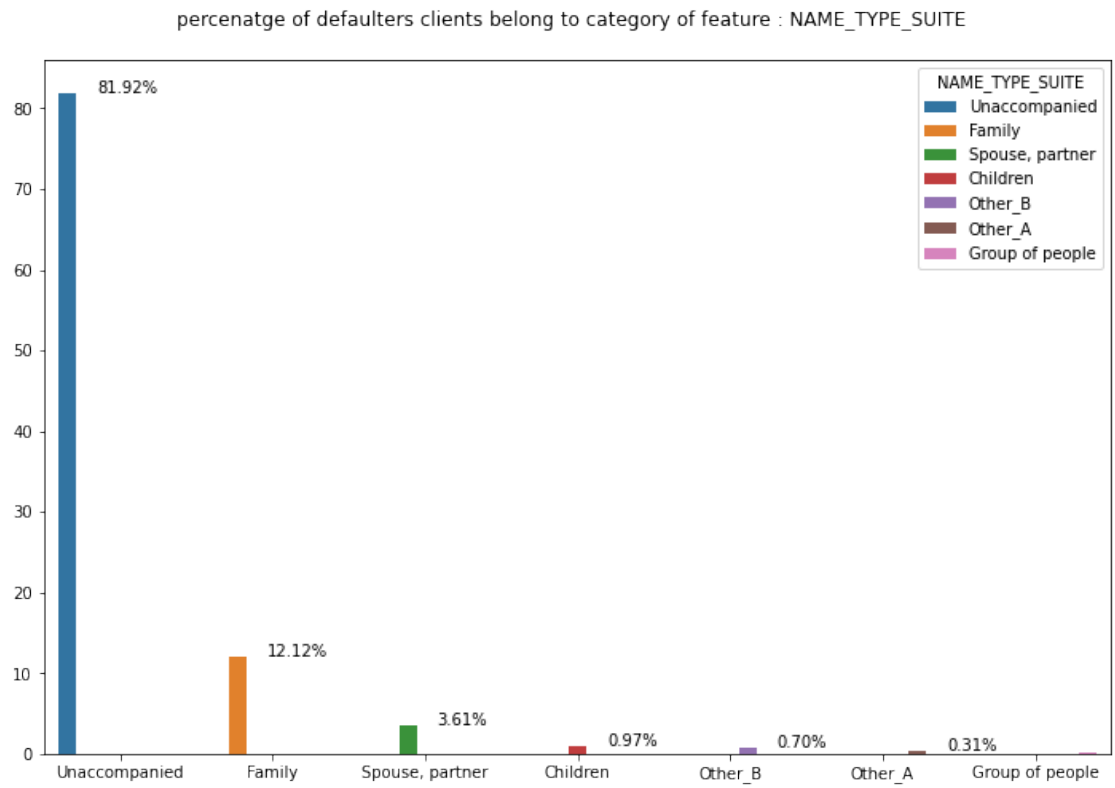
```
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
NAME_TYPE_SUITE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NAME_TYPE_SUITE')
```



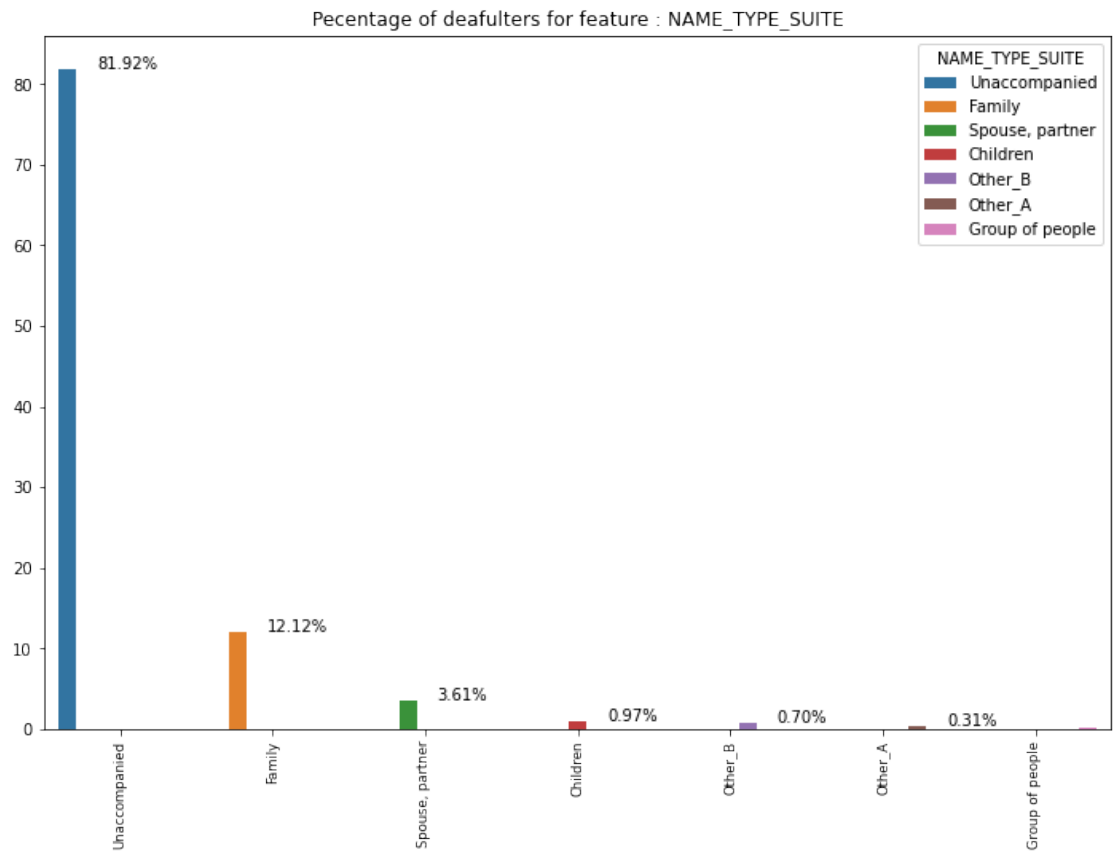
- It shows customer as most of customer were unaccompained while applying for loan ,So it shows more irregulairty in payments -> But this is just volume propotionality ,It dont have any proper meaning I suppose

```
plot_for_categorical_defaulters(df_train,'NAME_TYPE_SUITE')
```




```
#changing to set x labels as 90 rotaion
def plot_for_categorical_defaulters(df,feature):
    categories= df[feature].value_counts().index.to_list()
    category_defaulter_dict=dict()
    for i in categories:
        count_defaulter=df_train[(df[feature]==i) &
(df['TARGET']==1)].shape[0]
        count_total=df_train[(df['TARGET']==1)].shape[0]
        percetage_defaulter= float(count_defaulter*100/count_total)
        category_defaulter_dict[i]=percetage_defaulter
    #return category_defaulter_dict
    keys = list(category_defaulter_dict.keys())
    # get values in the same order as keys, and parse percentage
values
    vals = [float(category_defaulter_dict[k]) for k in keys]
    plt.figure(figsize=(12,8))
    ax=sns.barplot(x=keys, y=vals,hue=keys)

#ax.set_xticklabels(df_train[feature].value_counts().index.to_list(),r
otation=45)
    plt.xticks(rotation=90,fontsize="small")
    plt.title(f"Pecentage of deafulters for feature : {feature}")
    ax.legend(title=feature,loc='best')
    for p in ax.patches:
        ax.annotate('{:.2f}%'.format(p.get_height()), (p.get_x()+0.25,
```

```
p.get_height()+0.01))
    #return category_defaulter_dict

plot_for_categorical_defaulters(df_train,'NAME_TYPE_SUITE')
```




```
#Now its more clear, It doesnt matters as who accompany your with loan
applying

#We will first fill NA values with the "Unaccompanied" and then do
label encoding for this feature
#df_train['NAME_TYPE_SUITE']=
df_train['NAME_TYPE_SUITE'].fillna('Unaccompanied')
#df_test['NAME_TYPE_SUITE']=
df_test['NAME_TYPE_SUITE'].fillna('Unaccompanied')


#we can use labelEncoding
# from sklearn.preprocessing import LabelEncoder

# le_NAME_TYPE_SUITE=LabelEncoder()
# le_NAME_TYPE_SUITE.fit(df_train['NAME_TYPE_SUITE'])
#
df_train['NAME_TYPE_SUITE']=le_NAME_TYPE_SUITE.transform(df_train['NAM
E_TYPE_SUITE'])
#
```

```
df_test['NAME_TYPE_SUITE']=le_NAME_TYPE_SUITE.transform(df_test['NAME_
TYPE_SUITE'])

#df_train['NAME_TYPE_SUITE'].value_counts()
```

NAME_INCOME_TYPE

Clients income type (businessman, working, maternity leave etc)

```
#We will first fill NA values with the "Unaccompanied" and then do
label encoding for this feature
df_train['NAME_INCOME_TYPE'].value_counts()
```

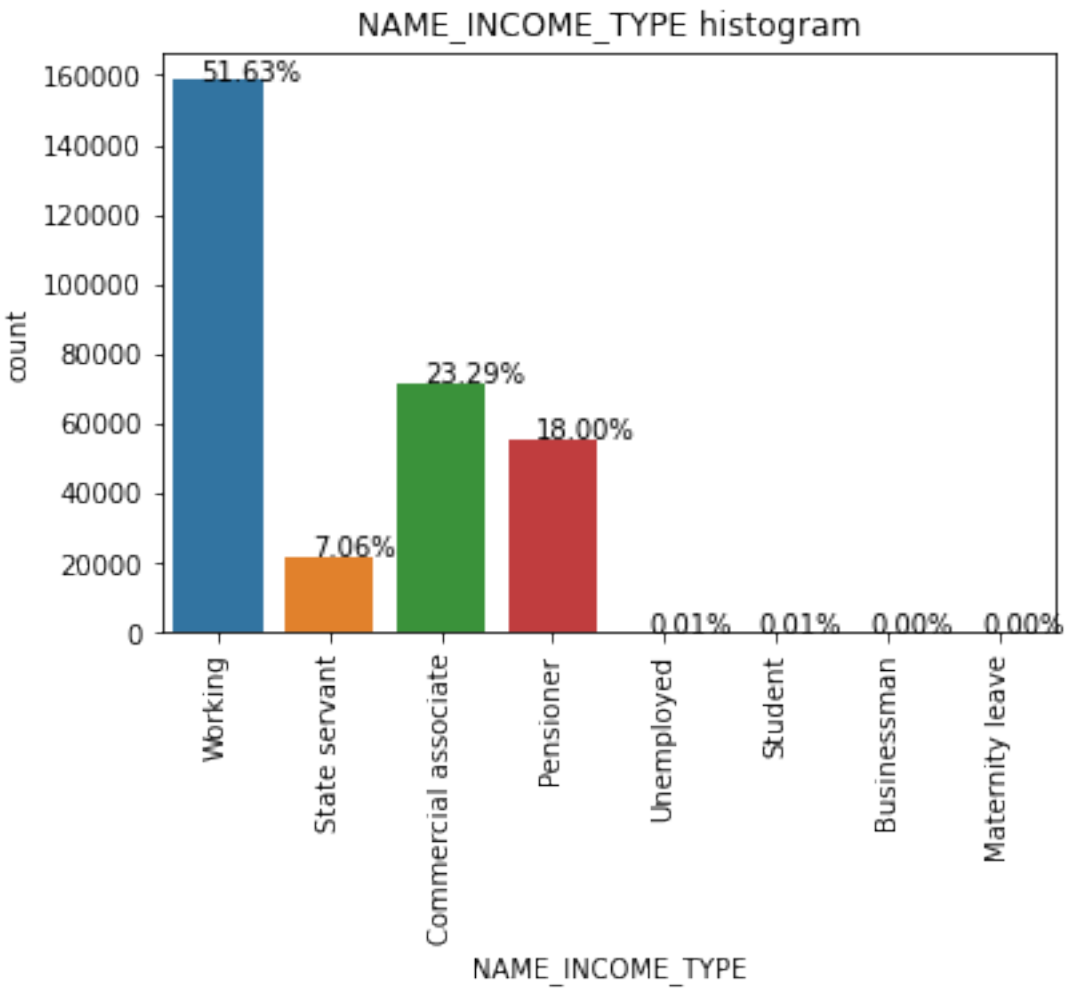
```
Working                 158774
Commercial associate     71617
Pensioner                55362
State servant            21703
Unemployed                  22
Student                     18
Businessman                 10
Maternity leave              5
Name: NAME_INCOME_TYPE, dtype: int64
```

```
df_train['NAME_INCOME_TYPE'].isna().sum()
#No NA values
```

```
0
```

```
#lets see its distribution

ax=sns.countplot(x='NAME_INCOME_TYPE',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("NAME_INCOME_TYPE histogram")
```

```
Text(0.5, 1.0, 'NAME_INCOME_TYPE histogram')
```

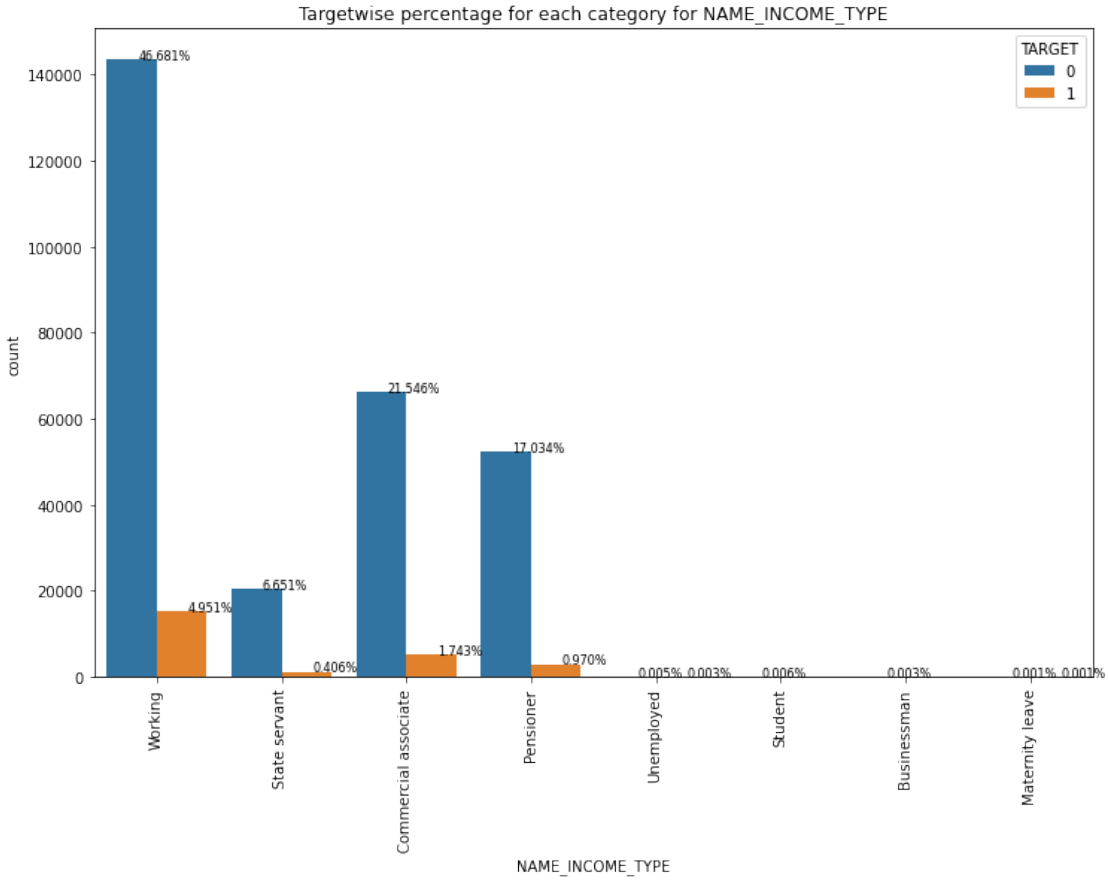


Most loan seekers are in descending orders as

```
- Working Professionals
- Pensioners
- State Servent
- Commercial Associates
```

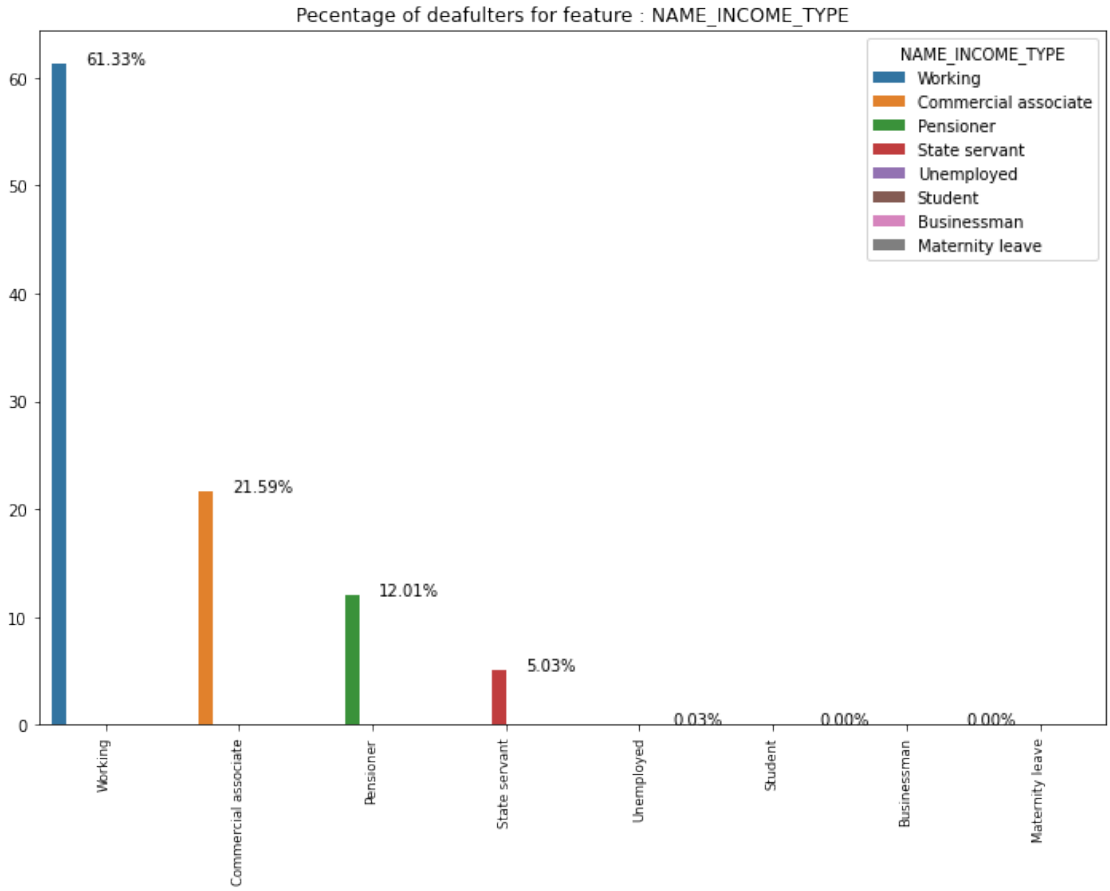
```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_INCOME_TYPE',hue='TARGET',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),size=8)
ax.set_title("Targetwise percentage for each category for
NAME_INCOME_TYPE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NAME_INCOME_TYPE')
```

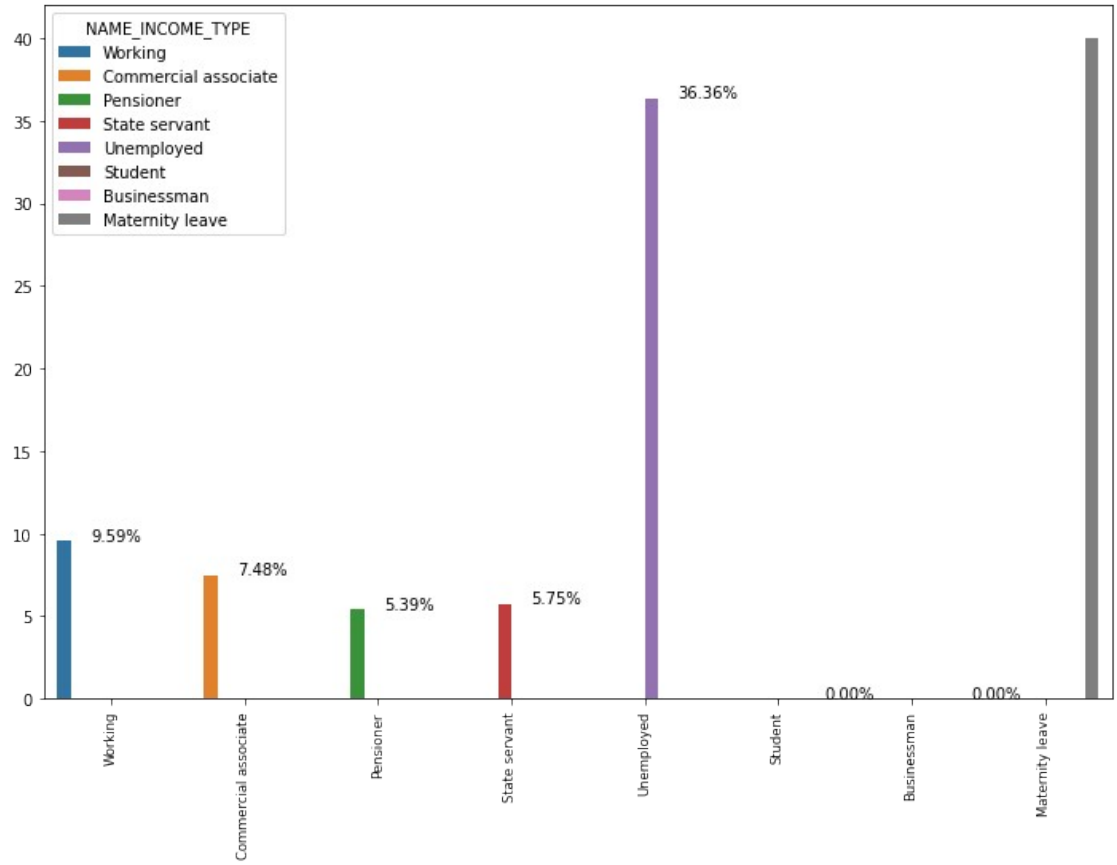



```
plot_for_categorical_defaulters(df_train,'NAME_INCOME_TYPE')
```

```
{'Working': 61.32527693856999,
 'Commercial associate': 21.59113796576032,
 'Pensioner': 12.012084592145015,
 'State servant': 5.031218529707956,
 'Unemployed': 0.032225579053373615,
 'Student': 0.0,
 'Businessman': 0.0,
 'Maternity leave': 0.008056394763343404}
```




```
#lets see in each category what is % of defaulters of same category
plot_for_deafulters_categorywise(df_train,'NAME_INCOME_TYPE')
```



- Its seems people who are unemployed have more problem in replayment of loan,But if we see the value counts, The uneployed are just 22 So 8 are deafaulters.We are not sure actually how a client NAME_INCOME_TYPE affects replayment

NAME_EDUCATION_TYPE

Level of highest education the client achieved

```
df_train['NAME_EDUCATION_TYPE'].value_counts().to_dict()
```

```
{'Secondary / secondary special': 218391,
 'Higher education': 74863,
 'Incomplete higher': 10277,
 'Lower secondary': 3816,
 'Academic degree': 164}
```

```
df_defaulter['NAME_EDUCATION_TYPE'].value_counts().to_dict()
```

```
{'Secondary / secondary special': 19524,
 'Higher education': 4009,
 'Incomplete higher': 872,
 'Lower secondary': 417,
 'Academic degree': 3}
```

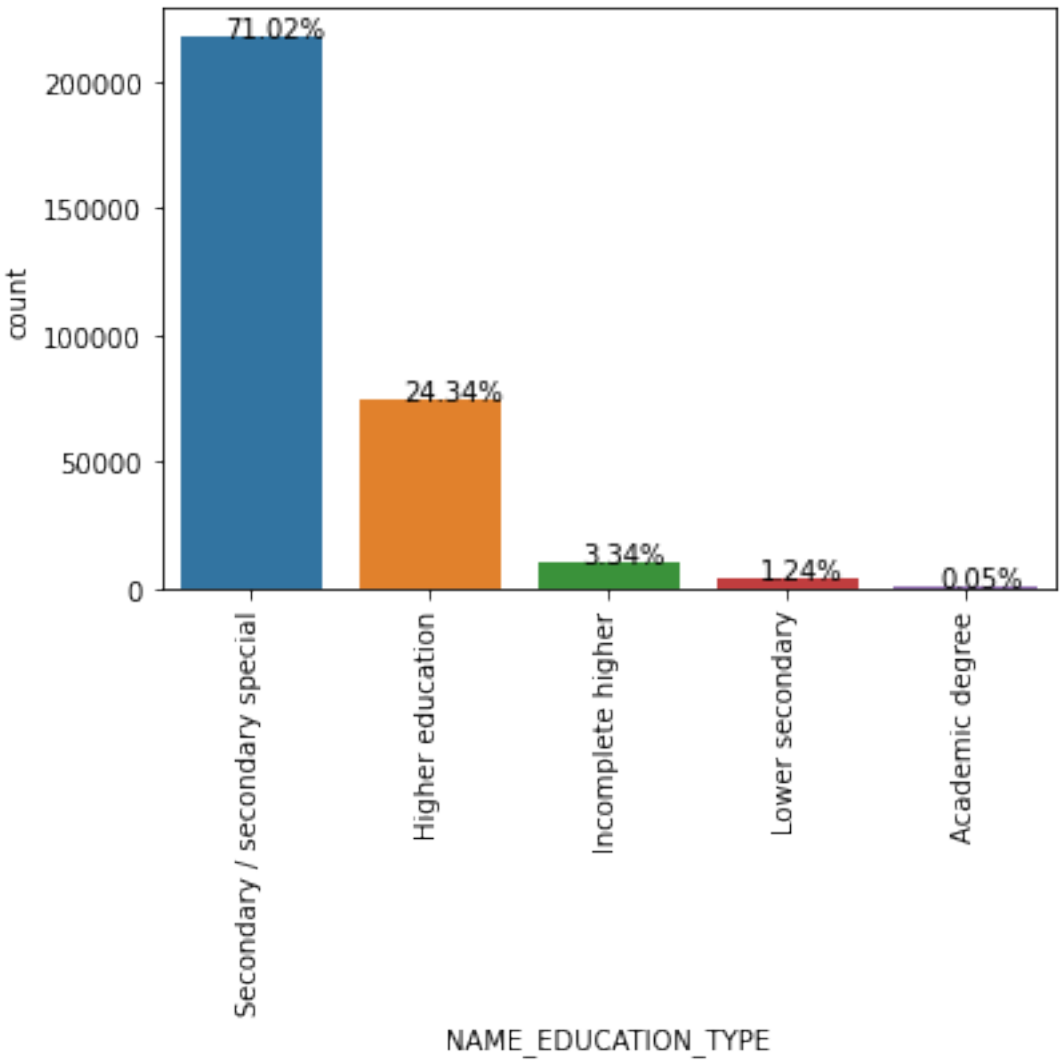
```
df_train['NAME_EDUCATION_TYPE'].isna().sum()
#No NA values
```

```
0
```

```
for k,v in
zip(df_train['NAME_EDUCATION_TYPE'].value_counts().to_dict(),df_defaul
ter['NAME_EDUCATION_TYPE'].value_counts().to_dict()):
    print(k,v)
    #print(np.round((v2/v1)))
```

```
Secondary / secondary special Secondary / secondary special
Higher education Higher education
Incomplete higher Incomplete higher
Lower secondary Lower secondary
Academic degree Academic degree
```

```
#lets see its distribution

ax=sns.countplot(x='NAME_EDUCATION_TYPE',data=df_train)
ax.set_xticklabels(df_train['NAME_EDUCATION_TYPE'].value_counts().inde
x.to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
```

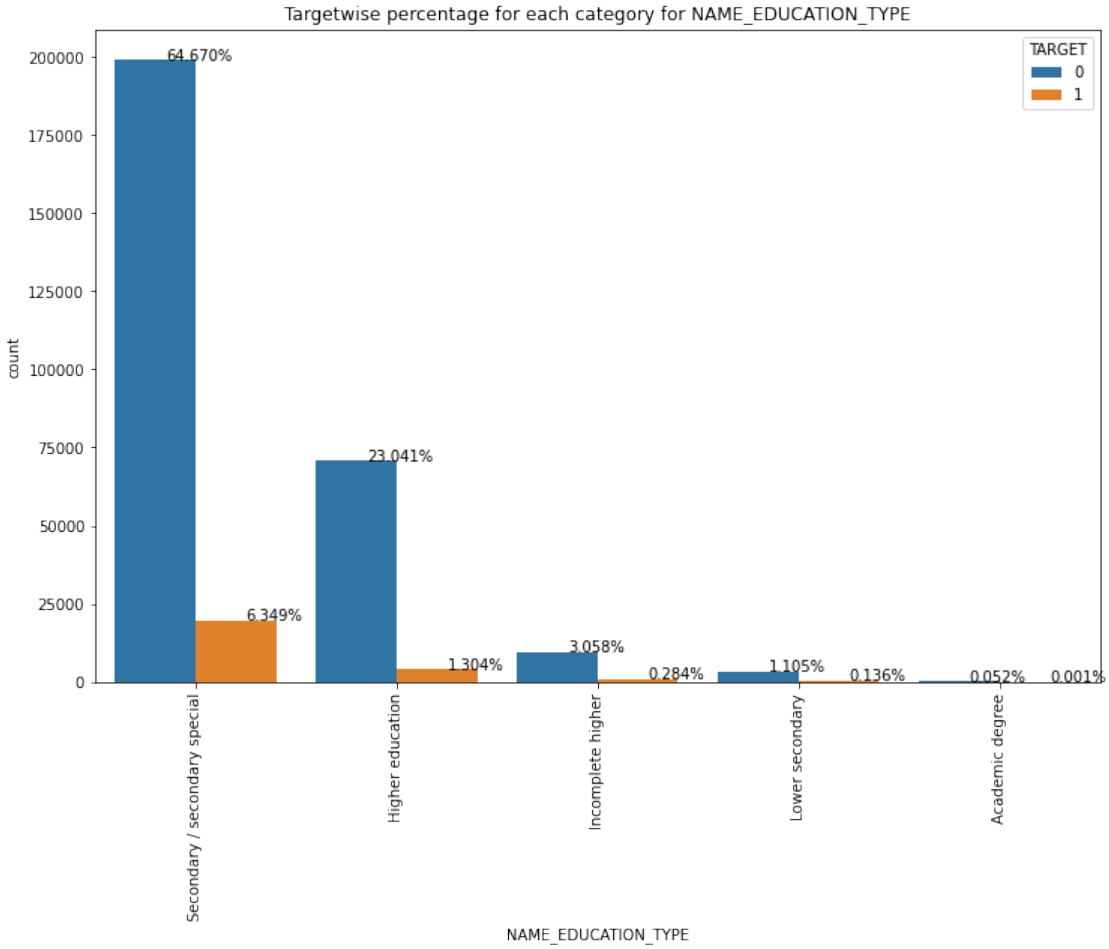


Most loan seekers are in descending orders as

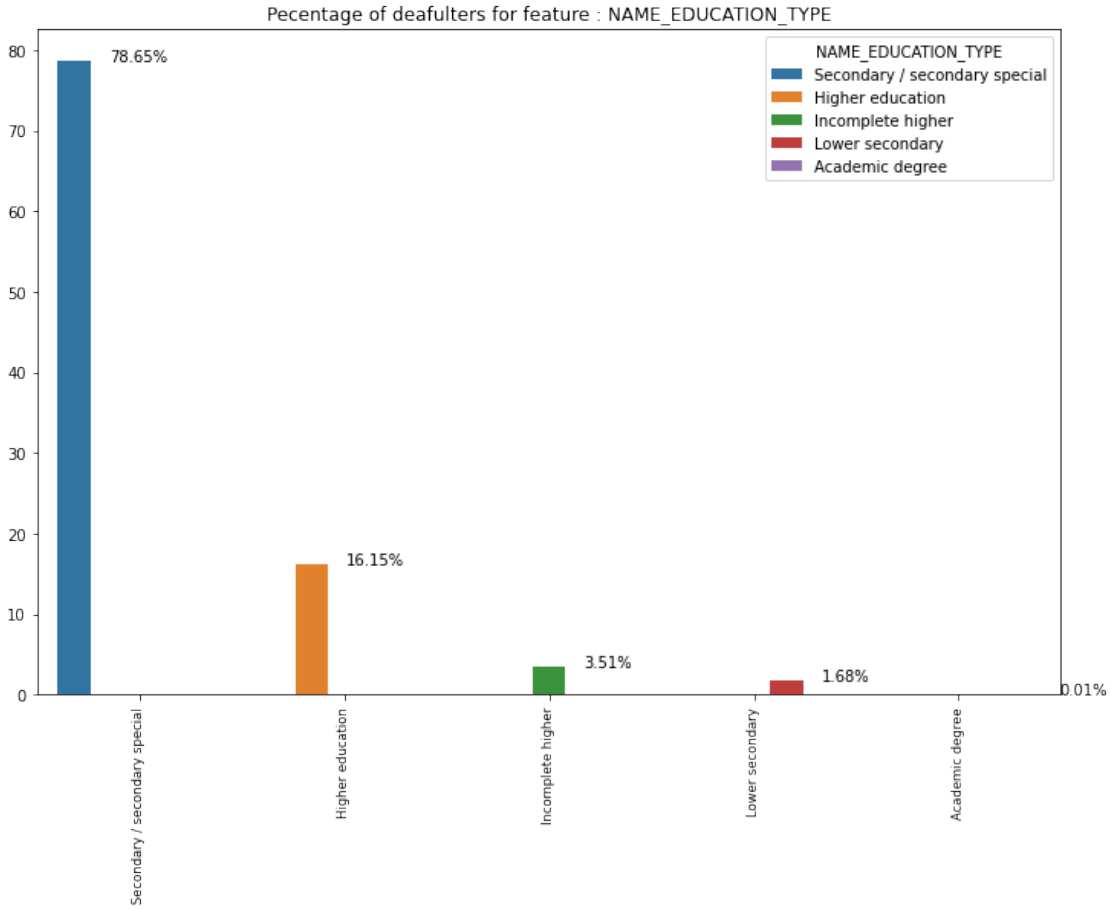
- Secondary / secondary special
- Higher Education

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_EDUCATION_TYPE',hue='TARGET',data=df_train)
ax.set_xticklabels(df_train['NAME_EDUCATION_TYPE'].value_counts().inde
x.to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
NAME_EDUCATION_TYPE")
```

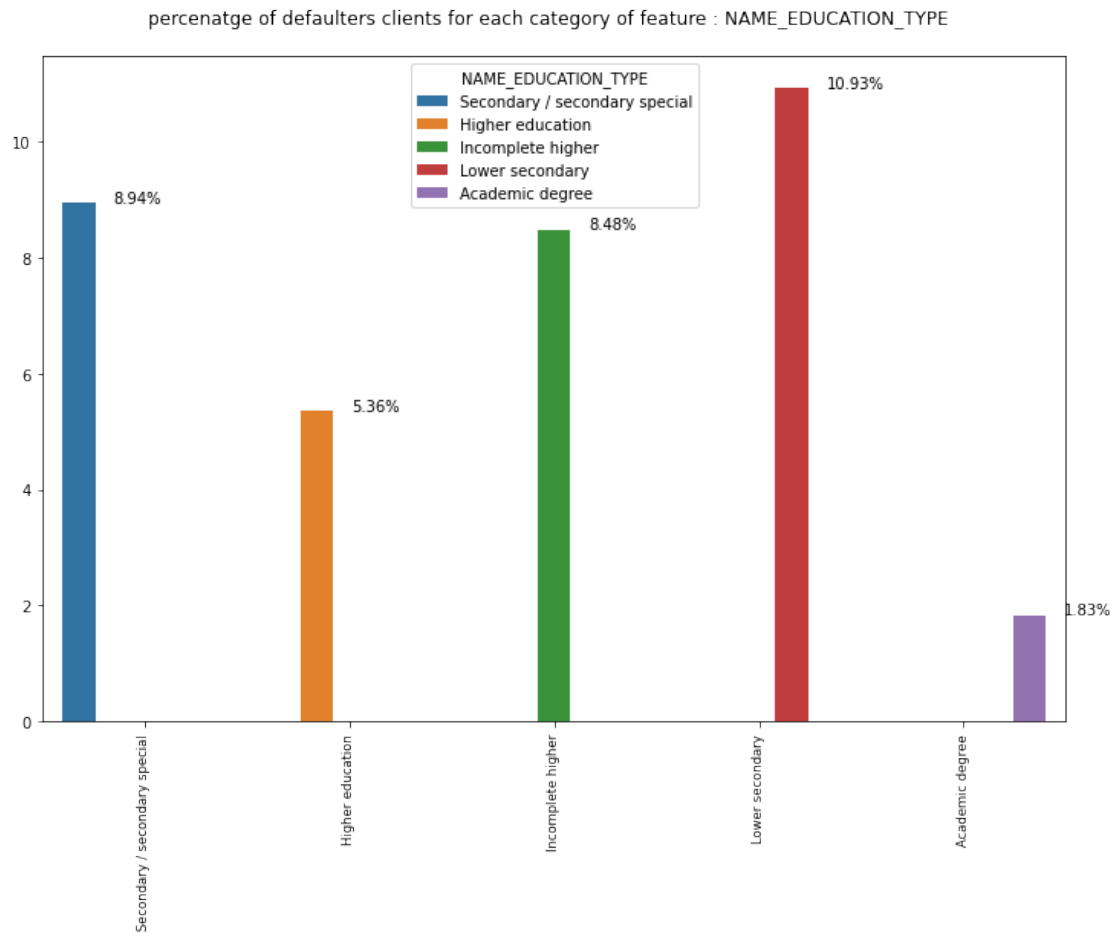
```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NAME_EDUCATION_TYPE')
```




```
plot_for_categorical_defaulters(df_train,'NAME_EDUCATION_TYPE')
```




```
plot_for_defaulters_categorywise(df_train,'NAME_EDUCATION_TYPE')
```

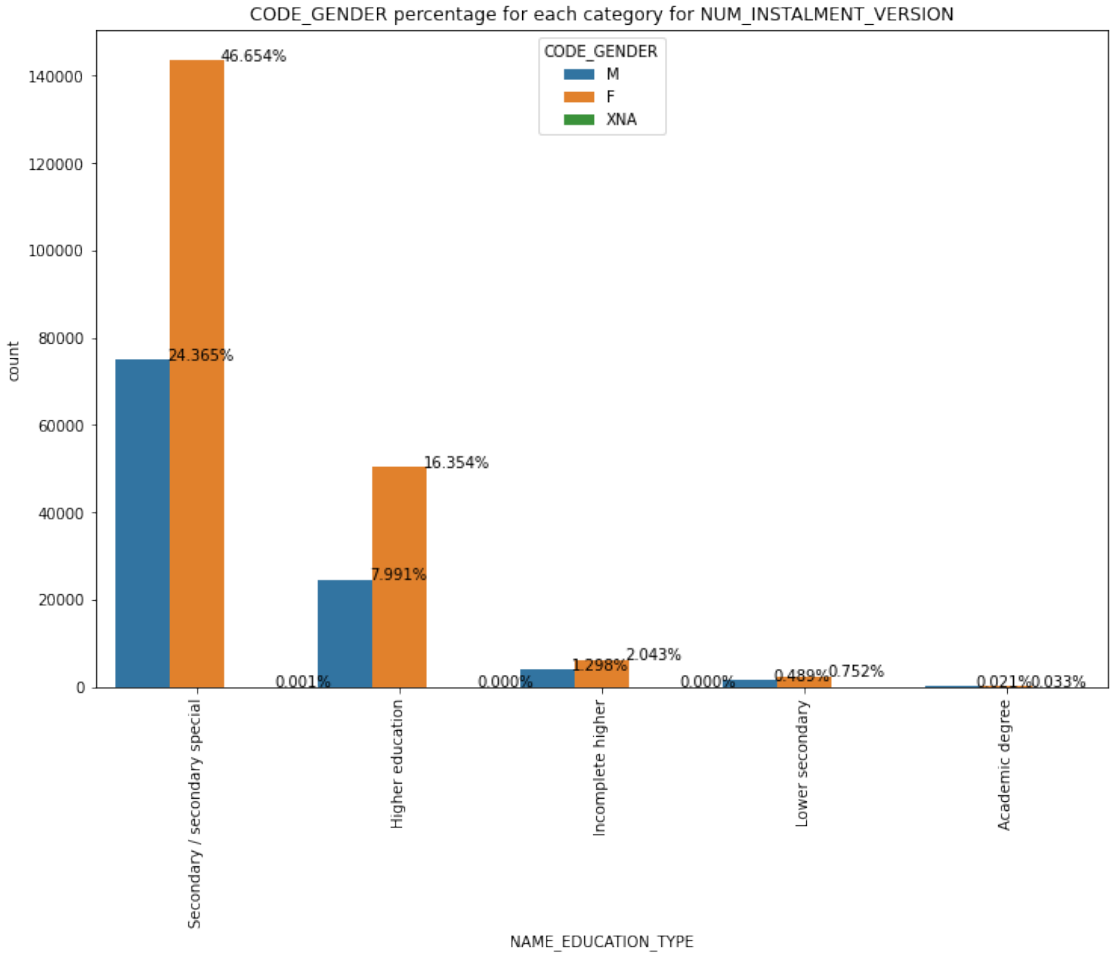


it seems people having education qulaifications as

```
- Secondary /secondary special and below Secondary are facing more
problem in repayment of loan because those are labourer or lower
middle class people
- Also the number of defaulters are in same propotion as the
ditribution of feature category
```

```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_EDUCATION_TYPE',hue='CODE_GENDER',data=df_tra
in)
ax.set_xticklabels(df_train['NAME_EDUCATION_TYPE'].value_counts().inde
x.to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("CODE_GENDER percentage for each category for
NUM_INSTALMENT_VERSION")
```

```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
NUM_INSTALMENT_VERSION')
```




```
#Here women seems to be more educated than men actually
```

NAME_FAMILY_STATUS

Family status of the client

```
df_train['NAME_FAMILY_STATUS'].value_counts().to_dict()
```

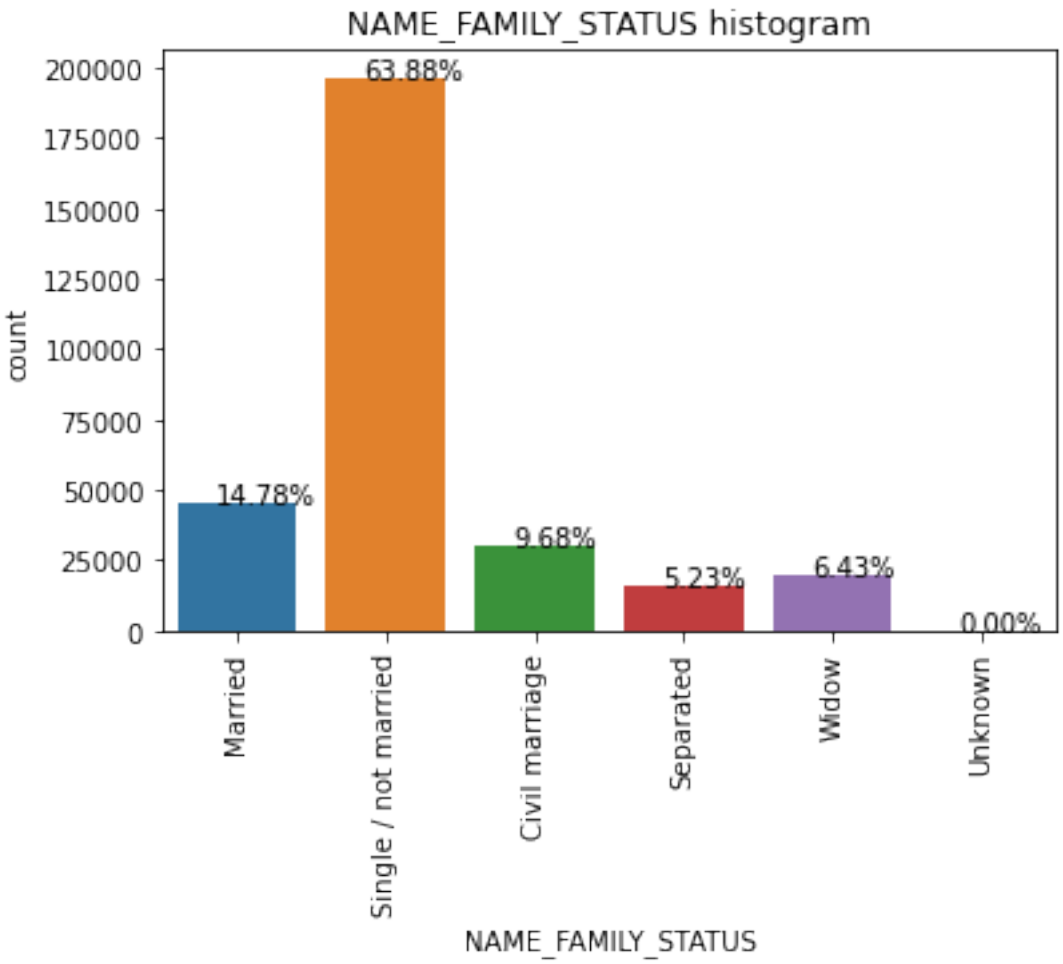
```
{'Married': 196432,
 'Single / not married': 45444,
 'Civil marriage': 29775,
 'Separated': 19770,
 'Widow': 16088,
 'Unknown': 2}
```

```
df_train['NAME_FAMILY_STATUS'].isna().sum()
#No NA values
```

```
0
```

```
#lets see its distribution

ax=sns.countplot(x='NAME_FAMILY_STATUS',data=df_train)
ax.set_xticklabels(df_train['NAME_FAMILY_STATUS'].value_counts().index
.to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("NAME_FAMILY_STATUS histogram")
```

```
Text(0.5, 1.0, 'NAME_FAMILY_STATUS histogram')
```



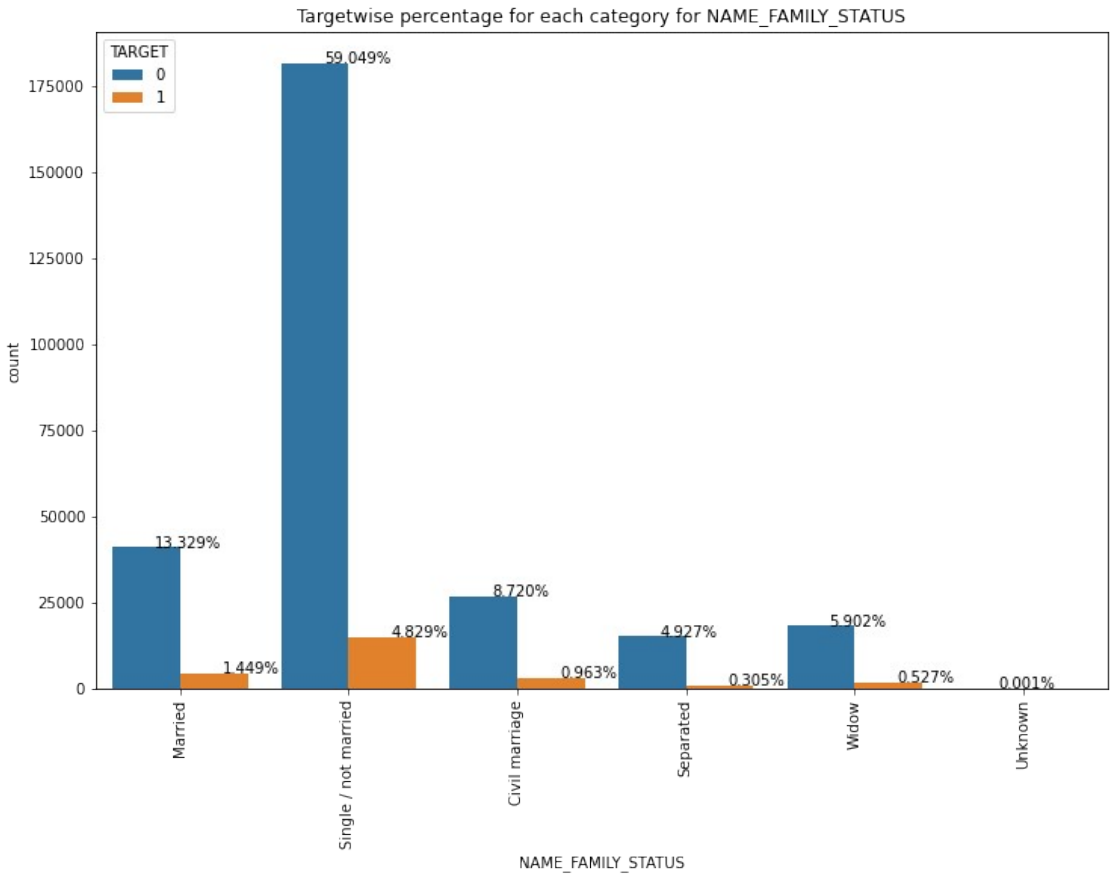
Most loan seekers are in descending orders as

- Singles
- Marraige

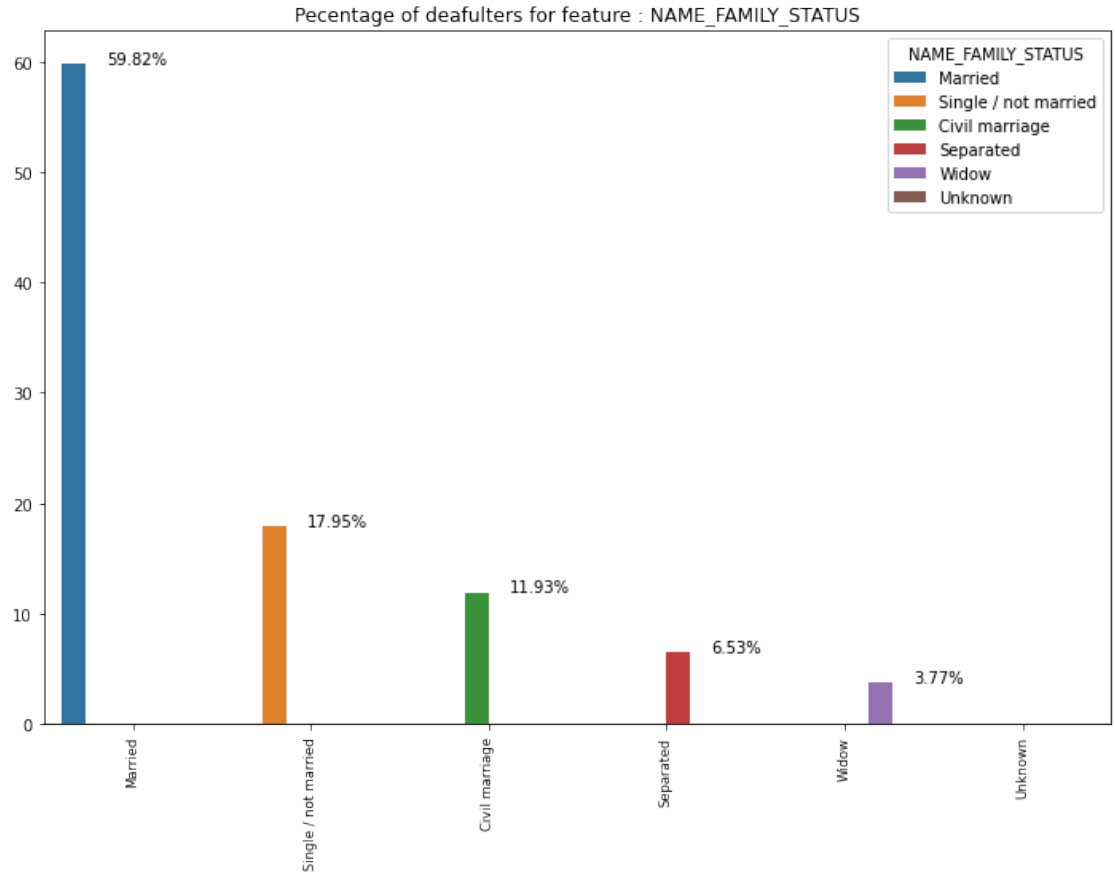
```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_FAMILY_STATUS',hue='TARGET',data=df_train)
ax.set_xticklabels(df_train['NAME_FAMILY_STATUS'].value_counts().index
.to_list(),rotation=90)
```

```
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Targetwise percentage for each category for
NAME_FAMILY_STATUS")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NAME_FAMILY_STATUS')
```




```
plot_for_categorical_defaulters(df_train,'NAME_FAMILY_STATUS')
```

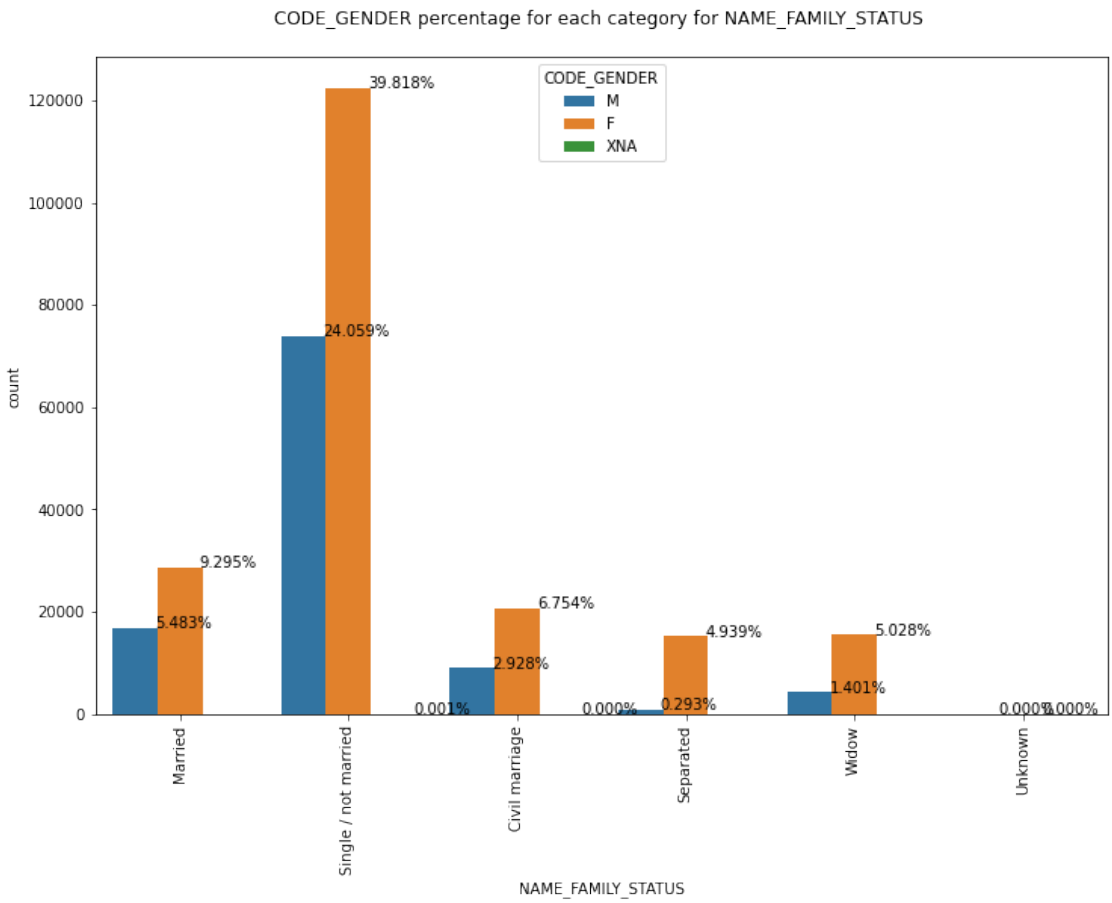


it seems people having Family status as in descending order

```
- Married: See in distribution , count is low but deafulter count are
high
- singles : better not good
- civil marriages

are facing more problem in repayment of loan.
```

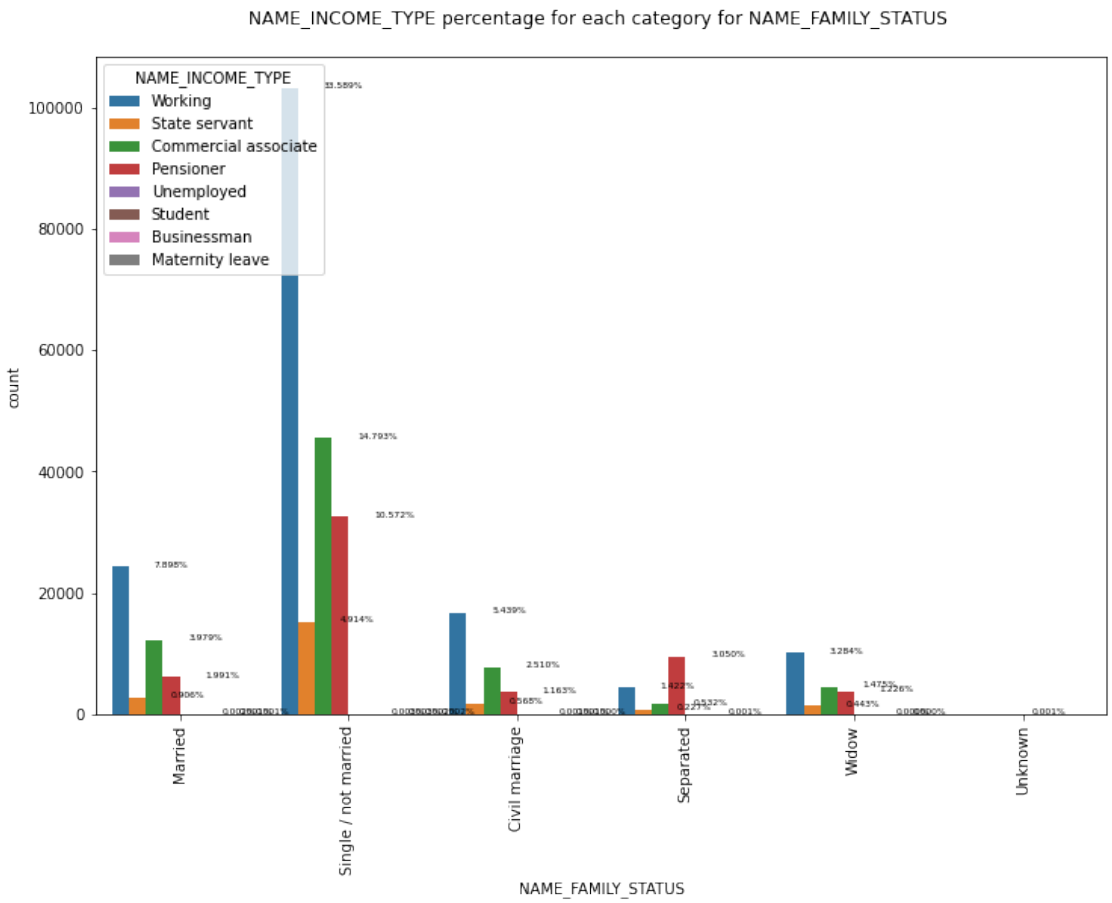
```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_FAMILY_STATUS',hue='CODE_GENDER',data=df_trai
n)
ax.set_xticklabels(df_train['NAME_FAMILY_STATUS'].value_counts().index
.to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("CODE_GENDER percentage for each category for
NAME_FAMILY_STATUS \n ")
```

```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
NAME_FAMILY_STATUS \n ')
```




```
#lets see its distribution with resepect to NAME_INCOME_TYPE variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_FAMILY_STATUS',hue='NAME_INCOME_TYPE',data=df
_train)
ax.set_xticklabels(df_train['NAME_FAMILY_STATUS'].value_counts().index
.to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),size=6)
ax.set_title("NAME_INCOME_TYPE percentage for each category for
NAME_FAMILY_STATUS \n ")
```

```
Text(0.5, 1.0, 'NAME_INCOME_TYPE percentage for each category for
NAME_FAMILY_STATUS \n ')
```




```
#mostly working people are single and then married
```

NAME_HOUSING_TYPE

What is the housing situation of the client (renting, living with parents, ...)

```
df_train['NAME_HOUSING_TYPE'].value_counts()
```

```
House / apartment      272868
With parents            14840
Municipal apartment     11183
Rented apartment         4881
Office apartment         2617
Co-op apartment          1122
Name: NAME_HOUSING_TYPE, dtype: int64
```

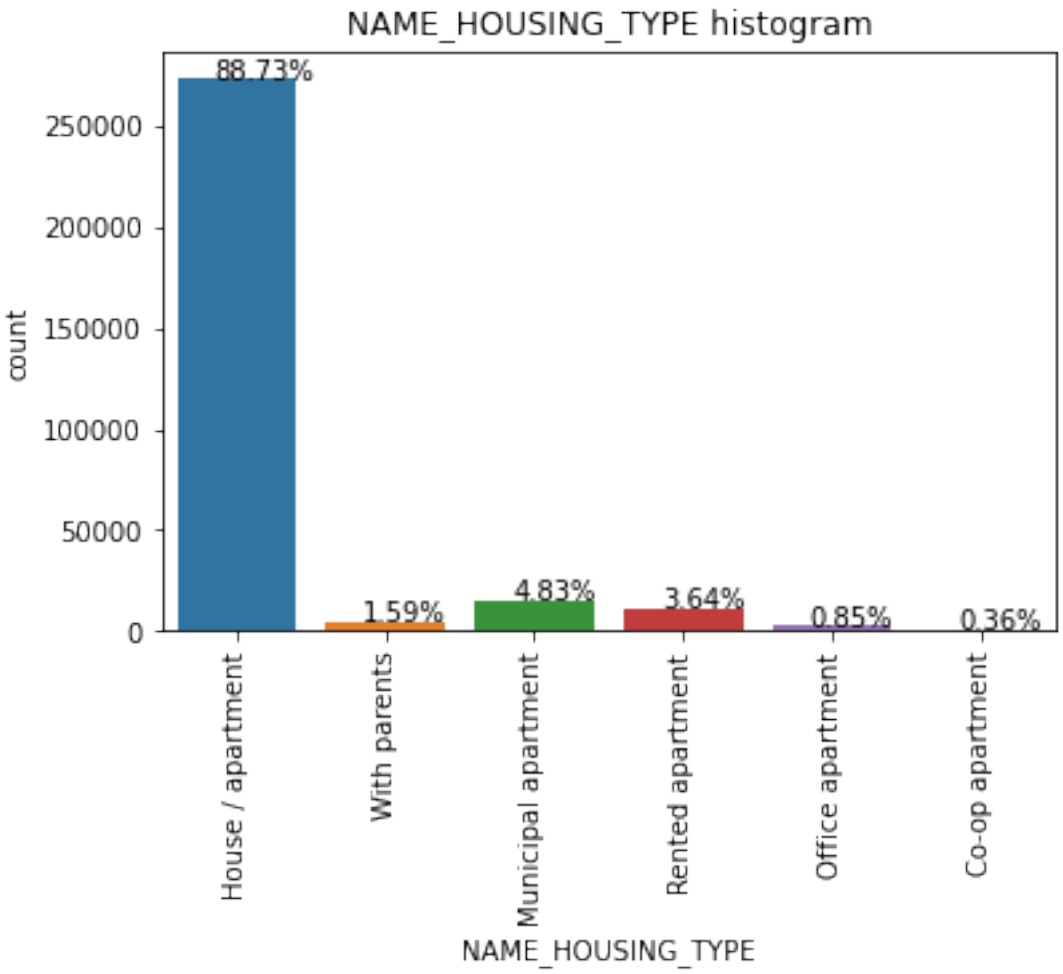
```
df_train['NAME_HOUSING_TYPE'].isna().sum()
#No NA values
```

```
0
```

```
#lets see its distribution
```

```
ax=sns.countplot(x='NAME_HOUSING_TYPE',data=df_train)
ax.set_xticklabels(df_train['NAME_HOUSING_TYPE'].value_counts().index.
to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("NAME_HOUSING_TYPE histogram")
```

```
Text(0.5, 1.0, 'NAME_HOUSING_TYPE histogram')
```



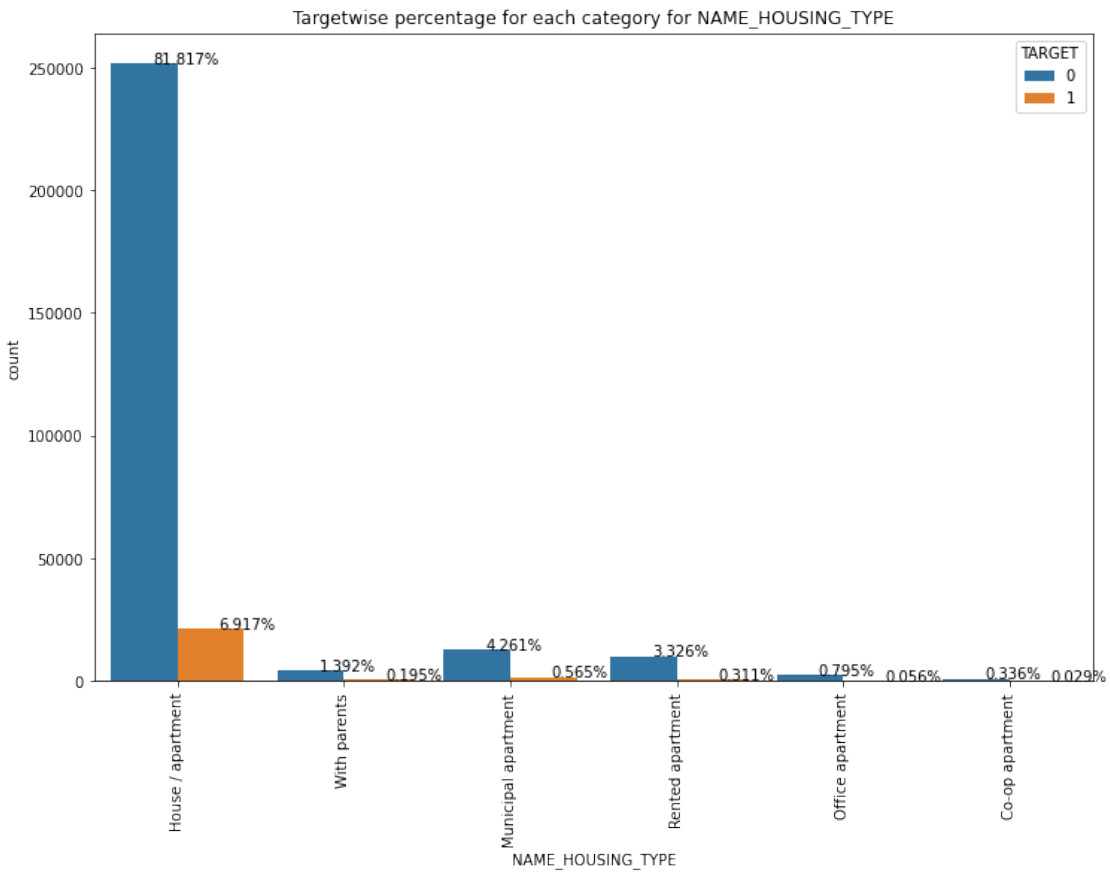
Most loan seekers are in descending orders as

- House/Apartment living people (maximum)
- Municiple Apartment

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_HOUSING_TYPE',hue='TARGET',data=df_train)
ax.set_xticklabels(df_train['NAME_HOUSING_TYPE'].value_counts().index.
to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
```

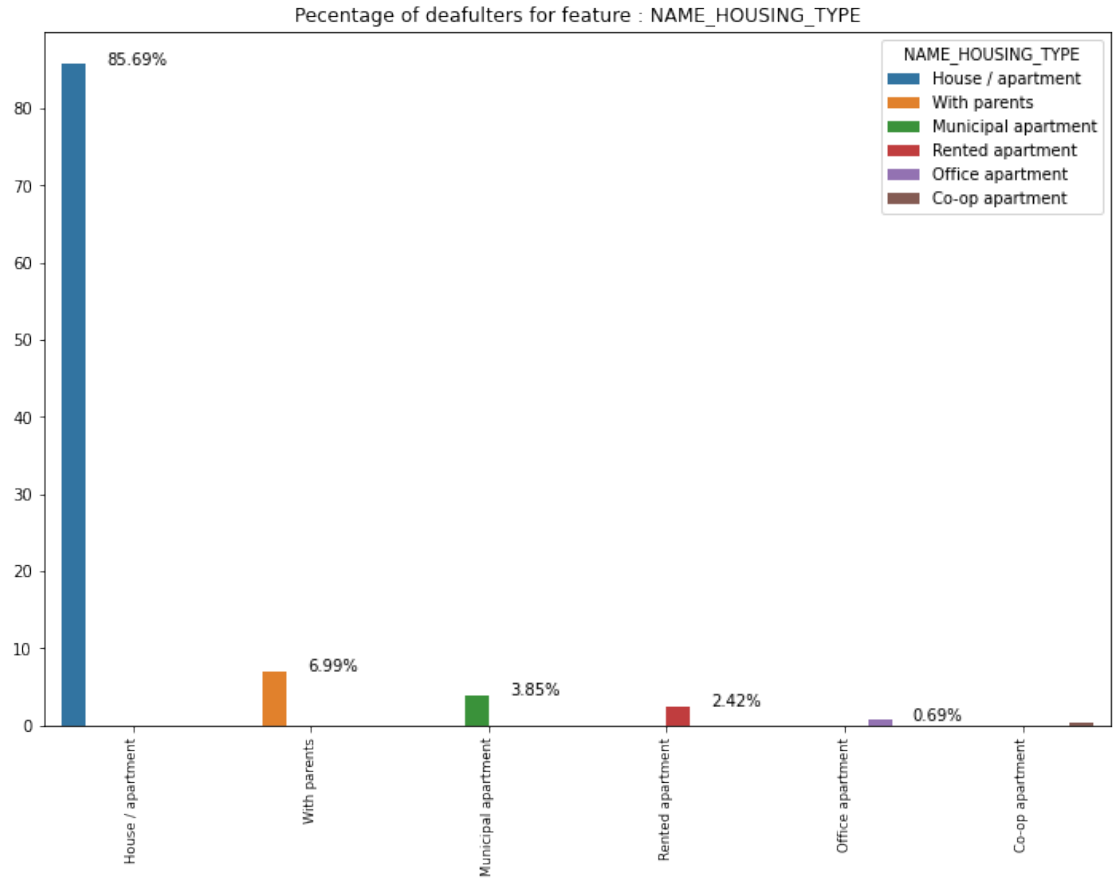
```
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Targetwise percentage for each category for
NAME_HOUSING_TYPE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NAME_HOUSING_TYPE')
```




```
plot_for_categorical_defaulters(df_train,'NAME_HOUSING_TYPE')
```

```
{'House / apartment': 85.68781470292045,
 'With parents': 6.992950654582074,
 'Municipal apartment': 3.8469284994964754,
 'Rented apartment': 2.420946626384693,
 'Office apartment': 0.6928499496475328,
 'Co-op apartment': 0.35850956696878145}
```

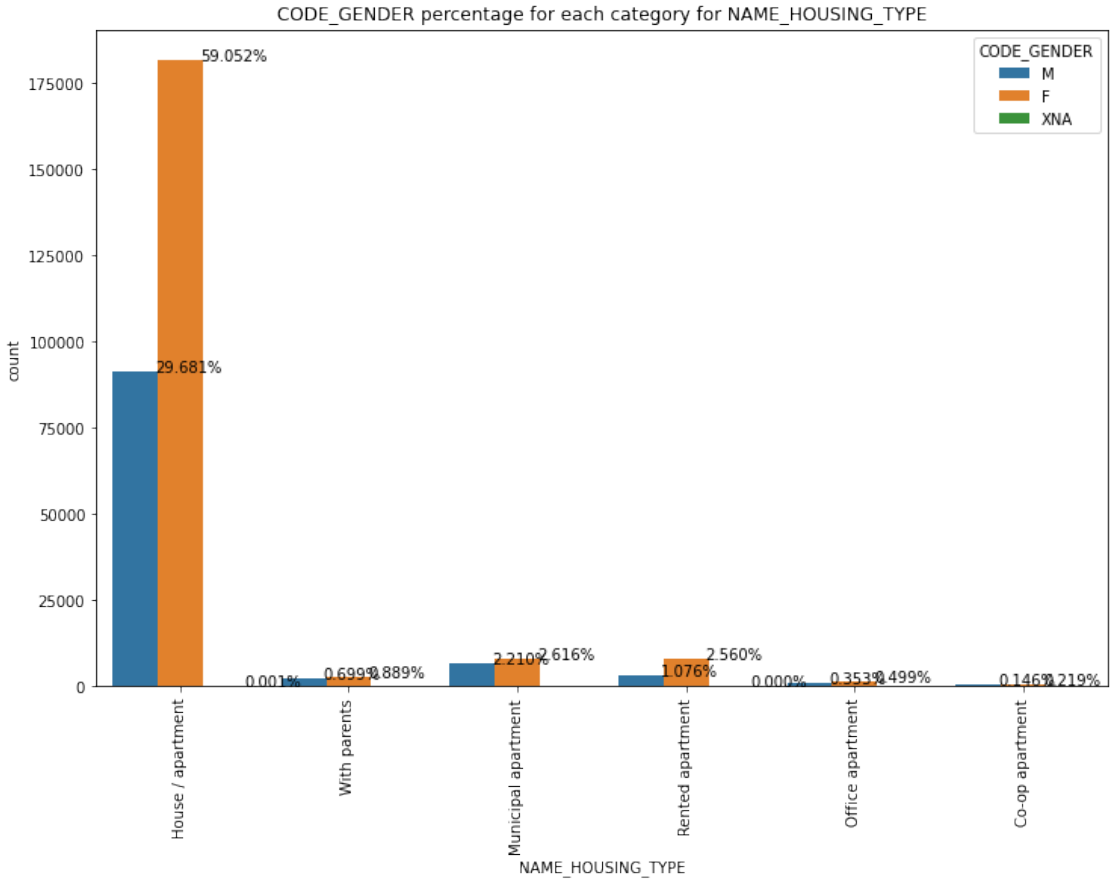


it seems people having Housing Type as in descending order

- House/Apartment
- Rented Aparment
- Municipal
- WIth parents (kind of people who dont own a home actuualy not even rented) are facing more problem in repayment of loan because those are labourer or lowe middle class people

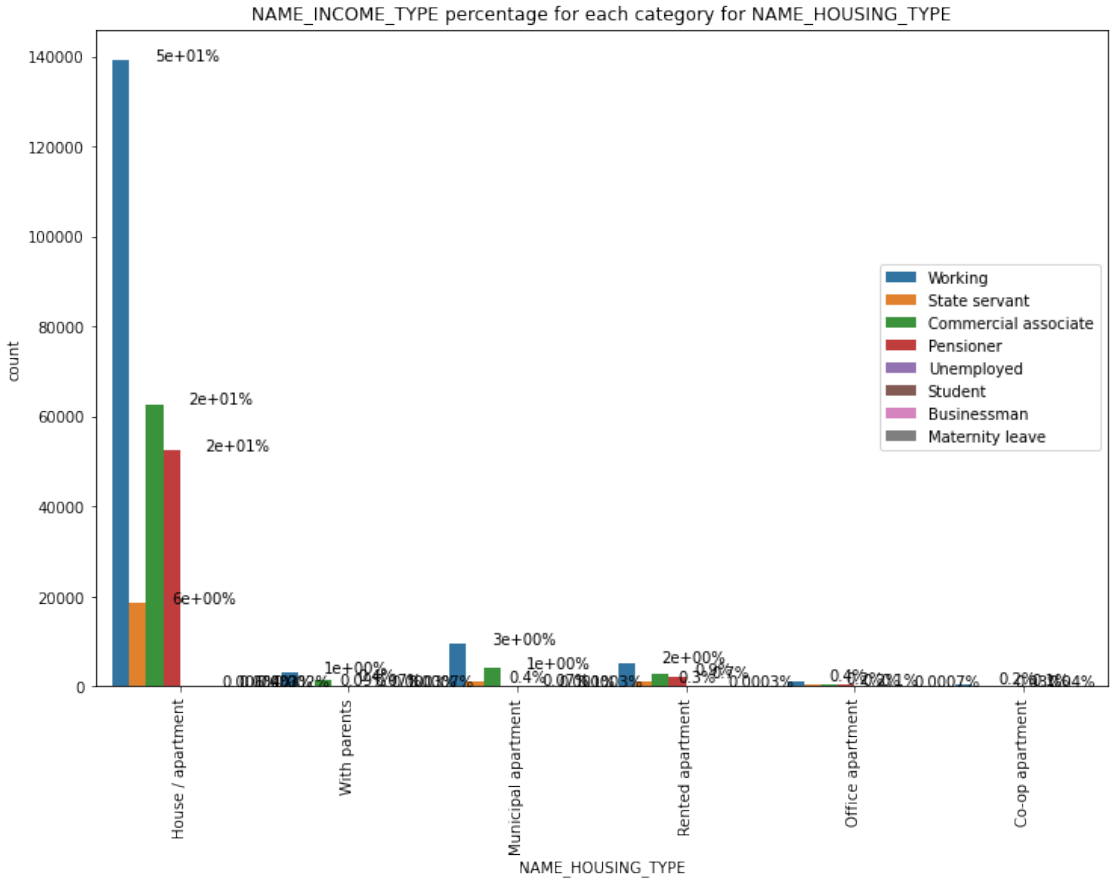
```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_HOUSING_TYPE',hue='CODE_GENDER',data=df_train
)
ax.set_xticklabels(df_train['NAME_HOUSING_TYPE'].value_counts().index.
to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("CODE_GENDER percentage for each category for
NAME_HOUSING_TYPE")
```

```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
NAME_HOUSING_TYPE')
```




```
#most women are living in House /Apartment

#lets see its distribution with resepect to NAME_INCOME_TYPE variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_HOUSING_TYPE',hue='NAME_INCOME_TYPE',data=df_
train)
ax.set_xticklabels(df_train['NAME_HOUSING_TYPE'].value_counts().index.
to_list(),rotation=90)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.1}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("NAME_INCOME_TYPE percentage for each category for
NAME_HOUSING_TYPE")
```

```
Text(0.5, 1.0, 'NAME_INCOME_TYPE percentage for each category for
NAME_HOUSING_TYPE')
```




```
#mostly people living in artments are working(all type)/Pensioner
```

OCCUPATION_TYPE

What kind of occupation does the client have

```
df_train['OCCUPATION_TYPE'].value_counts()
```

```
Laborers                 55186
Sales staff              32102
Core staff               27570
Managers                 21371
Drivers                  18603
High skill tech staff    11380
Accountants               9813
Medicine staff            8537
Security staff            6721
Cooking staff             5946
Cleaning staff            4653
Private service staff     2652
Low-skill Laborers        2093
Waiters/barmen staff      1348
Secretaries               1305
```

```
Realty agents              751
HR staff                   563
IT staff                   526
Name: OCCUPATION_TYPE, dtype: int64
```

```
df_train['OCCUPATION_TYPE'].isna().sum()
#have NA values
```

```
96391
```

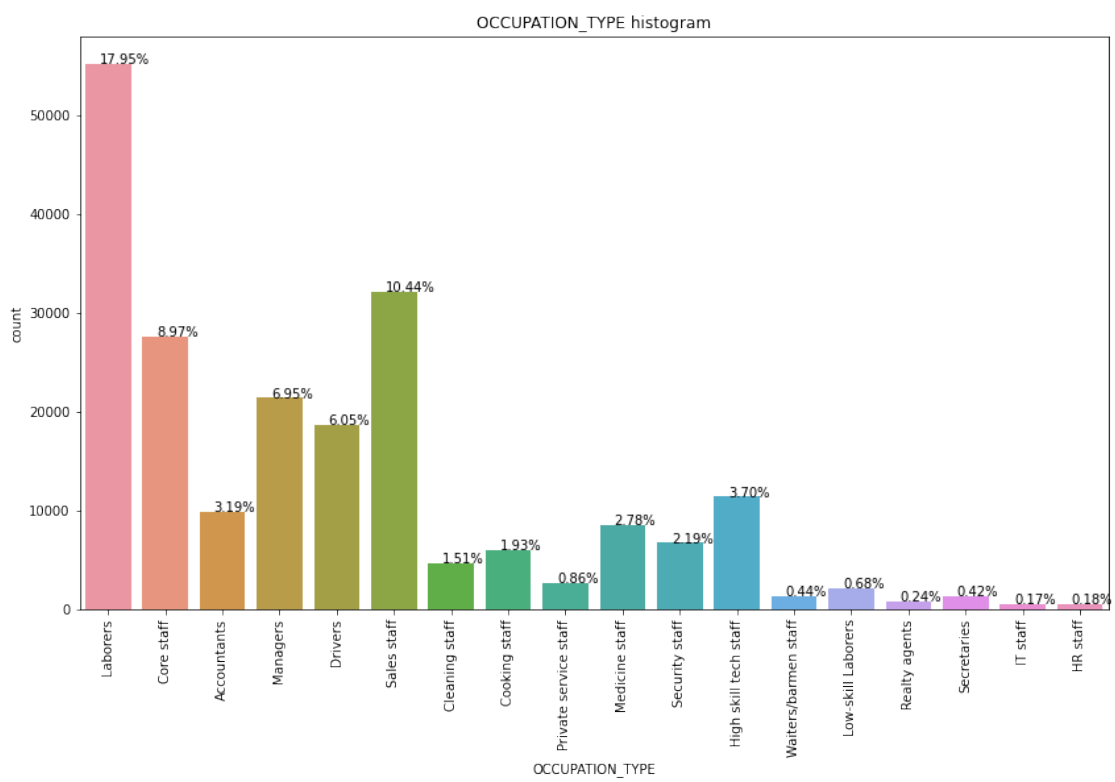
```
df_train['OCCUPATION_TYPE'].isna().sum()*100/df_train.shape[0]
```

```
31.345545362604916
```

- 32 % approx fields are blank actually

```
#lets see its distribution
plt.figure(figsize=(14,8))
ax=sns.countplot(x='OCCUPATION_TYPE',data=df_train)
#ax.set_xticklabels(df_train['OCCUPATION_TYPE'].value_counts().index.t
o_list(),rotation=90)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("OCCUPATION_TYPE histogram")
```

```
Text(0.5, 1.0, 'OCCUPATION_TYPE histogram')
```

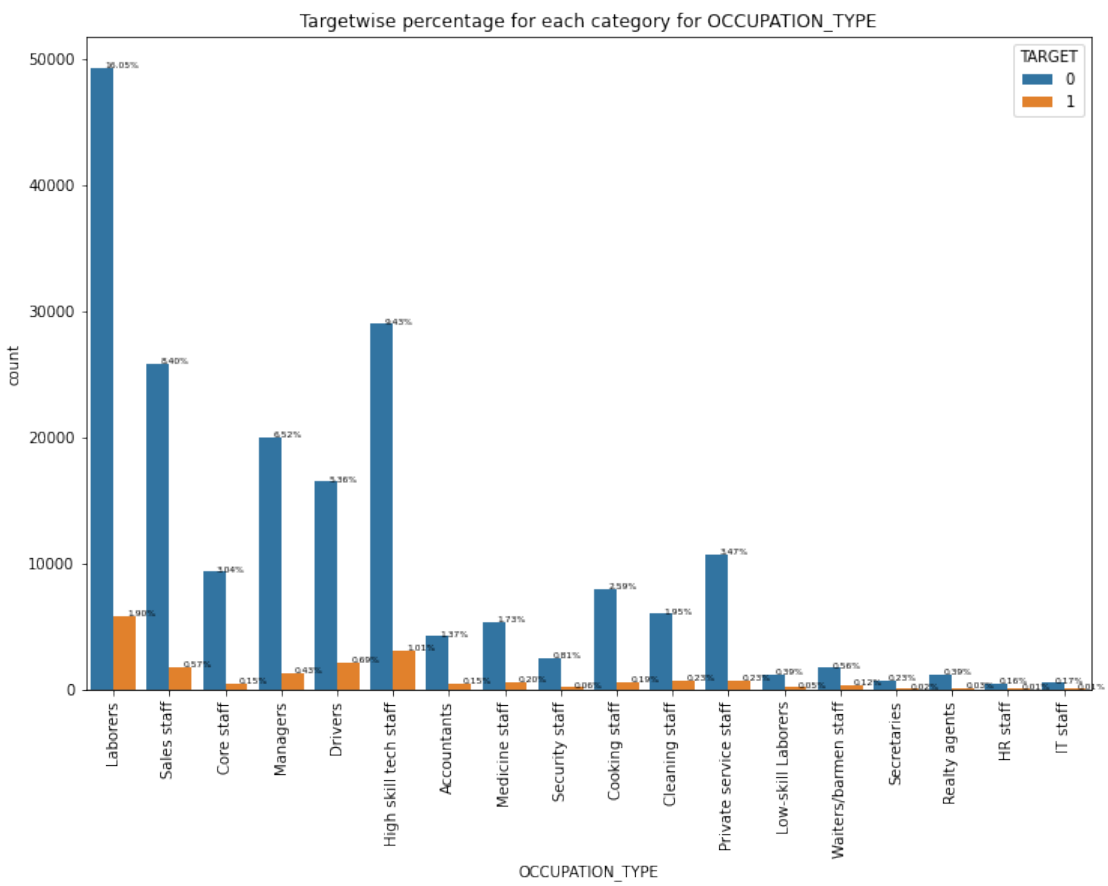


Most loan seekers are in descending orders as

- Laborers
- High skilled tech staffs
- Sales Staffs

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='OCCUPATION_TYPE',hue='TARGET',data=df_train)
ax.set_xticklabels(df_train['OCCUPATION_TYPE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),size=6)

ax.set_title("Targetwise percentage for each category for
OCCUPATION_TYPE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
OCCUPATION_TYPE')
```



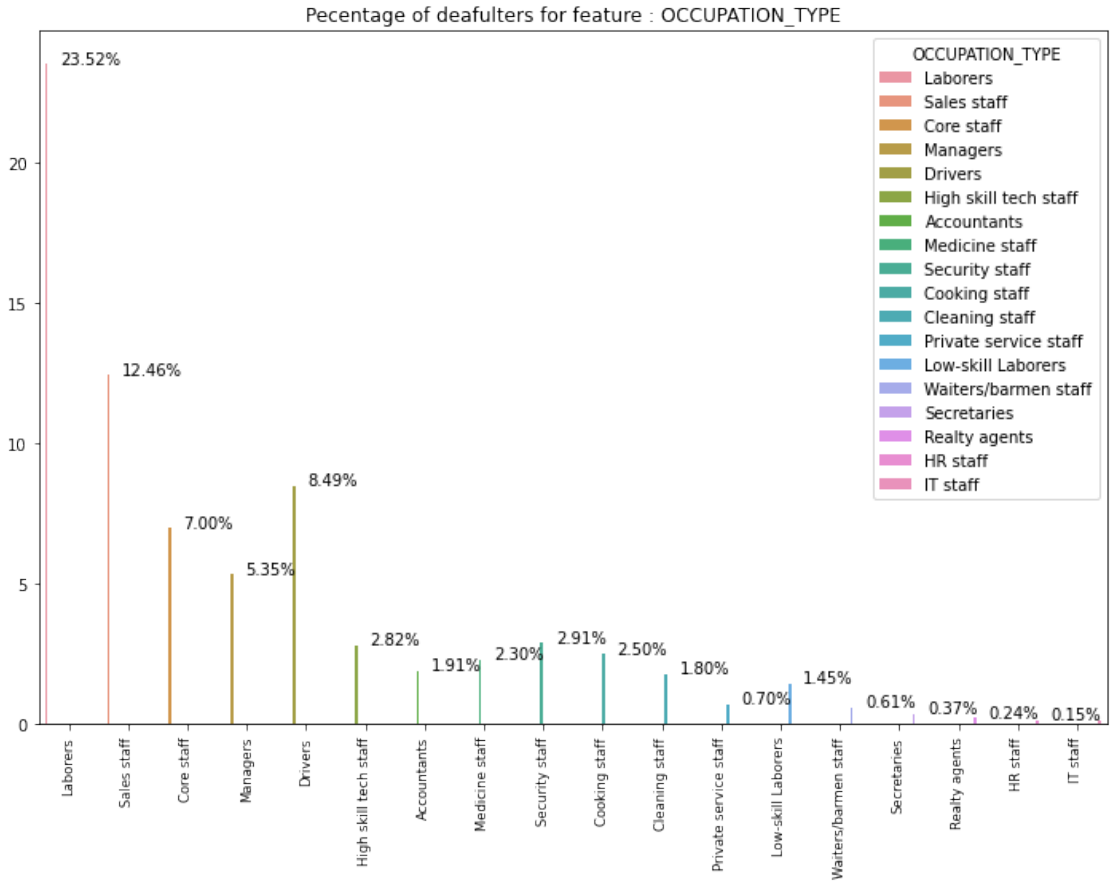

```
df_train['OCCUPATION_TYPE'].value_counts()
```

```
Laborers                 55186
Sales staff              32102
```

```
Core staff               27570
Managers                 21371
Drivers                  18603
High skill tech staff    11380
Accountants               9813
Medicine staff            8537
Security staff            6721
Cooking staff             5946
Cleaning staff            4653
Private service staff     2652
Low-skill Laborers        2093
Waiters/barmen staff      1348
Secretaries               1305
Realty agents              751
HR staff                   563
IT staff                   526
Name: OCCUPATION_TYPE, dtype: int64
```

```
plot_for_categorical_defaulters(df_train,'OCCUPATION_TYPE')
```

```
{'Laborers': 23.516616314199396,
 'Sales staff': 12.455186304128903,
 'Core staff': 7.001007049345418,
 'Managers': 5.34944612286002,
 'Drivers': 8.487411883182276,
 'High skill tech staff': 2.823766364551863,
 'Accountants': 1.9093655589123868,
 'Medicine staff': 2.3041289023162137,
 'Security staff': 2.908358509566969,
 'Cooking staff': 2.501510574018127,
 'Cleaning staff': 1.8006042296072509,
 'Private service staff': 0.7049345417925479,
 'Low-skill Laborers': 1.446122860020141,
 'Waiters/barmen staff': 0.6122860020140987,
 'Secretaries': 0.37059415911379656,
 'Realty agents': 0.23766364551863042,
 'HR staff': 0.14501510574018128,
 'IT staff': 0.13695871097683787}
```




```
x={'Laborers': 23.516616314199396,
 'Sales staff': 12.455186304128903,
 'Core staff': 7.001007049345418,
 'Managers': 5.34944612286002,
 'Drivers': 8.487411883182276,
 'High skill tech staff': 2.823766364551863,
 'Accountants': 1.9093655589123868,
 'Medicine staff': 2.3041289023162137,
 'Security staff': 2.908358509566969,
 'Cooking staff': 2.501510574018127,
 'Cleaning staff': 1.8006042296072509,
 'Private service staff': 0.7049345417925479,
 'Low-skill Laborers': 1.446122860020141,
 'Waiters/barmen staff': 0.6122860020140987,
 'Secretaries': 0.37059415911379656,
 'Realty agents': 0.23766364551863042,
 'HR staff': 0.14501510574018128,
 'IT staff': 0.13695871097683787}
```

```
{k: v for k, v in sorted(x.items(), key=lambda item: item[1])}
```

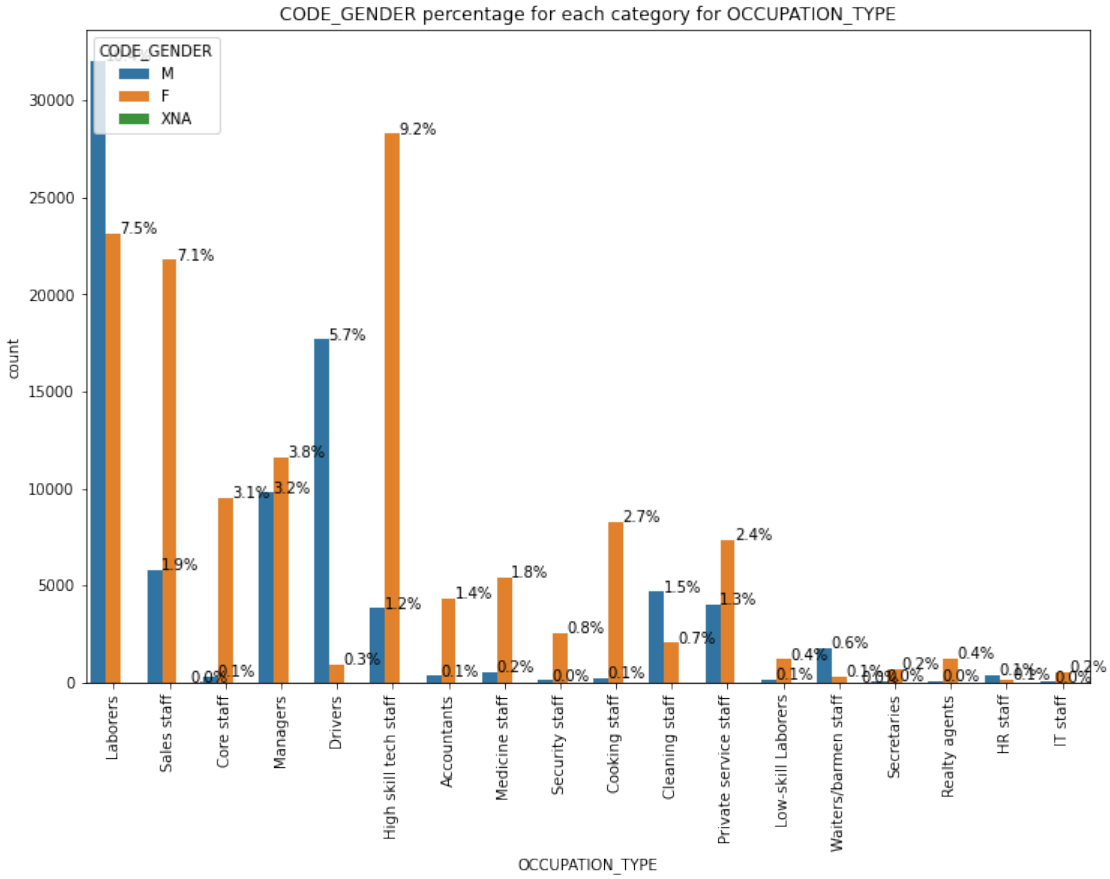
```
{'IT staff': 0.13695871097683787,
 'HR staff': 0.14501510574018128,
 'Realty agents': 0.23766364551863042,
 'Secretaries': 0.37059415911379656,
 'Waiters/barmen staff': 0.6122860020140987,
 'Private service staff': 0.7049345417925479,
 'Low-skill Laborers': 1.446122860020141,
 'Cleaning staff': 1.8006042296072509,
 'Accountants': 1.9093655589123868,
 'Medicine staff': 2.3041289023162137,
 'Cooking staff': 2.501510574018127,
 'High skill tech staff': 2.823766364551863,
 'Security staff': 2.908358509566969,
 'Managers': 5.34944612286002,
 'Core staff': 7.001007049345418,
 'Drivers': 8.487411883182276,
 'Sales staff': 12.455186304128903,
 'Laborers': 23.516616314199396}
```

it seems people having Occupation Type as in descending order

```
- Low skills laboureres/labourers
- Waiters/barmen staff,Drivers
- secuity staffs  are facing more problem in repayment of loan because
those are labourer or lower middle class people
```

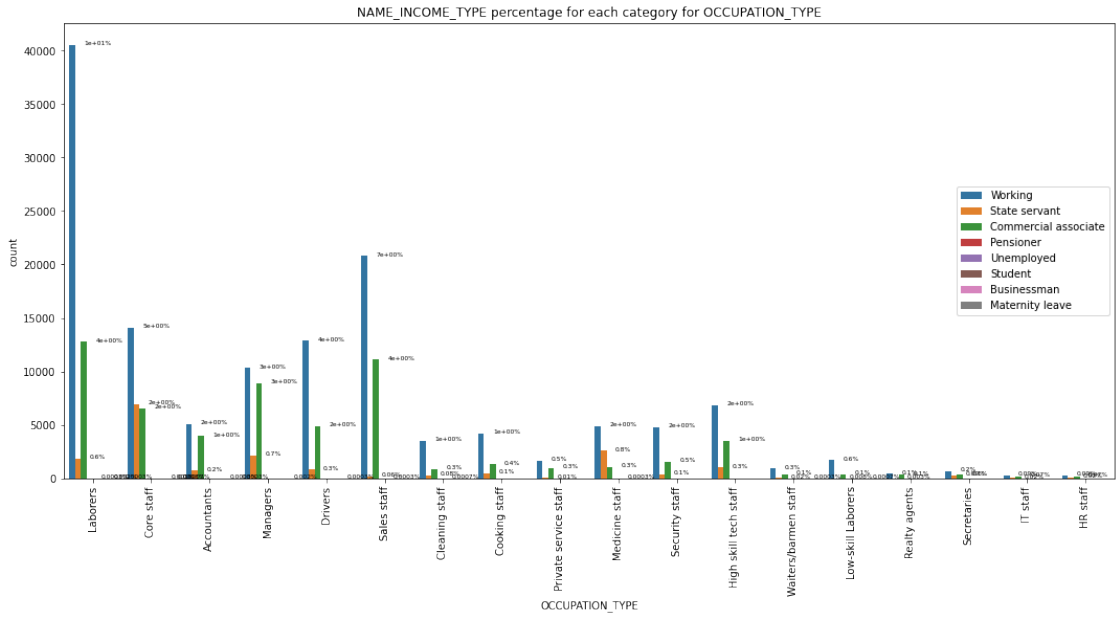
```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='OCCUPATION_TYPE',hue='CODE_GENDER',data=df_train)
ax.set_xticklabels(df_train['OCCUPATION_TYPE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("CODE_GENDER percentage for each category for
OCCUPATION_TYPE")
```

```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
OCCUPATION_TYPE')
```




```
#most women are living in High skills tech staffs and Labourers

#lets see its distribution with resepect to NAME_INCOME_TYPE variables
plt.figure(figsize=(18,8))
ax=sns.countplot(x='OCCUPATION_TYPE',hue='NAME_INCOME_TYPE',data=df_tr
ain)
plt.xticks(rotation=90)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.0}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),size=6)

ax.set_title("NAME_INCOME_TYPE percentage for each category for
OCCUPATION_TYPE")
```

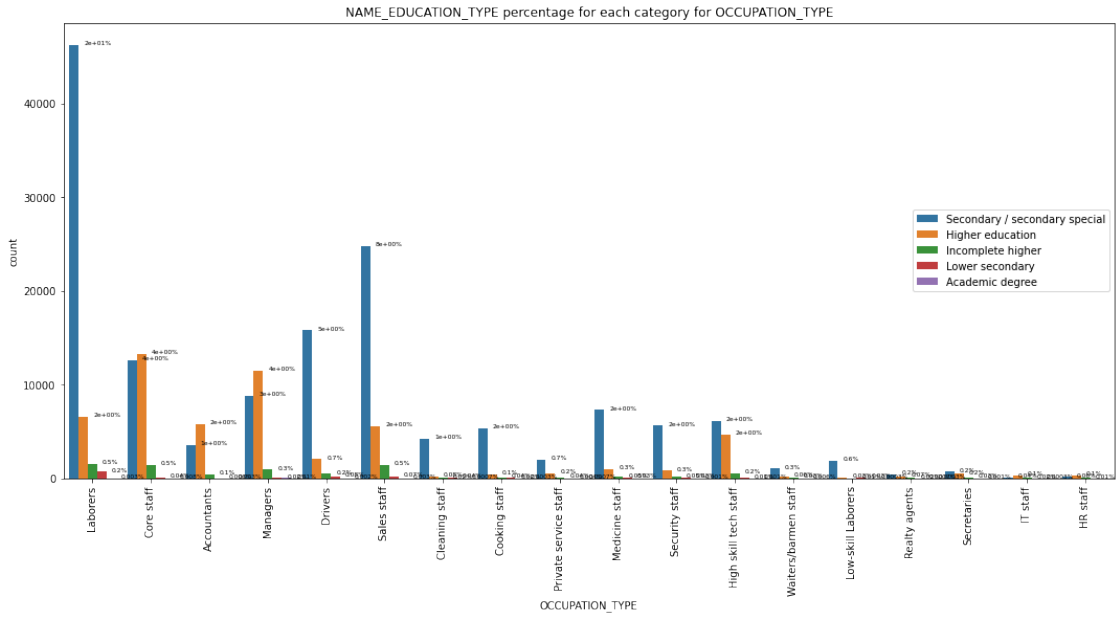
```
Text(0.5, 1.0, 'NAME_INCOME_TYPE percentage for each category for
OCCUPATION_TYPE')
```



- it shows that most of people whose NAME_INCOME_TYPE is working or commercial associate are belonging to working class
- we need to find a way to fill na values

```
#lets see its distribution with resepect to NAME_EDUCATION_TYPE variables
plt.figure(figsize=(18,8))
ax=sns.countplot(x='OCCUPATION_TYPE',hue='NAME_EDUCATION_TYPE',data=df_train)
plt.xticks(rotation=90)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.0}%'.format(p.get_height()*100/len(df_train)), (p.get_x()+0.25, p.get_height()+0.01),size=6)

ax.set_title("NAME_EDUCATION_TYPE percentage for each category for OCCUPATION_TYPE")
```

```
Text(0.5, 1.0, 'NAME_EDUCATION_TYPE percentage for each category for OCCUPATION_TYPE')
```



- it shows for most of working class eductation is secondary/secondary special

WEEKDAY_APPR_PROCESS_START

On which day of the week did the client apply for the loan

```
df_train['WEEKDAY_APPR_PROCESS_START'].value_counts()
```

```
TUESDAY      53901
WEDNESDAY    51934
MONDAY       50714
THURSDAY     50591
FRIDAY       50338
SATURDAY     33852
SUNDAY       16181
Name: WEEKDAY_APPR_PROCESS_START, dtype: int64
```

```
df_train['WEEKDAY_APPR_PROCESS_START'].isna().sum()
#No NA values
```

```
0
```

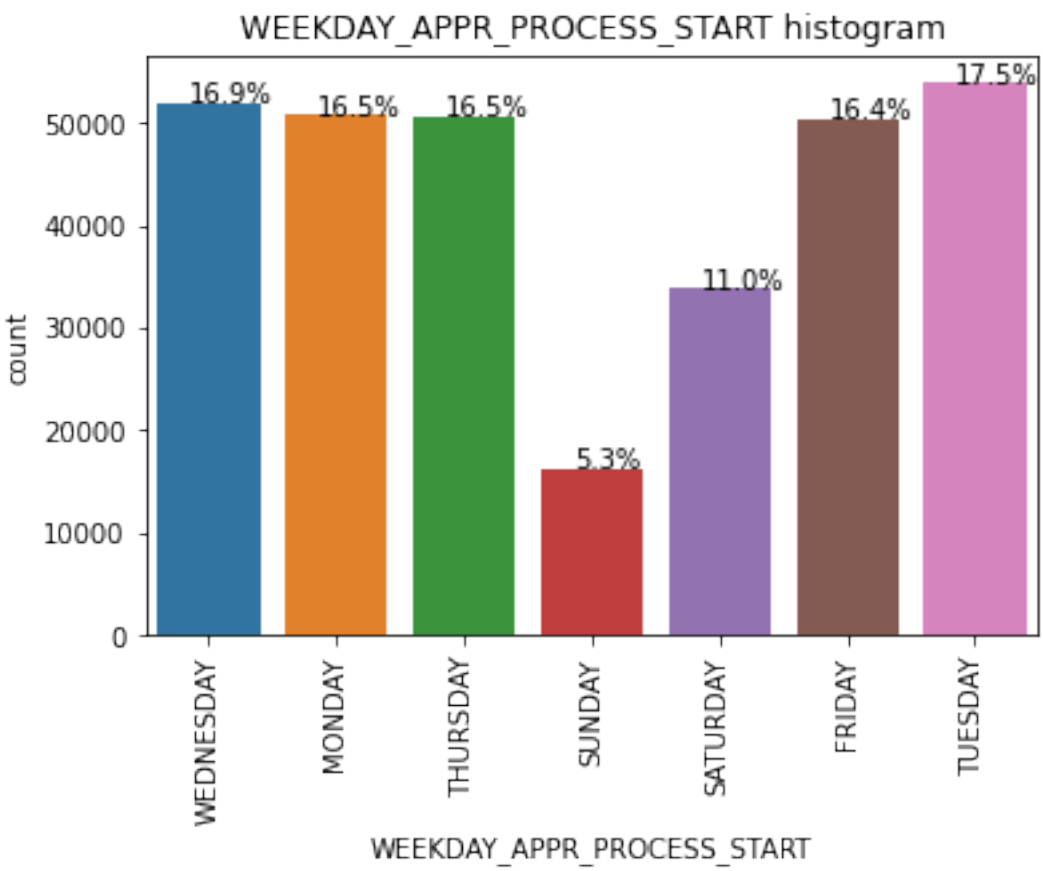
```
df_train['WEEKDAY_APPR_PROCESS_START'].value_counts()
```

```
TUESDAY      53901
WEDNESDAY    51934
MONDAY       50714
THURSDAY     50591
FRIDAY       50338
SATURDAY     33852
```

```
SUNDAY       16181
Name: WEEKDAY_APPR_PROCESS_START, dtype: int64
```

```
#lets see its distribution

ax=sns.countplot(x='WEEKDAY_APPR_PROCESS_START',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("WEEKDAY_APPR_PROCESS_START histogram")
```

```
Text(0.5, 1.0, 'WEEKDAY_APPR_PROCESS_START histogram')
```

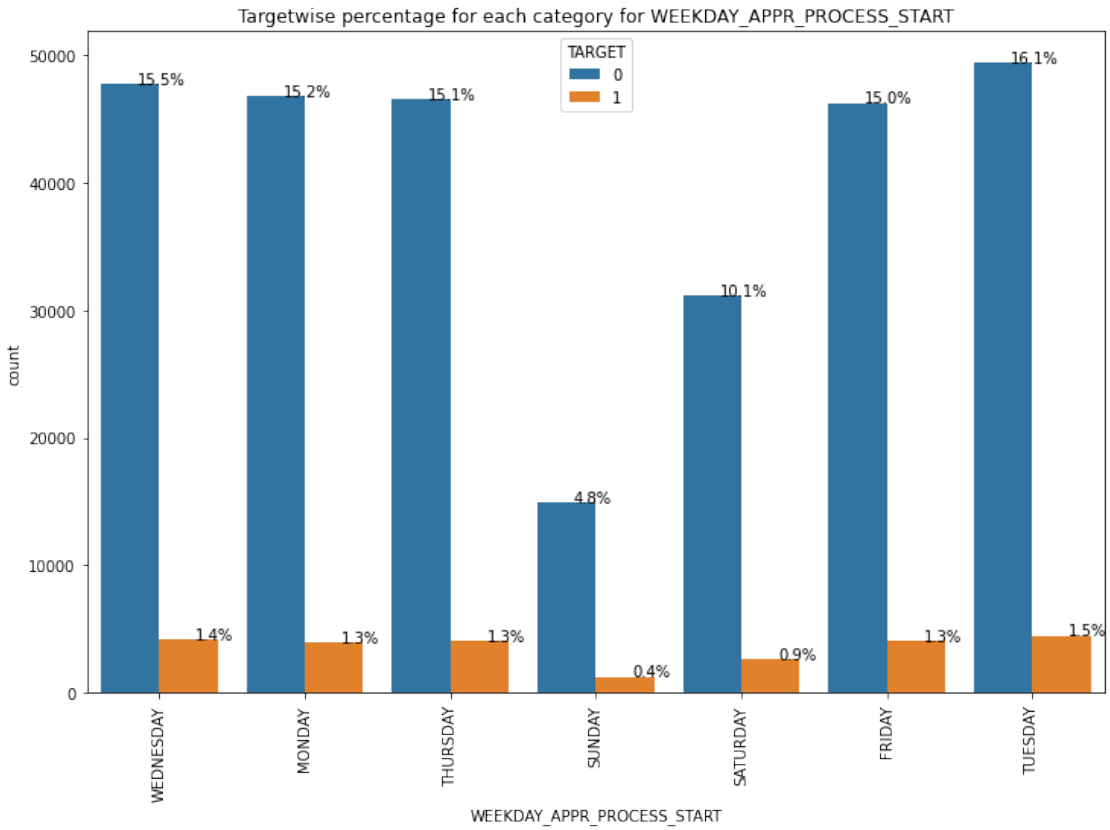


Most loan seekers have appied for loan weekdays

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='WEEKDAY_APPR_PROCESS_START',hue='TARGET',data=df_t
rain)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
```

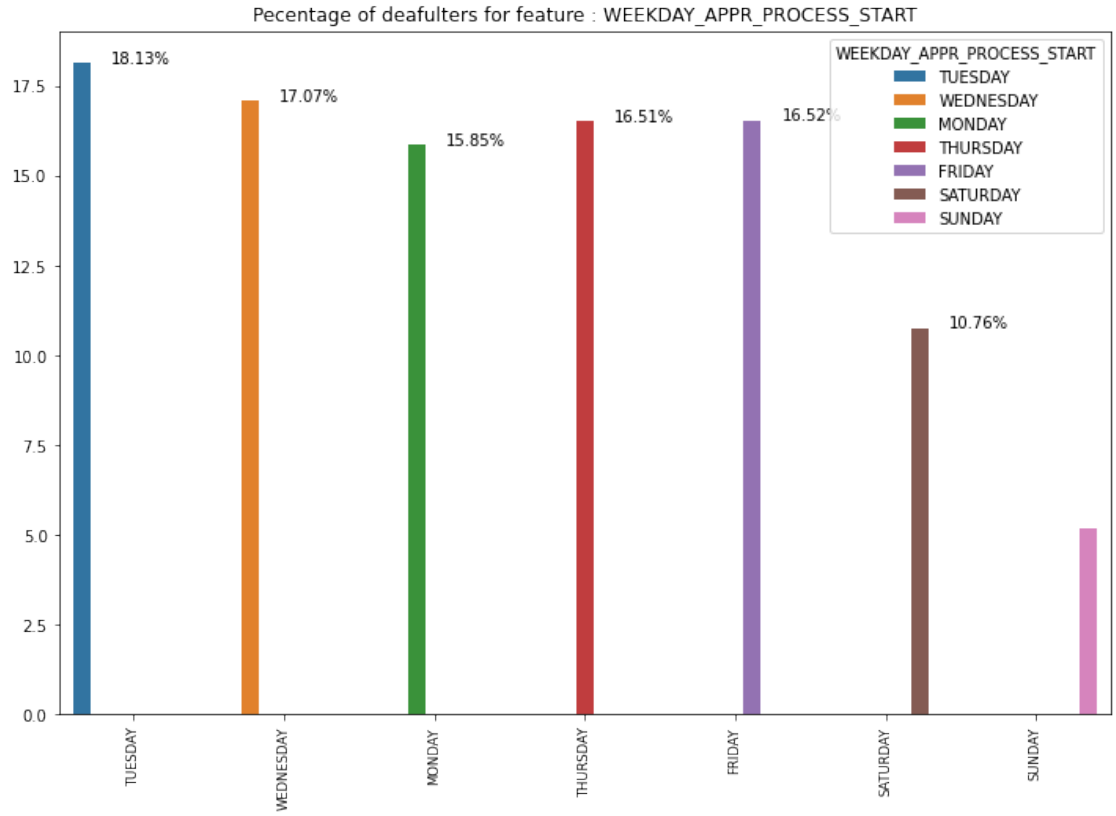
```
ax.set_title("Targetwise percentage for each category for
WEEKDAY_APPR_PROCESS_START")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
WEEKDAY_APPR_PROCESS_START')
```




```
plot_for_categorical_defaulters(df_train,'WEEKDAY_APPR_PROCESS_START')
```

```
{'TUESDAY': 18.13091641490433,
 'WEDNESDAY': 17.07150050352467,
 'MONDAY': 15.846928499496475,
 'THURSDAY': 16.507552870090635,
 'FRIDAY': 16.51963746223565,
 'SATURDAY': 10.755287009063444,
 'SUNDAY': 5.168177240684794}
```



it seems loan appied day doesnt matter for how is the loan repayments status

ORGANIZATION_TYPE

Type of organization where client works

```
df_train['ORGANIZATION_TYPE'].value_counts()
```

```
Business Entity Type 3    67992
XNA                       55374
Self-employed             38412
Other                     16683
Medicine                  11193
Business Entity Type 2    10553
Government                10404
School                     8893
Trade: type 7              7831
Kindergarten               6880
Construction               6721
Business Entity Type 1     5984
Transport: type 4          5398
Trade: type 3              3492
Industry: type 9           3368
Industry: type 3           3278
```

```
Security                    3247
Housing                     2958
Industry: type 11           2704
Military                    2634
Bank                        2507
Agriculture                 2454
Police                      2341
Transport: type 2           2204
Postal                      2157
Security Ministries         1974
Trade: type 2               1900
Restaurant                  1811
Services                    1575
University                  1327
Industry: type 7            1307
Transport: type 3           1187
Industry: type 1            1039
Hotel                        966
Electricity                  950
Industry: type 4             877
Trade: type 6                631
Industry: type 5             599
Insurance                    597
Telecom                      577
Emergency                    560
Industry: type 2             458
Advertising                  429
Realtor                      396
Culture                      379
Industry: type 12            369
Trade: type 1                348
Mobile                       317
Legal Services               305
Cleaning                     260
Transport: type 1            201
Industry: type 6             112
Industry: type 10            109
Religion                      85
Industry: type 13             67
Trade: type 4                 64
Trade: type 5                 49
Industry: type 8              24
Name: ORGANIZATION_TYPE, dtype: int64
```

```
len(df_train['ORGANIZATION_TYPE'].value_counts())
```

```
58
```

```
len(df_test['ORGANIZATION_TYPE'].value_counts())#Looks good
```

```
58
```

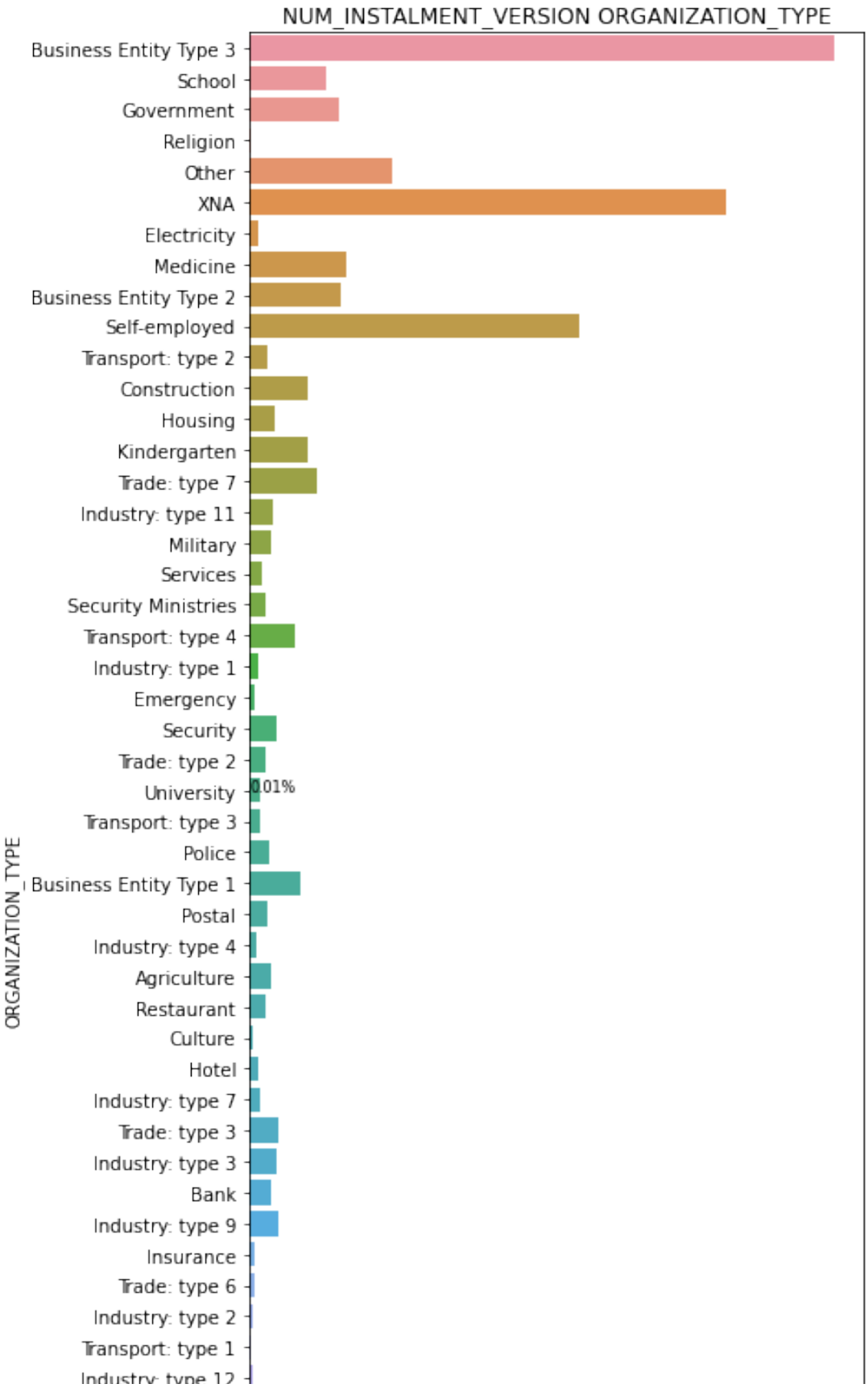
- So there are actually 58 available categories in train data

```
df_train['ORGANIZATION_TYPE'].isna().sum()
#No NA values
```

```
0
```

```
#lets see its distribution
plt.figure(figsize=(6,18))
ax=sns.countplot(y='ORGANIZATION_TYPE',data=df_train)
#plt.yticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_width()*100/len(df_train)),
(p.get_x()+0.25, p.get_width()+0.01),size=8)

ax.set_title("NUM_INSTALMENT_VERSION ORGANIZATION_TYPE")
```

```
Text(0.5, 1.0, 'NUM_INSTALMENT_VERSION ORGANIZATION_TYPE')
```

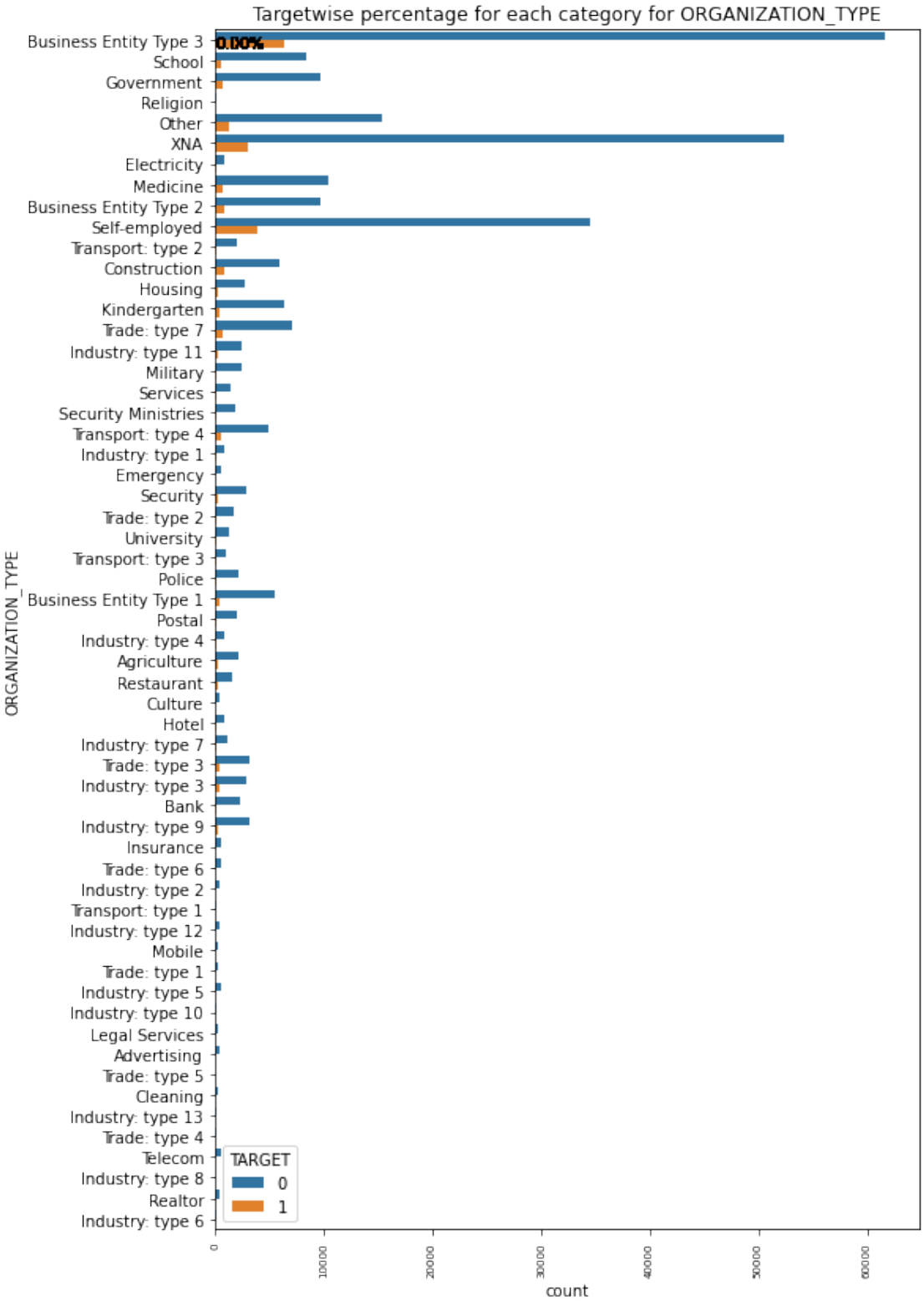

NUM_INSTALMENT_VERSION ORGANIZATION_TYPE
ORGANIZATION_TYPE
Business Entity Type 3
School
Government
Religion
Other
XNA
Electricity
Medicine
Business Entity Type 2
Self-employed
Transport: type 2
Construction
Housing
Kindergarten
Trade: type 7
Industry: type 11
Military
Services
Security Ministries
Transport: type 4
Industry: type 1
Emergency
Security
Trade: type 2
University
0.01%
Transport: type 3
Police
Business Entity Type 1
Postal
Industry: type 4
Agriculture
Restaurant
Culture
Hotel
Industry: type 7
Trade: type 3
Industry: type 3
Bank
Industry: type 9
Insurance
Trade: type 6
Industry: type 2
Transport: type 1
Industry: type 12

Most loan seekers are in descending orders as

- Business Entity Type 3 : 67992
- XNA : 55374
- Self-employed : 38412
- Other : 16683
- Medicine : 11193
- Business Entity Type 2 : 10553
- Government : 10404

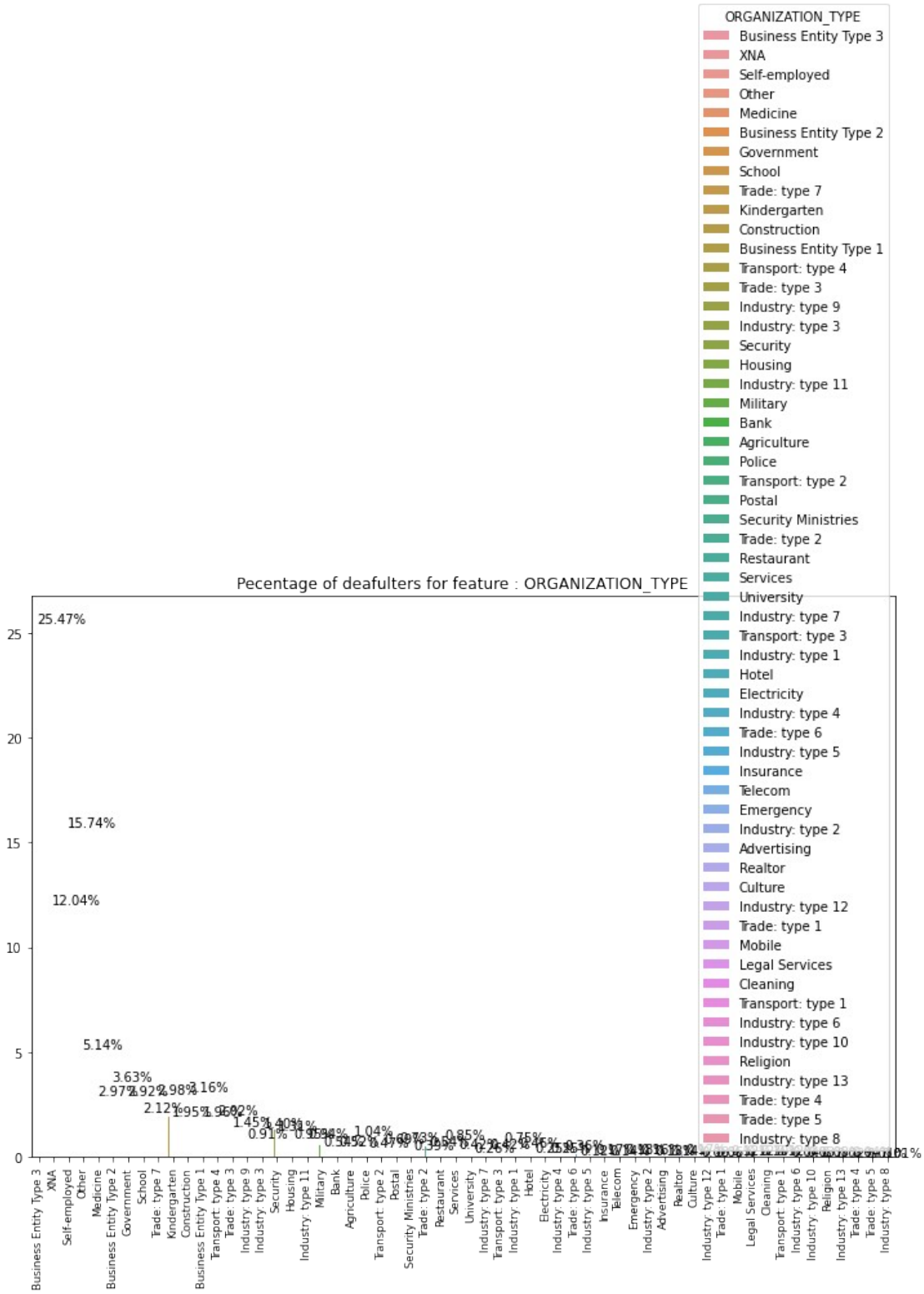
```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(8,14))
ax=sns.countplot(y='ORGANIZATION_TYPE',hue='TARGET',data=df_train)
plt.xticks(rotation=90,fontsize='x-small')
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
ORGANIZATION_TYPE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
ORGANIZATION_TYPE')
```




```
plt.figure(figsize=(8,12))
plot_for_categorical_defaulters(df_train,'ORGANIZATION_TYPE')
```

```
<Figure size 576x864 with 0 Axes>
```



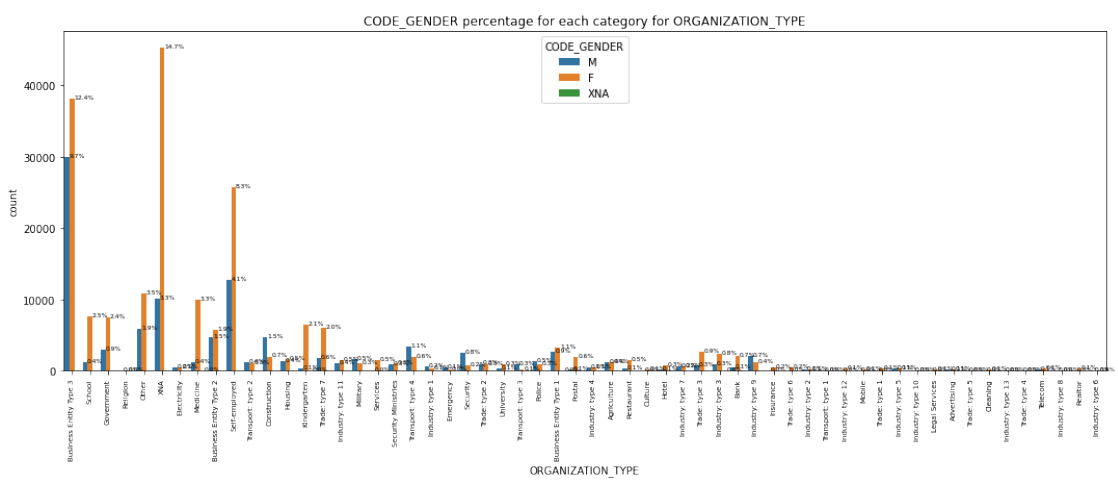
it seems people having ORGANIZATION_TYPE as in descending order

- Transport type 3
- Industry type 13

- Industry type 8 facing more problem in repayment of loan because those are labourer or lower middle class people

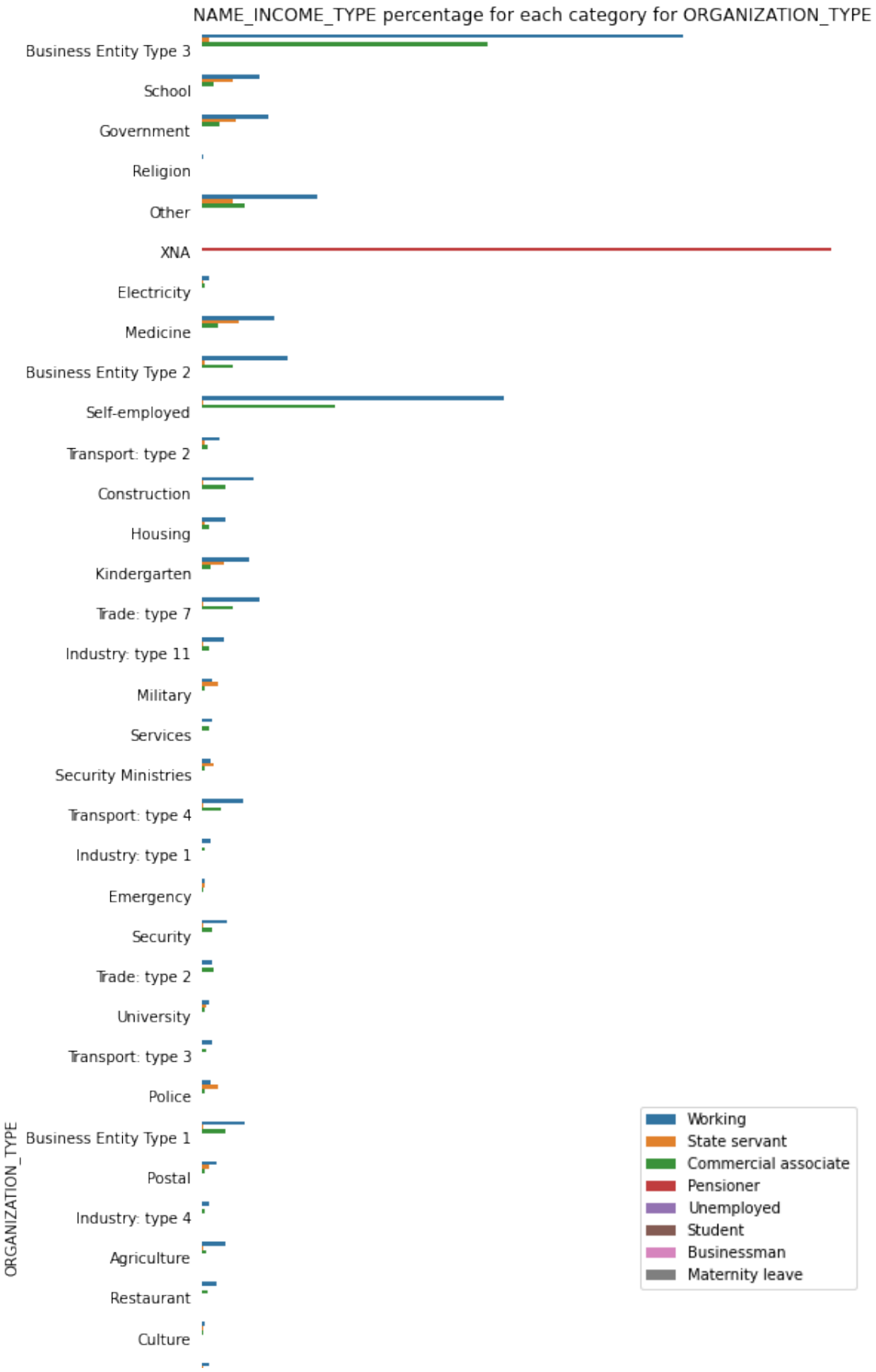
```
map_org_type={k: v for k, v in sorted(x.items(), key=lambda item: item[1])}

#map_org_type

map_org_type_encode={}
for i,j in map_org_type.items():
    if j >= 10:
        map_org_type_encode[i]=2
    elif j<10 and j>=1:
        map_org_type_encode[i]=1
    else:
        map_org_type_encode[i]=0


#map_org_type_encode

#lets see its distribution with resepect to gender variables
plt.figure(figsize=(18,6))
ax=sns.countplot(x='ORGANIZATION_TYPE',hue='CODE_GENDER',data=df_train
)
plt.xticks(rotation=90,fontsize='x-small')
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),size=6)

ax.set_title("CODE_GENDER percentage for each category for
ORGANIZATION_TYPE")
```

```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
ORGANIZATION_TYPE')
```



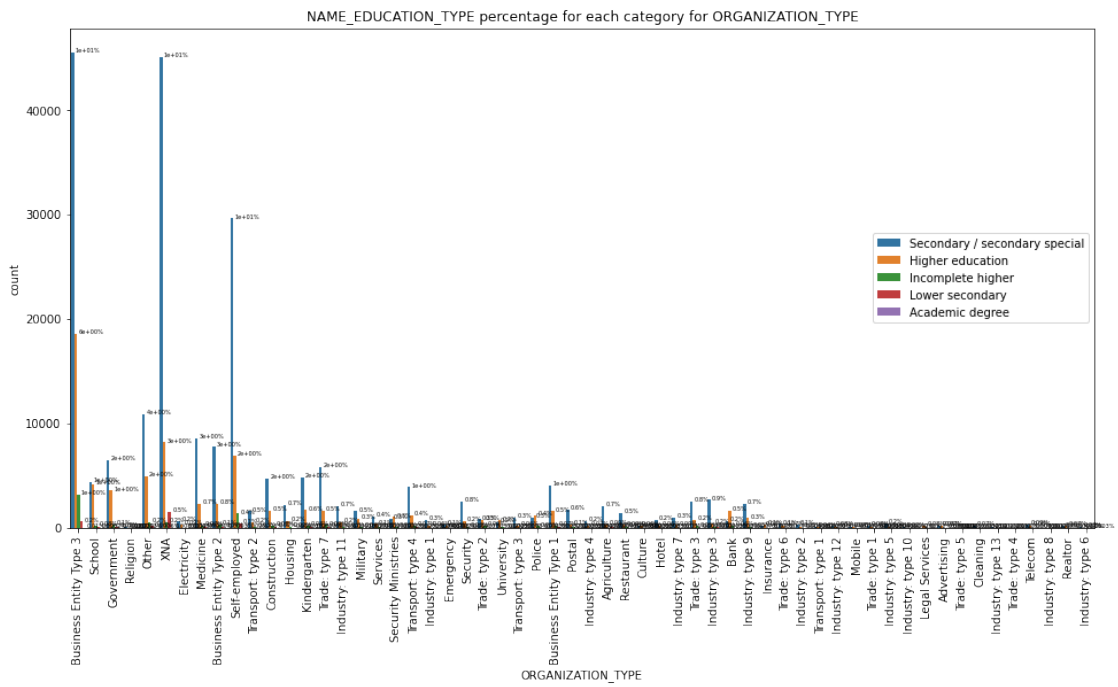
- most women are non working,and who are working, they mostly working in occupation type Business entity type 3 and self employed

```
#lets see its distribution with resepect to NAME_INCOME_TYPE variables
plt.figure(figsize=(8,29))
ax=sns.countplot(y='ORGANIZATION_TYPE',hue='NAME_INCOME_TYPE',data=df_
train)
plt.xticks(rotation=90)
ax.legend(loc='right')

ax.set_title("NAME_INCOME_TYPE percentage for each category for
ORGANIZATION_TYPE")
# for p in ax.spines.values:
#     ax.annotate('{:.0}%'.format(p.get_width()*100/len(df_train)),
(p.get_y()+0.25, p.get_width()+0.01))
for spine in ax.spines.values():
    spine.set_visible(False)

ax.axes.get_xaxis().set_visible(True)
ax.tick_params(axis="y", left=False)
plt.show()
```


NAME_INCOME_TYPE percentage for each category for ORGANIZATION_TYPE
ORGANIZATION_TYPE
Business Entity Type 3
School
Government
Religion
Other
XNA
Electricity
Medicine
Business Entity Type 2
Self-employed
Transport: type 2
Construction
Housing
Kindergarten
Trade: type 7
Industry: type 11
Military
Services
Security Ministries
Transport: type 4
Industry: type 1
Emergency
Security
Trade: type 2
University
Transport: type 3
Police
Business Entity Type 1
Postal
Industry: type 4
Agriculture
Restaurant
Culture
Working
State servant
Commercial associate
Pensioner
Unemployed
Student
Businessman
Maternity leave

```
#it shows that most of people whose NAME_INCOME_TYPE is working or
commercial associate are belonging to working class
# we need to find a way to fill na values

#lets see its distribution with resepect to NAME_EDUCATION_TYPE
variables
plt.figure(figsize=(16,8))
ax=sns.countplot(x='ORGANIZATION_TYPE',hue='NAME_EDUCATION_TYPE',data=
df_train)
plt.xticks(rotation=90,)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.0}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=5)

ax.set_title("NAME_EDUCATION_TYPE percentage for each category for
ORGANIZATION_TYPE")
```

```
Text(0.5, 1.0, 'NAME_EDUCATION_TYPE percentage for each category for
ORGANIZATION_TYPE')
```



- In most categories of Organiazation type people having Secondy class eductaion are dominating

FONDKAPREMONT_MODE

Normalized information about building where the client lives,Mode

```
df_train['FONDKAPREMONT_MODE'].value_counts()
```

```
reg oper account         73830
reg oper spec account    12080
not specified             5687
org spec account          5619
Name: FONDKAPREMONT_MODE, dtype: int64
```

```
df_train['FONDKAPREMONT_MODE'].isna().sum()
#Have NA values
```

```
210295
```

```
df_train['FONDKAPREMONT_MODE'].isna().sum()*100/df_train.shape[0]
```

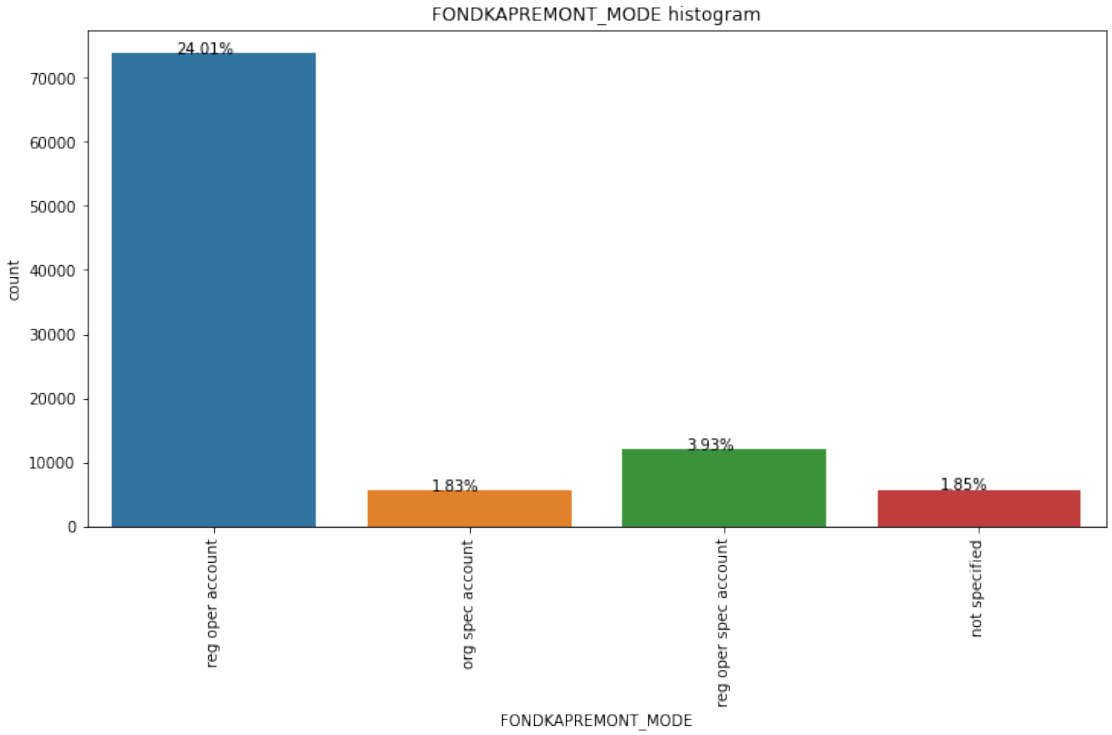
```
68.38617155158677
```

- 68 % approx fields are NAN actually

```
#lets see its distribution
plt.figure(figsize=(12,6))
ax=sns.countplot(x='FONDKAPREMONT_MODE',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
```

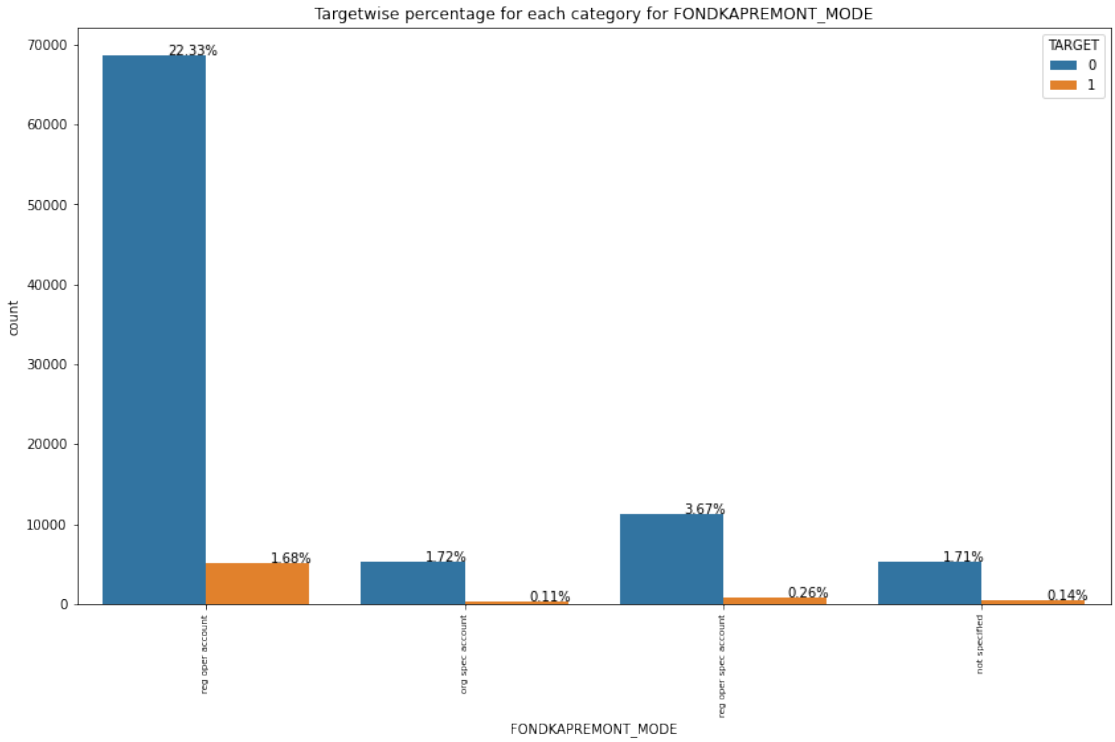
```
ax.set_title("FONDKAPREMONT_MODE histogram")
```

```
Text(0.5, 1.0, 'FONDKAPREMONT_MODE histogram')
```

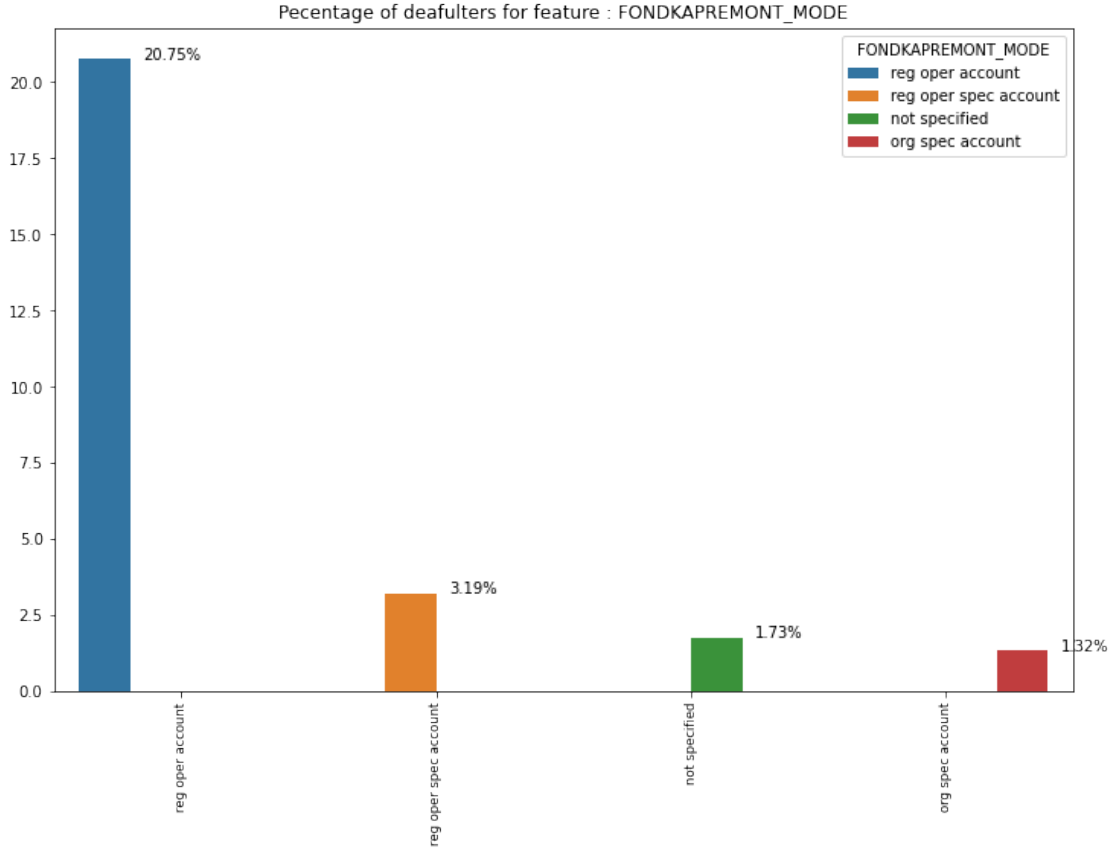



```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(14,8))
ax=sns.countplot(x='FONDKAPREMONT_MODE',hue='TARGET',data=df_train)
plt.xticks(rotation=90,fontsize='x-small')
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
FONDKAPREMONT_MODE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FONDKAPREMONT_MODE')
```




```
x=plot_for_categorical_defaulters(df_train,'FONDKAPREMONT_MODE')
```



```
x
```

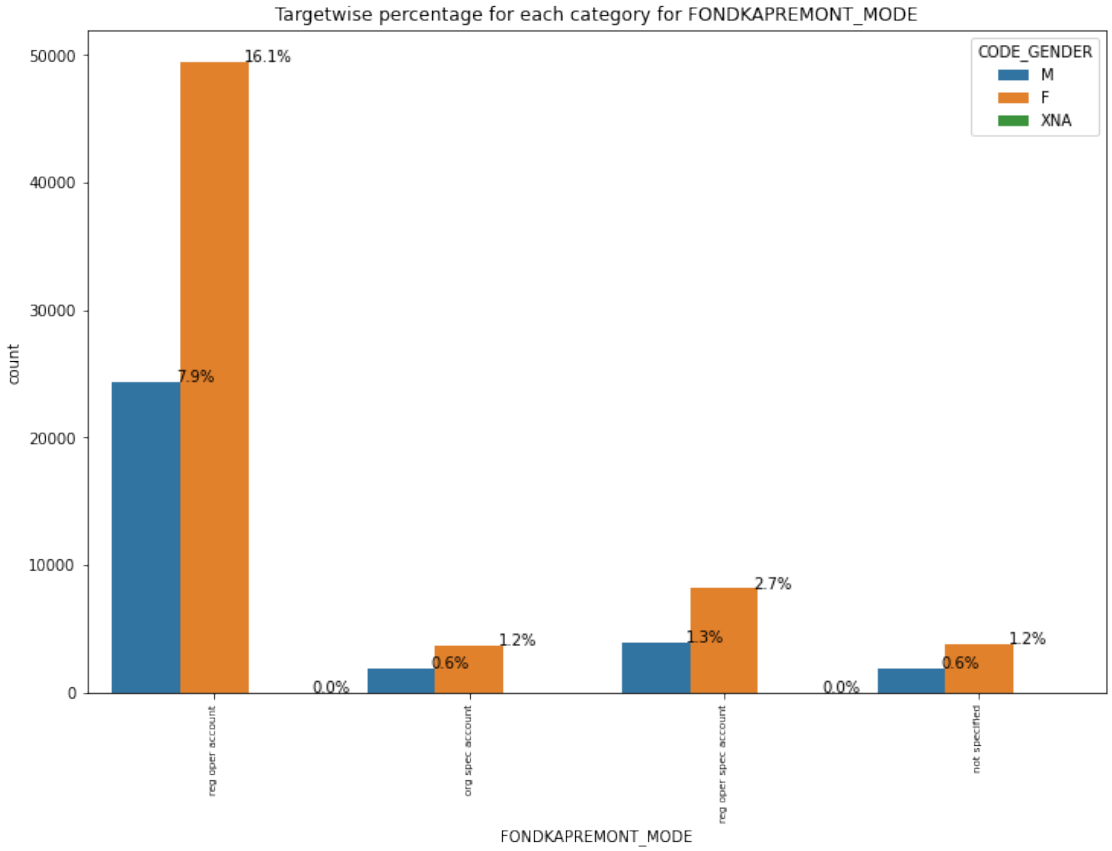
```
{'reg oper account': 20.75327291037261,
 'reg oper spec account': 3.190332326283988,
 'not specified': 1.7280966767371602,
 'org spec account': 1.3172205438066464}
```

it seems people having FONDKAPREMONT_MODE "not specified" facing more problem in repayment of loan because their living building's one of attribite is not specified but it seems to be one of bank acccount information

- reg oper account : regular operational account
- reg oper spec account : regular spcific operational account
- not specified : Not specific
- org spec account : Organzation specific account

```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FONDKAPREMONT_MODE',hue='CODE_GENDER',data=df_trai
n)
plt.xticks(rotation=90,fontsize='x-small')
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
FONDKAPREMONT_MODE")
```

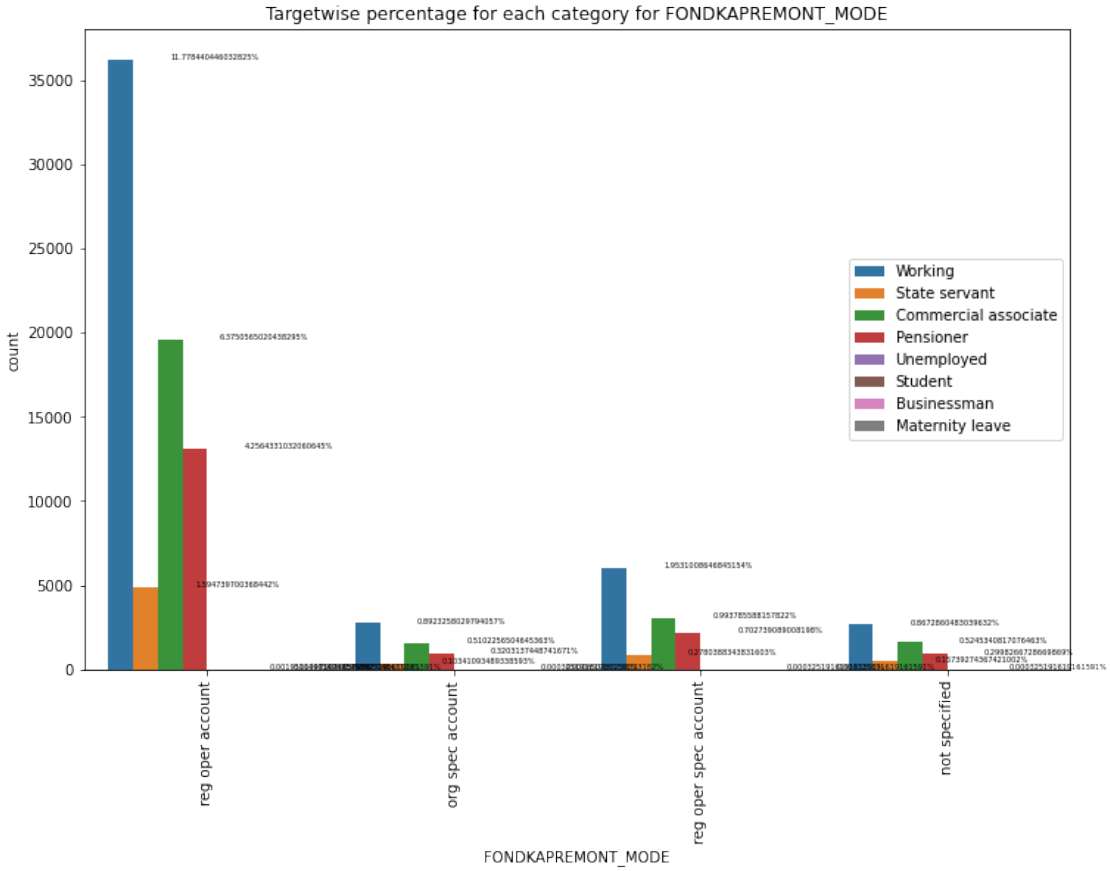
```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FONDKAPREMONT_MODE')
```



- most women have a specific account

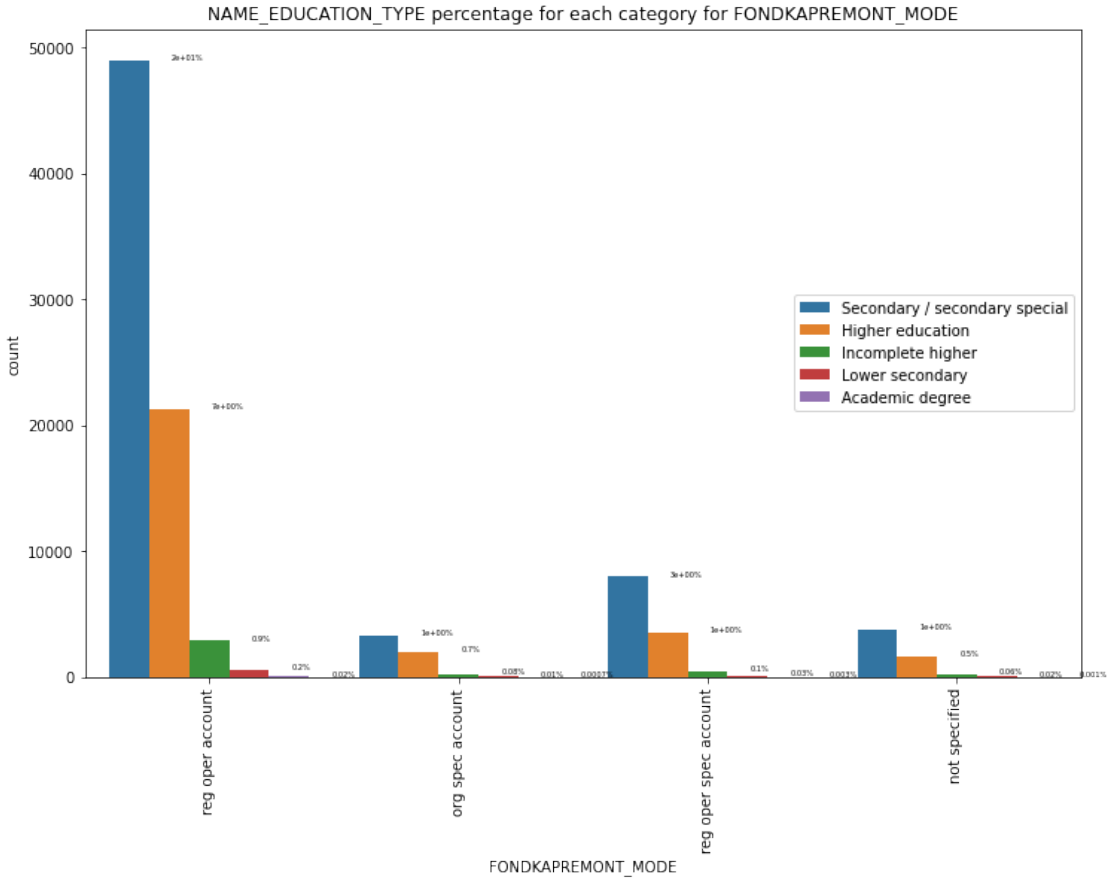
```
#lets see its distribution with resepect to NAME_INCOME_TYPE variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FONDKAPREMONT_MODE',hue='NAME_INCOME_TYPE',data=df
_train)
plt.xticks(rotation=90)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=5)

ax.set_title("Targetwise percentage for each category for
FONDKAPREMONT_MODE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FONDKAPREMONT_MODE')
```




```
#it shows that most of people whose FONDKAPREMONT_MODE is not
specified are state govt people and commerical associate
# we need to find a way to fill na values

#lets see its distribution with resepect to NAME_EDUCATION_TYPE
variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FONDKAPREMONT_MODE',hue='NAME_EDUCATION_TYPE',data
=df_train)
plt.xticks(rotation=90,)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.1}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=5)
ax.set_title("NAME_EDUCATION_TYPE percentage for each category for
FONDKAPREMONT_MODE")
```

```
Text(0.5, 1.0, 'NAME_EDUCATION_TYPE percentage for each category for
FONDKAPREMONT_MODE')
```




```
#we can fill na values with 'Not specified' because it not avilable
```

HOUSETYPE_MODE

Normalized information about building where the client lives,MODE

```
df_train['HOUSETYPE_MODE'].value_counts()
```

```
block of flats      150503
specific housing      1499
terraced house        1212
Name: HOUSETYPE_MODE, dtype: int64
```

```
df_train['HOUSETYPE_MODE'].isna().sum()
#Have NA values
```

```
154297
```

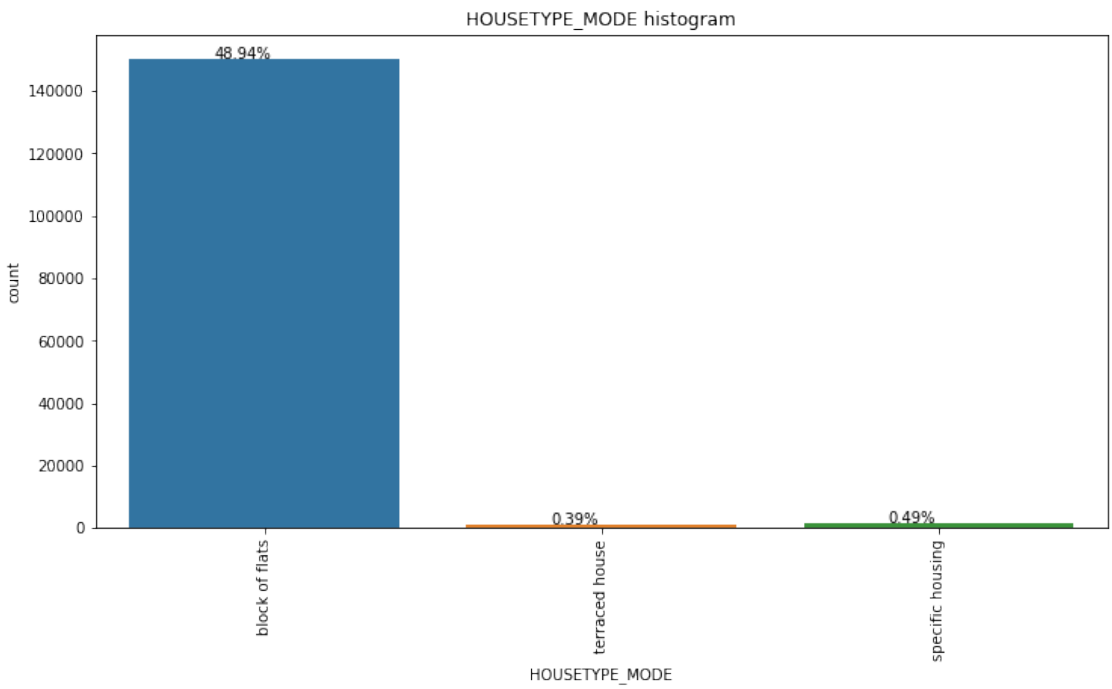
```
df_train['HOUSETYPE_MODE'].isna().sum()*100/df_train.shape[0]
```

```
50.176091261776
```

- 50 % approx fields are NAN actually

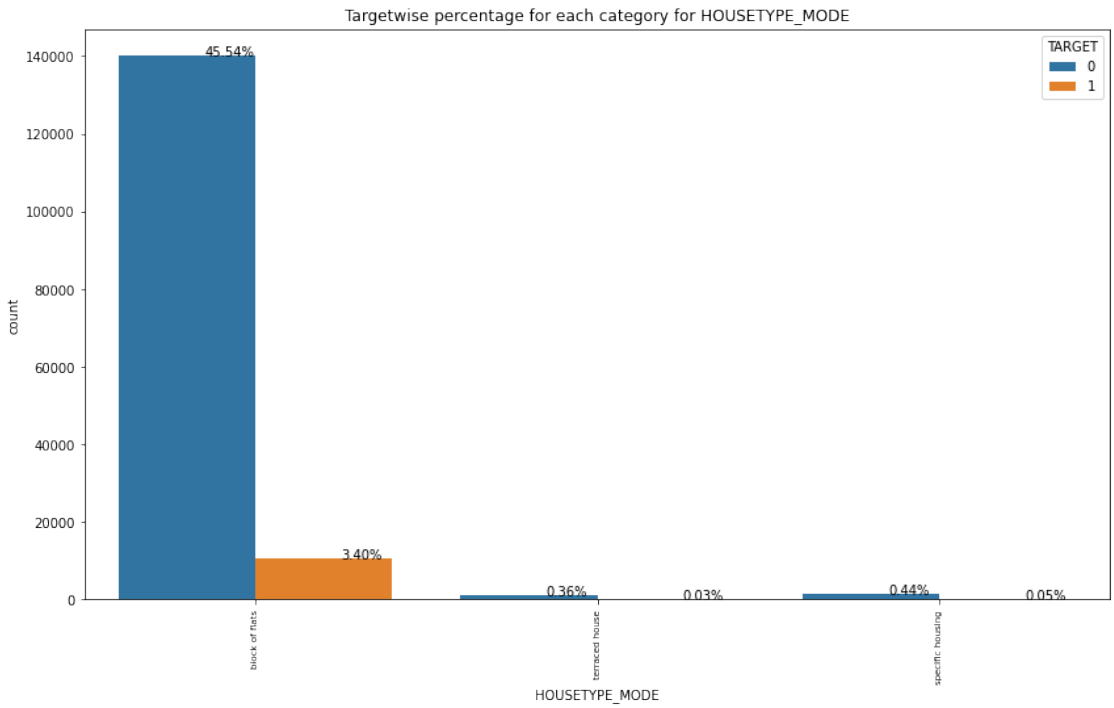
```
#lets see its distribution
plt.figure(figsize=(12,6))
ax=sns.countplot(x='HOUSETYPE_MODE',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))


ax.set_title("HOUSETYPE_MODE histogram")
```

```
Text(0.5, 1.0, 'HOUSETYPE_MODE histogram')
```



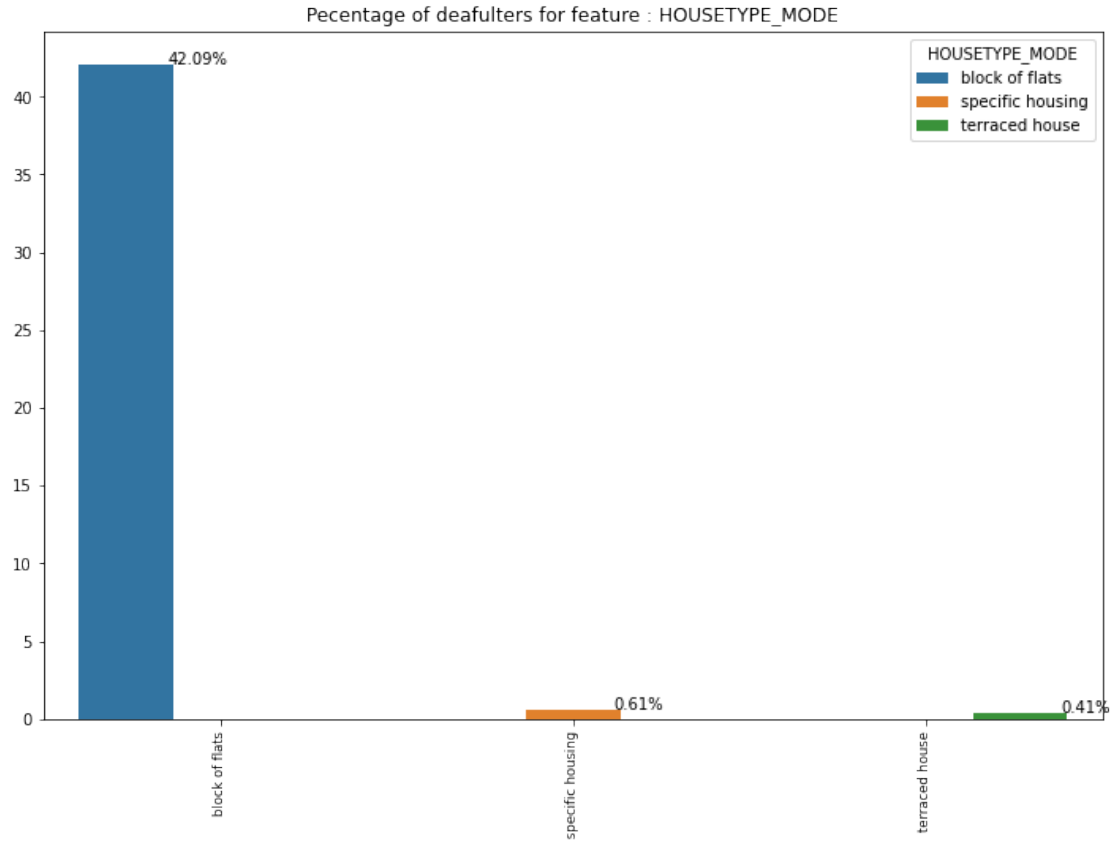

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(14,8))
ax=sns.countplot(x='HOUSETYPE_MODE',hue='TARGET',data=df_train)
plt.xticks(rotation=90,fontsize='x-small')
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Targetwise percentage for each category for
HOUSETYPE_MODE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
HOUSETYPE_MODE')
```




```
plot_for_categorical_defaulters(df_train,'HOUSETYPE_MODE')
```

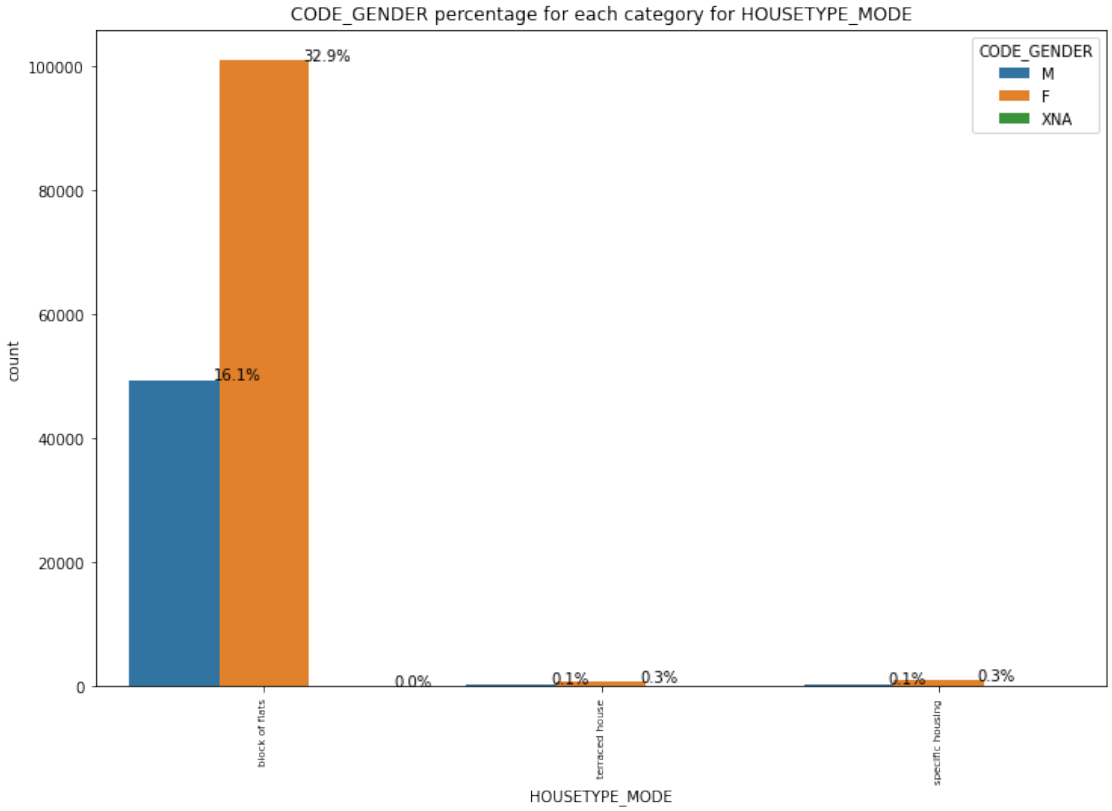
```
{'block of flats': 42.09466263846929,
 'specific housing': 0.6122860020140987,
 'terraced house': 0.4149043303121853}
```



it seems people having HOUSETYPE_MODE "specifc housing" facing more problem in repayment of loan because their housing is not that much enriched like Apartmet and Tarreced Home

```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='HOUSETYPE_MODE',hue='CODE_GENDER',data=df_train)
plt.xticks(rotation=90,fontsize='x-small')
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("CODE_GENDER percentage for each category for
HOUSETYPE_MODE")
```

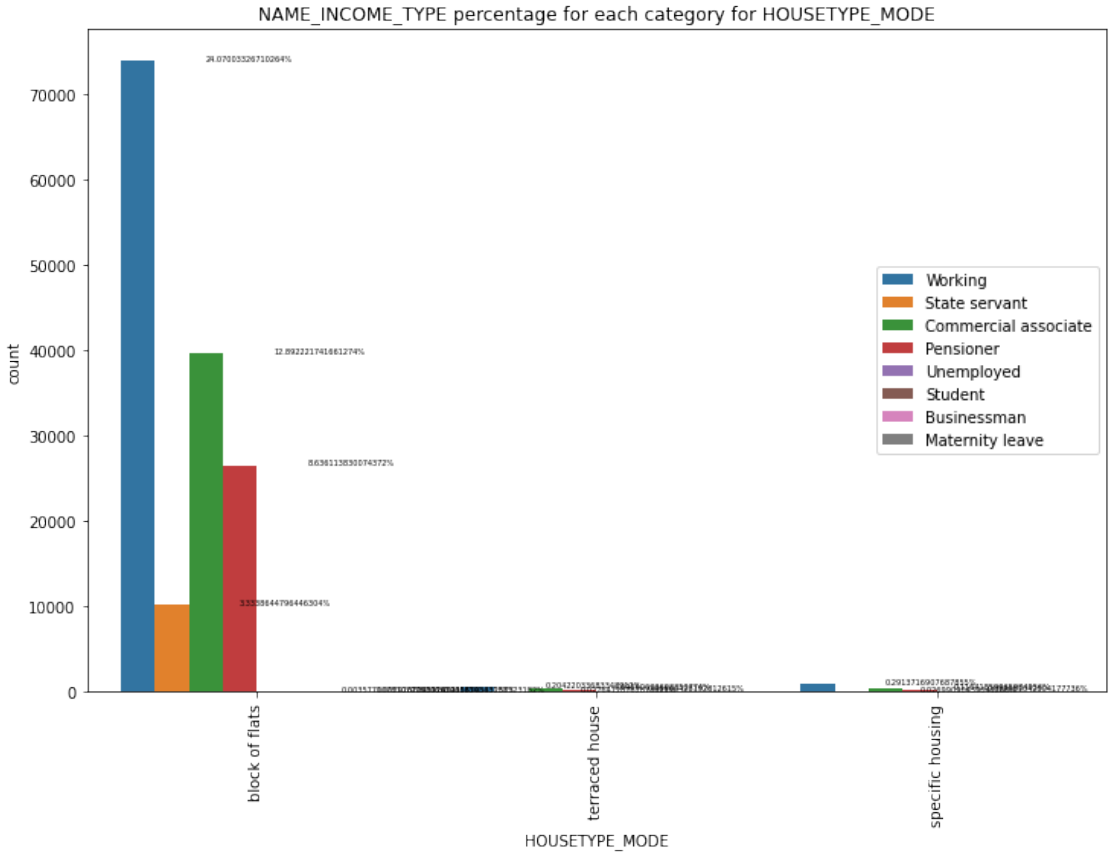
```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
HOUSETYPE_MODE')
```



- most women are living in aparments

```
#lets see its distribution with resepect to NAME_INCOME_TYPE variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='HOUSETYPE_MODE',hue='NAME_INCOME_TYPE',data=df_tra
in)
plt.xticks(rotation=90)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=5)
ax.set_title("NAME_INCOME_TYPE percentage for each category for
HOUSETYPE_MODE")
```

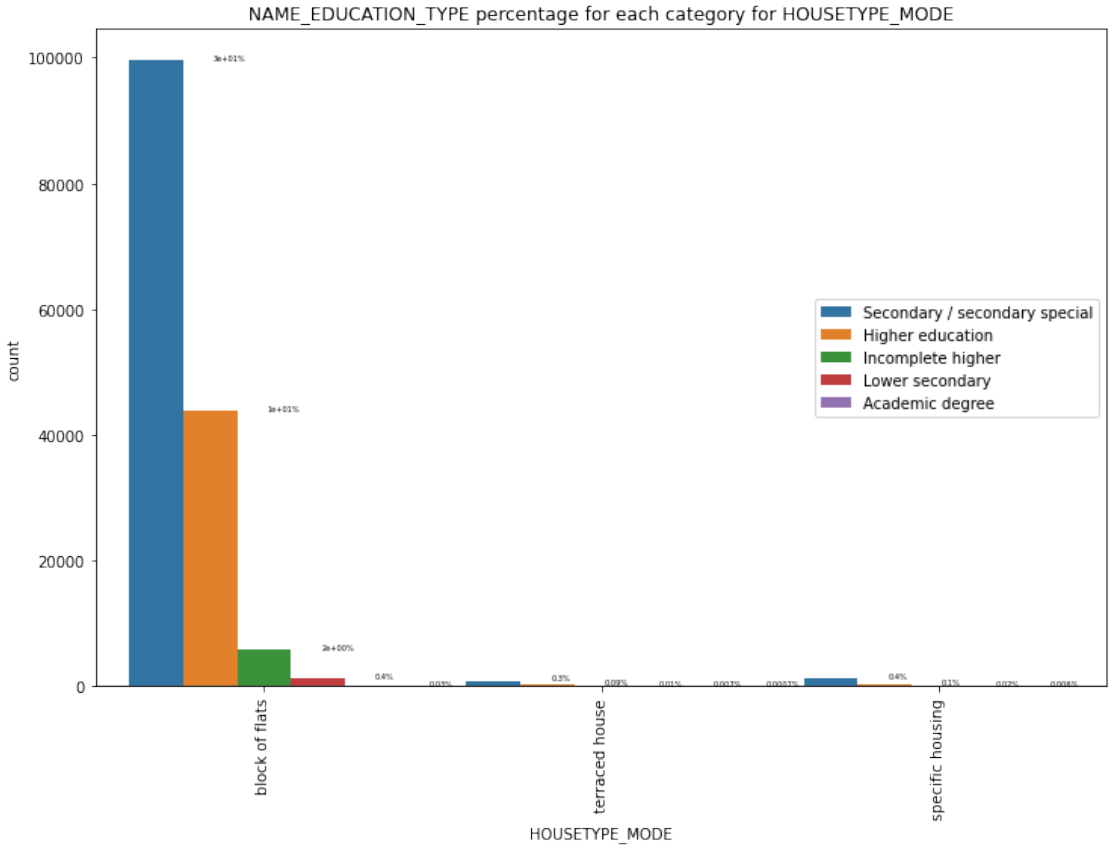
```
Text(0.5, 1.0, 'NAME_INCOME_TYPE percentage for each category for
HOUSETYPE_MODE')
```



- it shows that most of people whose HOUSETYPE_MODE is "block of flats" having education type working,state govt,commercial associate ,pensioner
- we need to find a way to fill na values

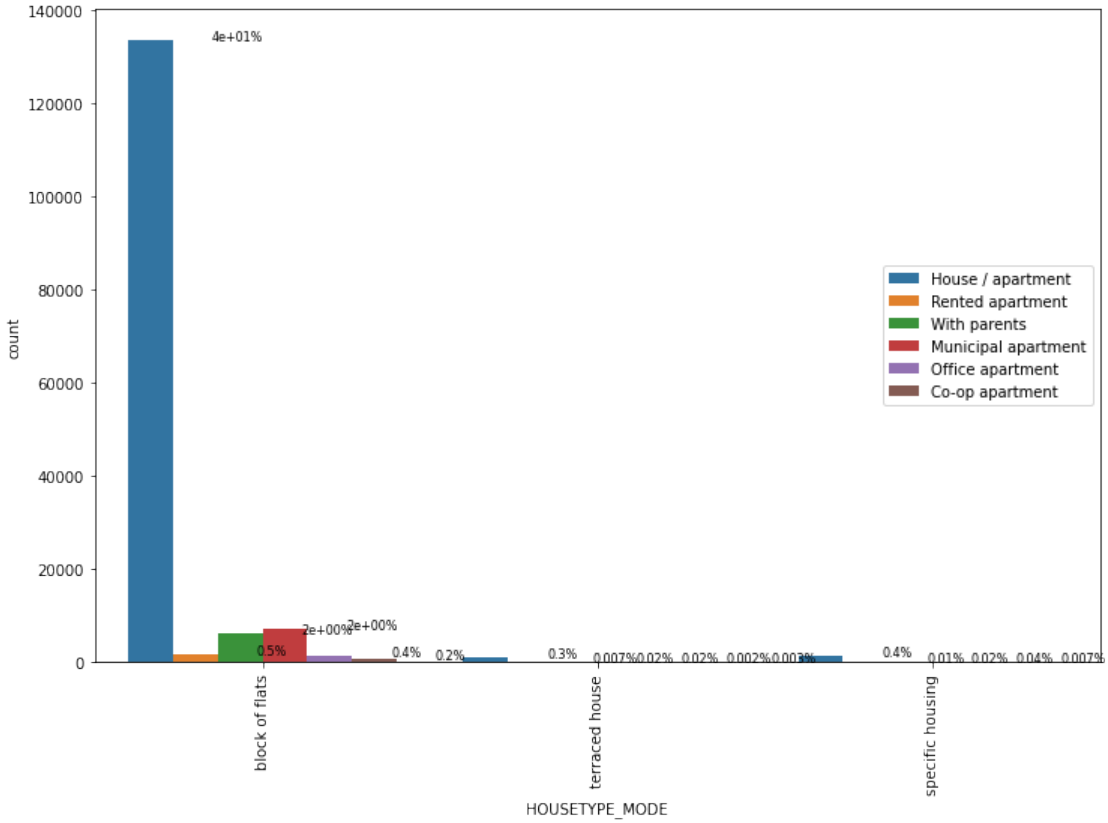
```
#lets see its distribution with resepect to NAME_EDUCATION_TYPE variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='HOUSETYPE_MODE',hue='NAME_EDUCATION_TYPE',data=df_train)
plt.xticks(rotation=90,)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.1}%'.format(p.get_height()*100/len(df_train)), (p.get_x()+0.25, p.get_height()+0.01),fontsize=5)
ax.set_title("NAME_EDUCATION_TYPE percentage for each category for HOUSETYPE_MODE")
```

```
Text(0.5, 1.0, 'NAME_EDUCATION_TYPE percentage for each category for HOUSETYPE_MODE')
```



- People having good eductaion seems to living in block of flats

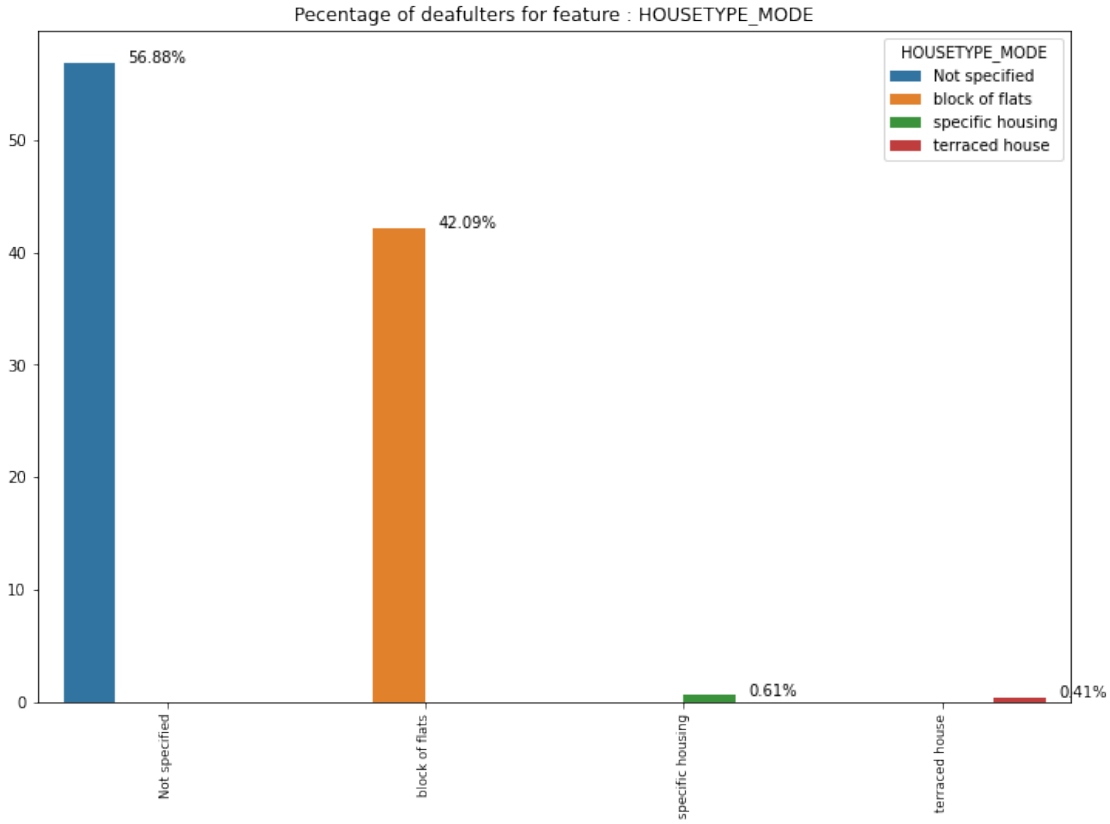
```
#lets see its distribution with resepect to NAME_HOUSING_TYPE
variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='HOUSETYPE_MODE',hue='NAME_HOUSING_TYPE',data=df_tr
ain)
plt.xticks(rotation=90)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.1}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=8)
ax.set_title("NAME_HOUSING_TYPE percentage for each category for
HOUSETYPE_MODE")
```



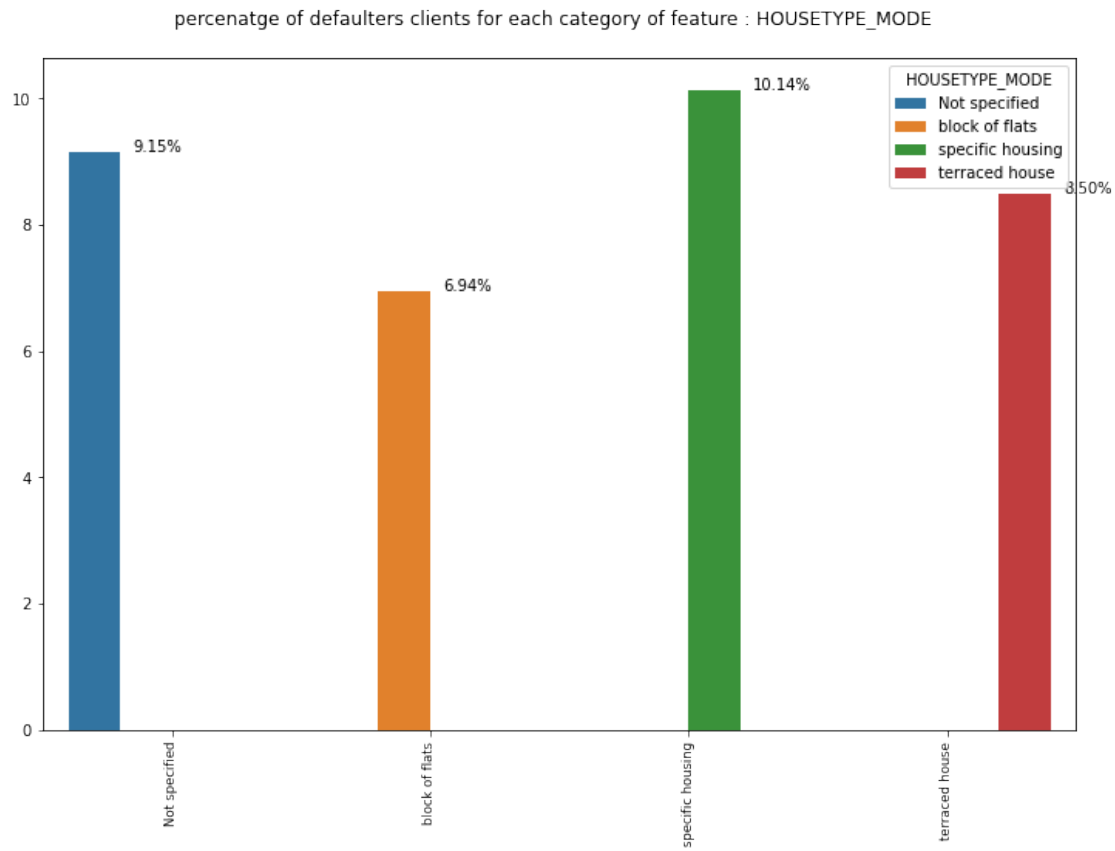

```
#block of flats exactly means Apartments/House
#Mostly "NAN" blank

#we can fill na values with 'Not specified' because it not avilable

df_HOUSETYPE_MODE_na=df_train[df_train['HOUSETYPE_MODE'].isna()]

df_HOUSETYPE_MODE_na['NAME_HOUSING_TYPE'].value_counts()
```

```
House / apartment      136897
With parents             8433
Municipal apartment      3872
Rented apartment         3201
Office apartment         1411
Co-op apartment           483
Name: NAME_HOUSING_TYPE, dtype: int64
```

```
df_train_copy= df_train.copy()

df_train_copy['HOUSETYPE_MODE']=df_train_copy['HOUSETYPE_MODE'].fillna
('Not specified')

plot_for_categorical_defaulters(df_train_copy,'HOUSETYPE_MODE')
```




```
plot_for_defaulters_categorywise(df_train_copy,'HOUSETYPE_MODE')
```



WALLSMATERIAL_MODE

Normalized information about building where the client lives

```
df_train['WALLSMATERIAL_MODE'].value_counts()
```

```
Panel           66040
Stone, brick    64815
Block            9253
Wooden           5362
Mixed            2296
Monolithic       1779
Others           1625
Name: WALLSMATERIAL_MODE, dtype: int64
```

```
df_train['WALLSMATERIAL_MODE'].isna().sum()
#Have NA values
```

```
156341
```

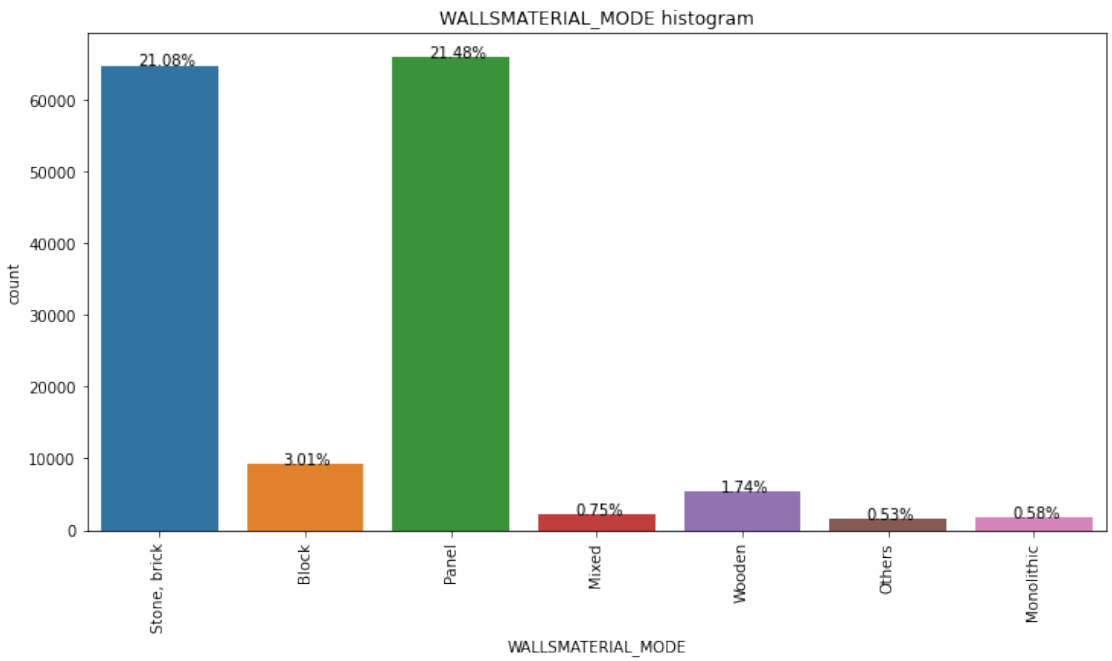
```
df_train['WALLSMATERIAL_MODE'].isna().sum()*100/df_train.shape[0]
```

```
50.8407829313423
```

- 50 % approx fields are NAN actually

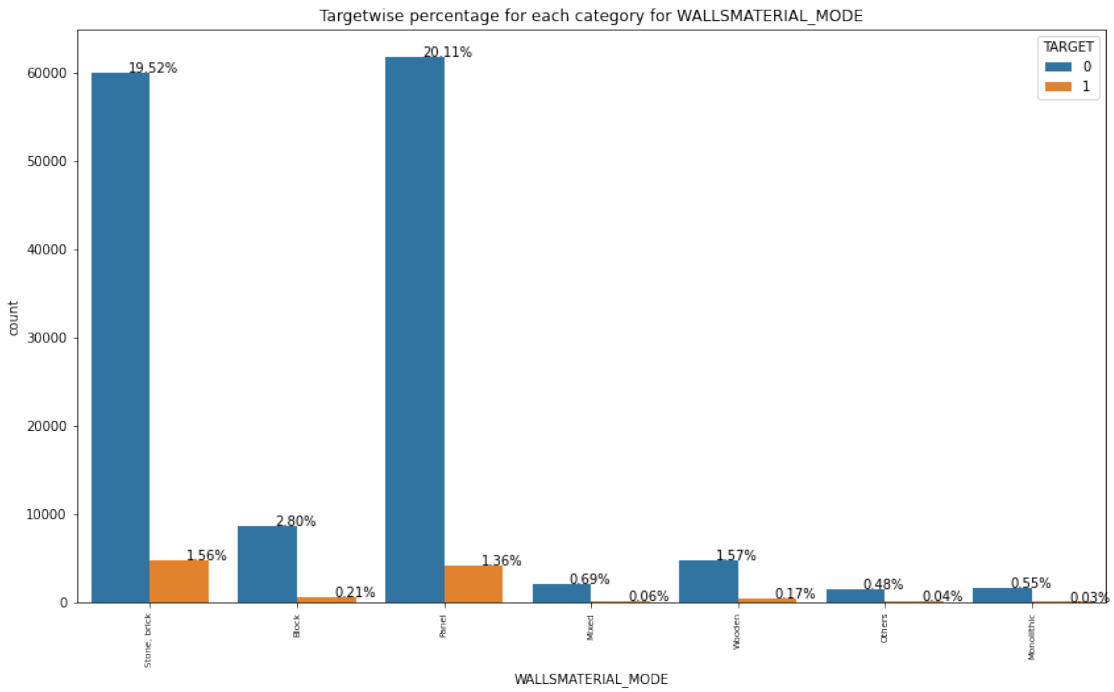
```
#lets see its distribution
plt.figure(figsize=(12,6))
ax=sns.countplot(x='WALLSMATERIAL_MODE',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("WALLSMATERIAL_MODE histogram")
```

```
Text(0.5, 1.0, 'WALLSMATERIAL_MODE histogram')
```

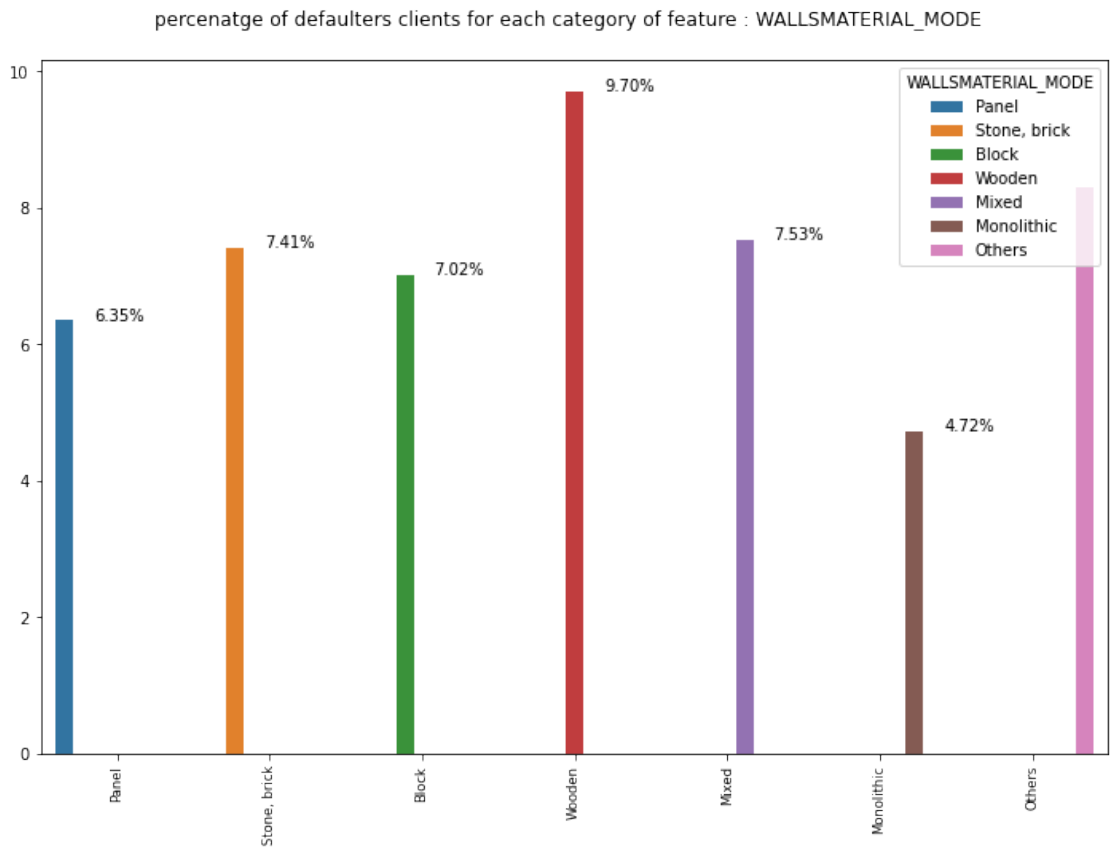



```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(14,8))
ax=sns.countplot(x='WALLSMATERIAL_MODE',hue='TARGET',data=df_train)
plt.xticks(rotation=90,fontsize='x-small')
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
WALLSMATERIAL_MODE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
WALLSMATERIAL_MODE')
```



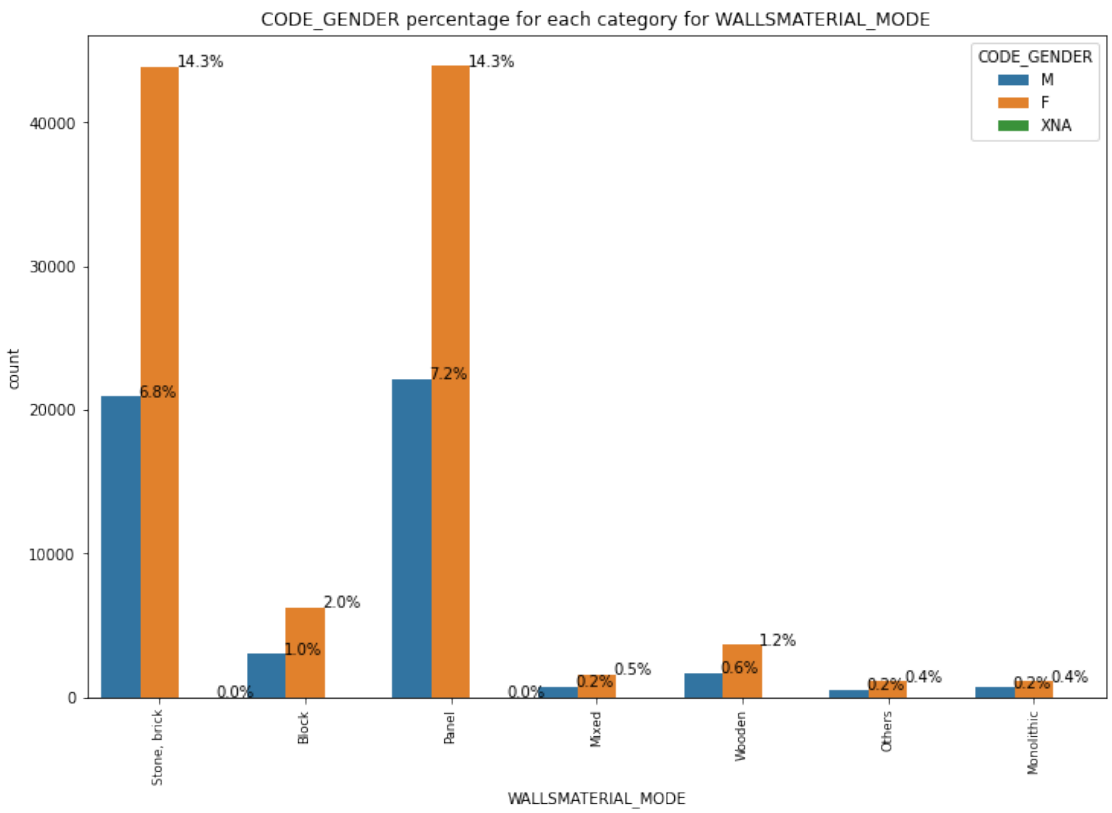

```
plot_for_defaulters_categorywise(df_train,'WALLSMATERIAL_MODE')
```



it seems people having WALLSMATERIAL_MODE "Wooden" and 'others' facing more problem in repayment of loan as their housing is also not that better ones

```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='WALLSMATERIAL_MODE',hue='CODE_GENDER',data=df_trai
n)
plt.xticks(rotation=90,fontsize='small')
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=10)

ax.set_title("CODE_GENDER percentage for each category for
WALLSMATERIAL_MODE")
```

```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
WALLSMATERIAL_MODE')
```

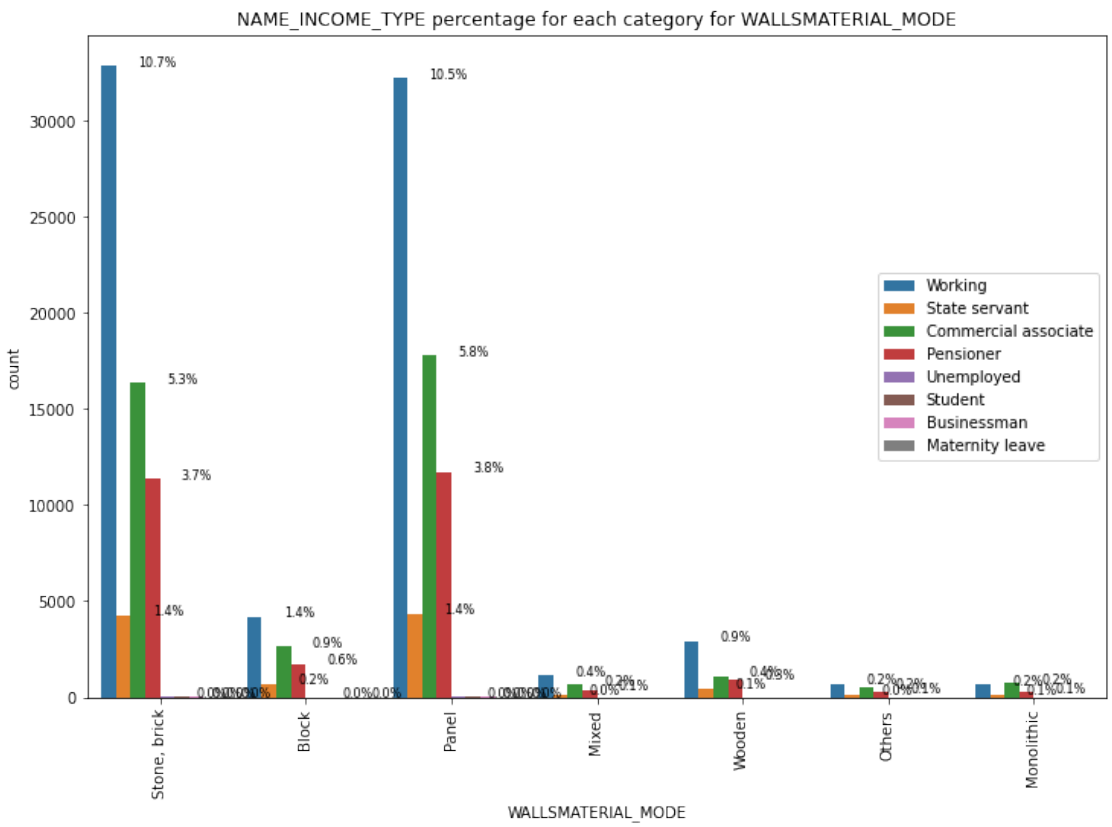


- most women are living in better houses

```
#lets see its distribution with resepect to NAME_INCOME_TYPE variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='WALLSMATERIAL_MODE',hue='NAME_INCOME_TYPE',data=df
_train)
plt.xticks(rotation=90)
ax.legend(loc='right')
for p in ax.patches:
```

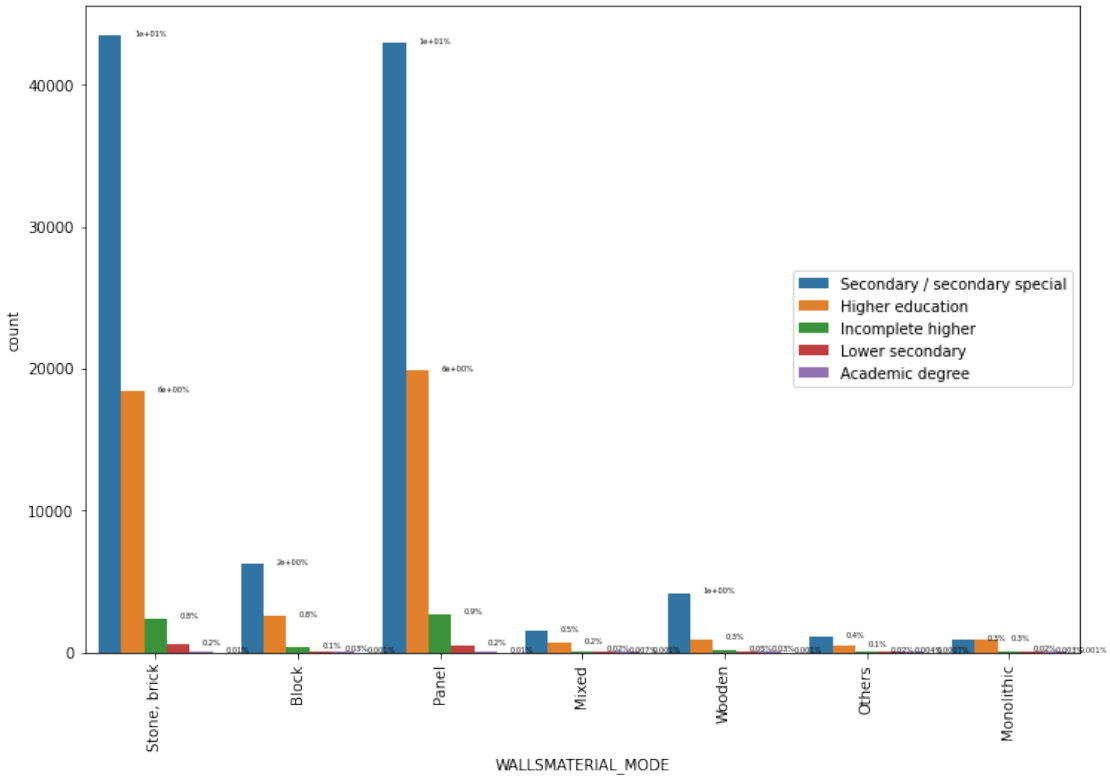
```
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=8)

ax.set_title("NAME_INCOME_TYPE percentage for each category for
WALLSMATERIAL_MODE")
```

```
Text(0.5, 1.0, 'NAME_INCOME_TYPE percentage for each category for
WALLSMATERIAL_MODE')
```



- it shows that most of people whose WALLSMATERIAL_MODE are kind of made of concrete and all having income type type working,state govt,commercial associate ,pensioner
- we need to find a way to fill na values

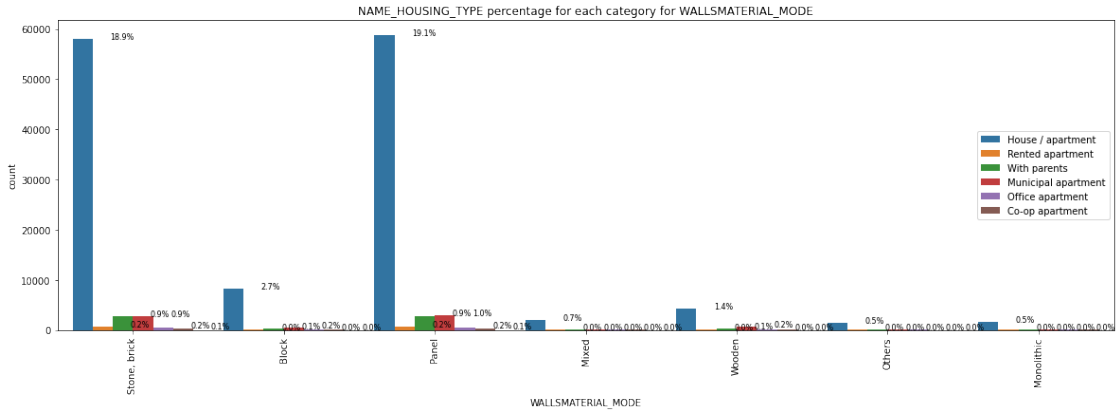
```
#lets see its distribution with resepect to NAME_EDUCATION_TYPE
variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='WALLSMATERIAL_MODE',hue='NAME_EDUCATION_TYPE',data
=df_train)
plt.xticks(rotation=90,)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.1}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=5)
```



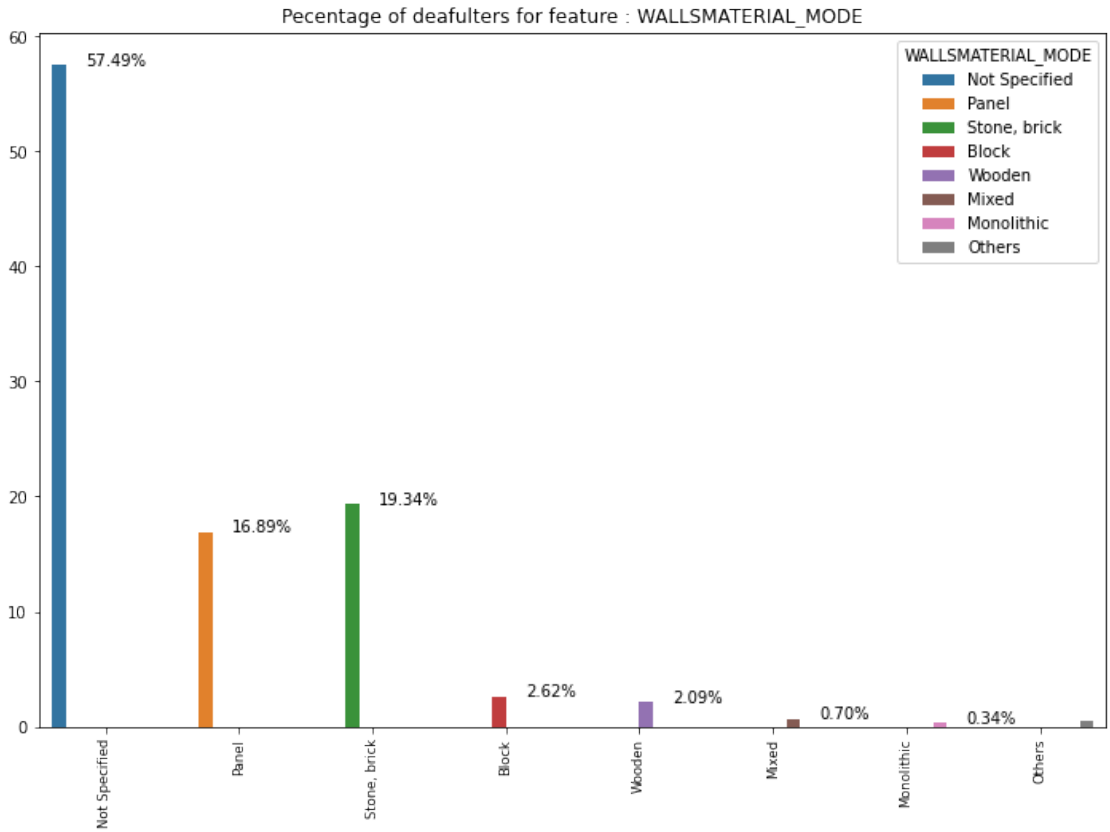
- People having good eductaion seems to living strong houses

```
#lets see its distribution with resepect to NAME_HOUSING_TYPE
variables
plt.figure(figsize=(20,6))
ax=sns.countplot(x='WALLSMATERIAL_MODE',hue='NAME_HOUSING_TYPE',data=d
f_train)
plt.xticks(rotation=90)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=8)

ax.set_title("NAME_HOUSING_TYPE percentage for each category for
WALLSMATERIAL_MODE")
```

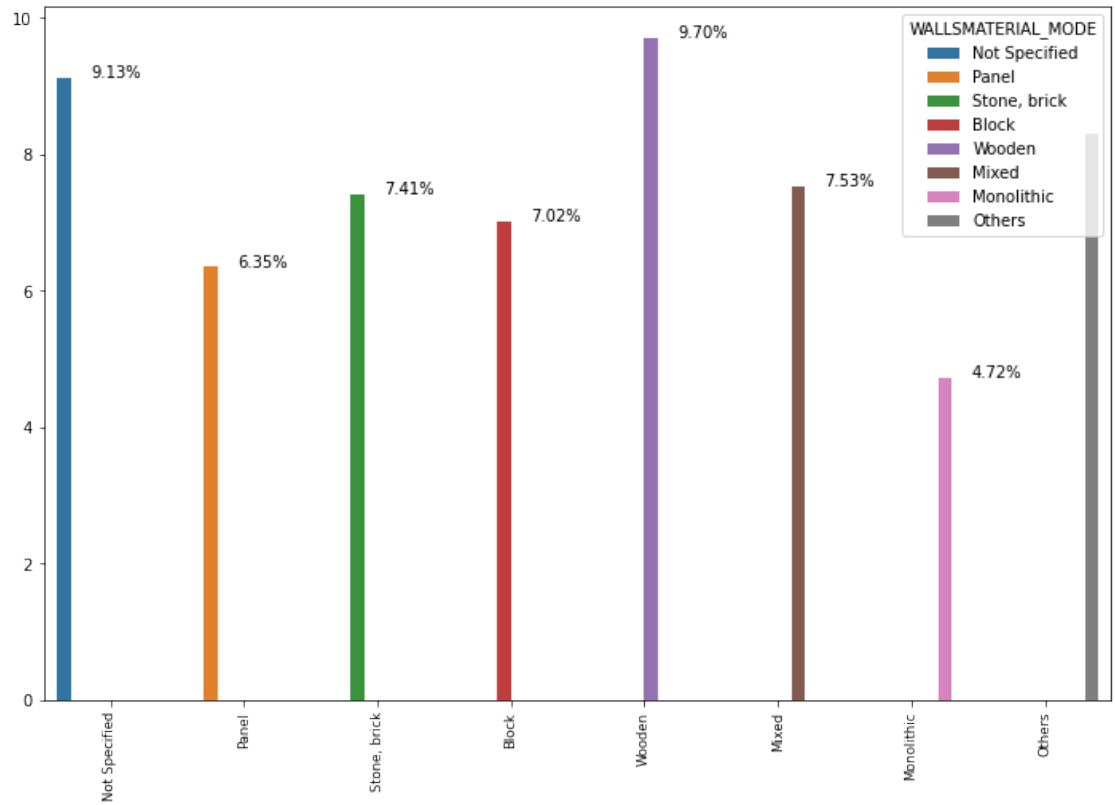
```
Text(0.5, 1.0, 'NAME_HOUSING_TYPE percentage for each category for
WALLSMATERIAL_MODE')
```




```
df_train_copy= df_train.copy()

df_train_copy['WALLSMATERIAL_MODE']=df_train_copy['WALLSMATERIAL_MODE'
].fillna("Not Specified")

x=plot_for_categorical_defaulters(df_train_copy,'WALLSMATERIAL_MODE')
```




```
plot_for_deafulters_categorywise(df_train_copy,'WALLSMATERIAL_MODE')
```




```
map_wallsmaterial_type={k: v for k, v in sorted(x.items(), key=lambda
item: item[1])}
```

```
map_wallsmaterial_type
```

```
{'Monolithic': 0.338368580060423,
 'Others': 0.5438066465256798,
 'Mixed': 0.6968781470292045,
 'Wooden': 2.094662638469285,
 'Block': 2.618328298086606,
 'Panel': 16.886203423967775,
 'Stone, brick': 19.335347432024168,
 'Not Specified': 57.48640483383686}
```

```
map_wallsmaterial_type={'Monolithic': 0,
                        'Others': 0,
                        'Mixed': 0,
                        'Wooden': 1,
                        'Block': 1,
                        'Panel': 1,
                        'Stone, brick': 1,
                        'Not Specified':2}
```

EMERGENCYSTATE_MODE

Normalized information about building where the client lives,Is that state have emergency kind of situtaion ?

```
df_train['EMERGENCYSTATE_MODE'].value_counts()
```

```
No     159428
Yes      2328
Name: EMERGENCYSTATE_MODE, dtype: int64
```

```
df_train['EMERGENCYSTATE_MODE'].isna().sum()
#Have NA values
```

```
145755
```

```
df_train['EMERGENCYSTATE_MODE'].isna().sum()*100/df_train.shape[0]
```

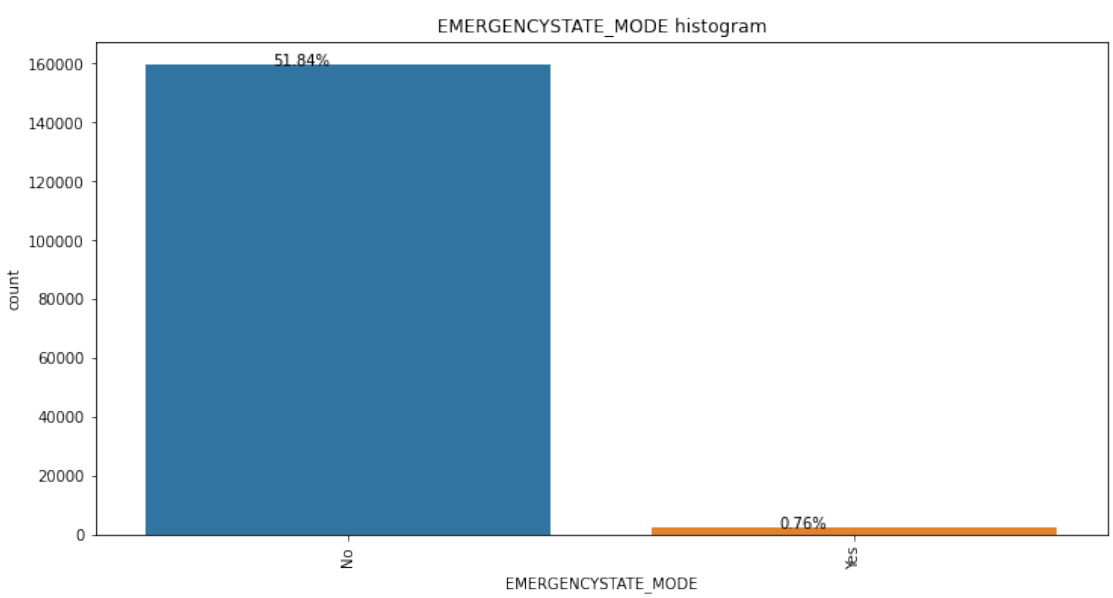
```
47.39830445089769
```

- 47 % approx fields are NAN actually,we can fill na here with "NO"

```
#lets see its distribution
plt.figure(figsize=(12,6))
ax=sns.countplot(x='EMERGENCYSTATE_MODE',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
```

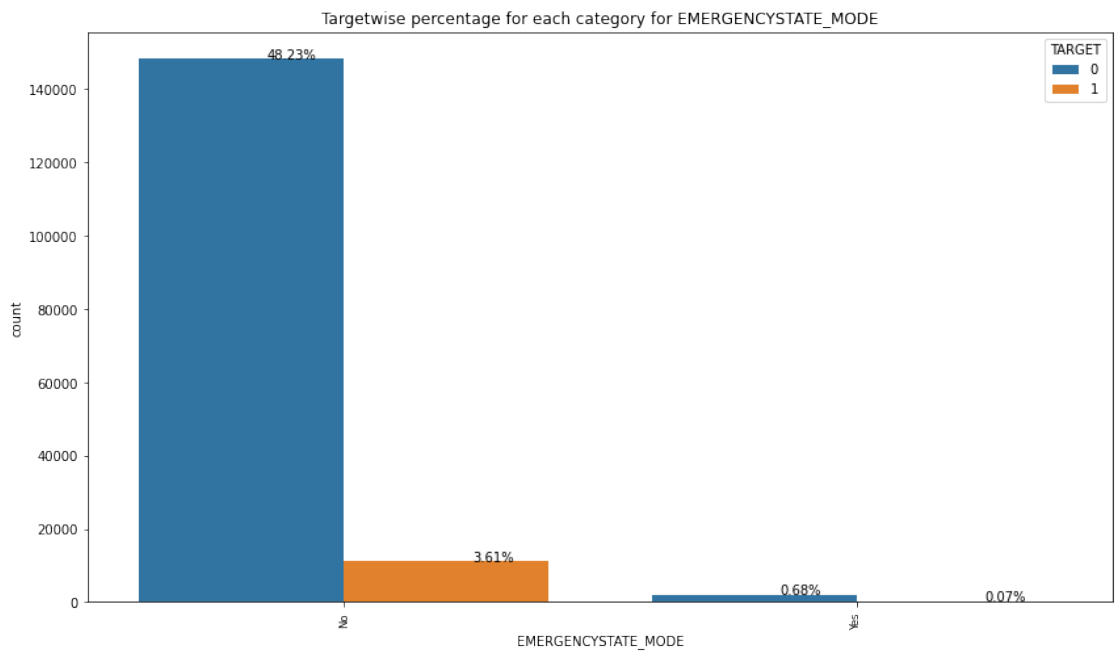
```
ax.set_title("EMERGENCYSTATE_MODE histogram")
```

```
Text(0.5, 1.0, 'EMERGENCYSTATE_MODE histogram')
```

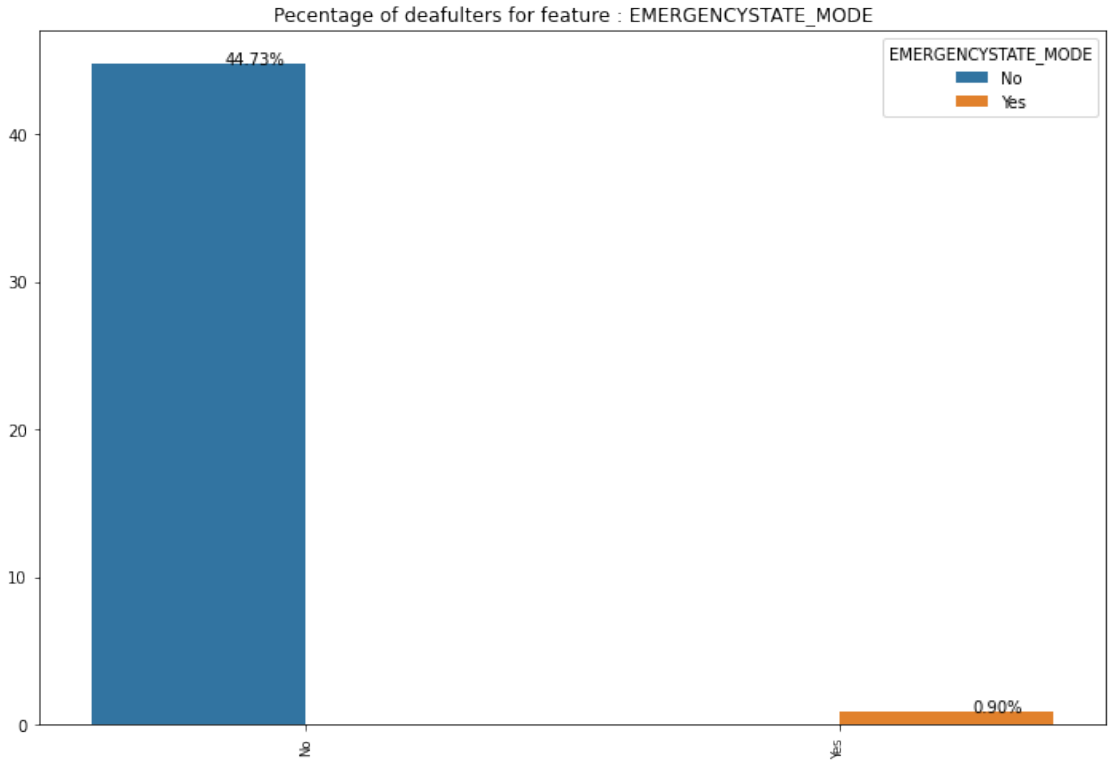


```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(14,8))
ax=sns.countplot(x='EMERGENCYSTATE_MODE',hue='TARGET',data=df_train)
plt.xticks(rotation=90,fontsize='small')
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
EMERGENCYSTATE_MODE")
```

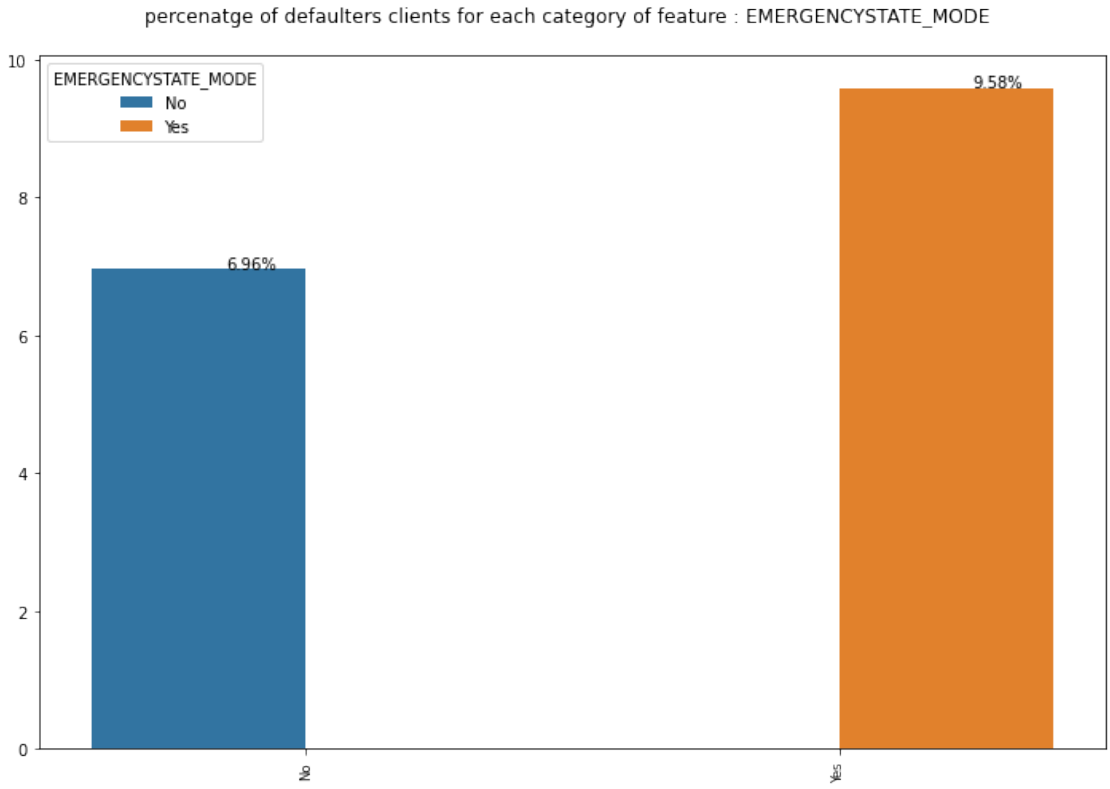
```
Text(0.5, 1.0, 'Targetwise percentage for each category for
EMERGENCYSTATE_MODE')
```




```
plot_for_categorical_defaulters(df_train,'EMERGENCYSTATE_MODE')
```



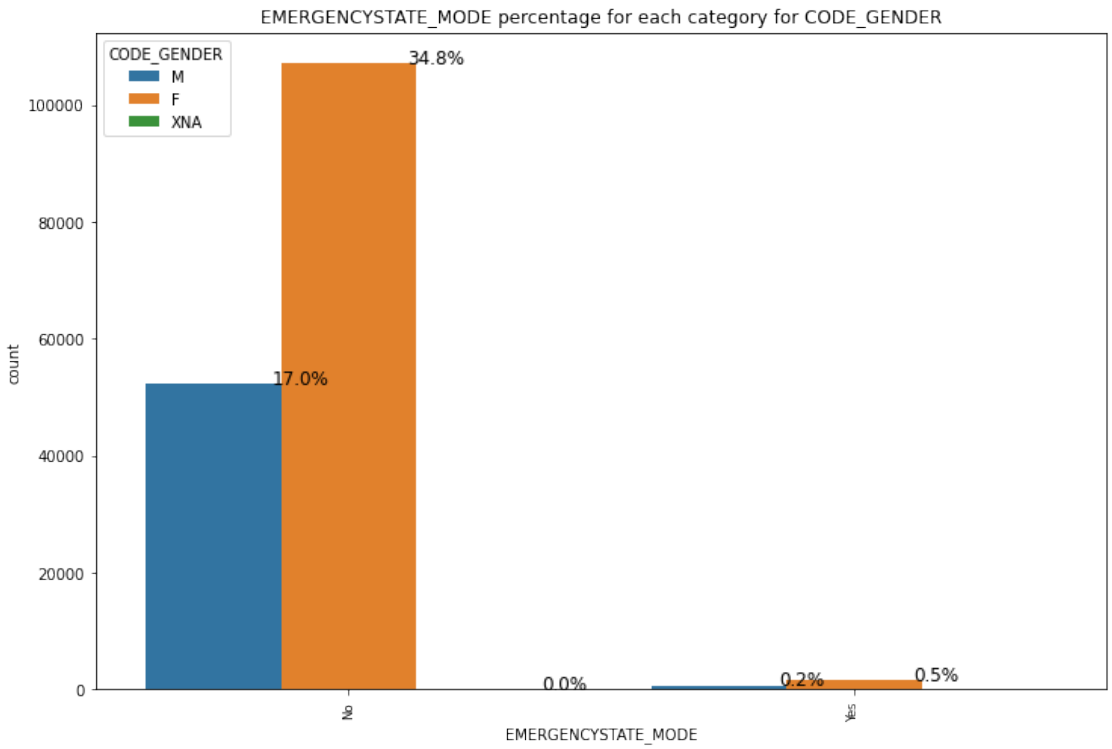

```
plot_for_defaulters_categorywise(df_train,'EMERGENCYSTATE_MODE')
```



it seems people having EMERGENCYSTATE_MODE as "YES " facing more problem in repayment of loan as their living state mostly have emergency kind of situtauon, like Eathqukae,Flood,Violence and crime

```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='EMERGENCYSTATE_MODE',hue='CODE_GENDER',data=df_tra
in)
plt.xticks(rotation=90,fontsize='small')
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=12)

ax.set_title("EMERGENCYSTATE_MODE percentage for each category for
CODE_GENDER")
```

```
Text(0.5, 1.0, 'EMERGENCYSTATE_MODE percentage for each category for
CODE_GENDER')
```

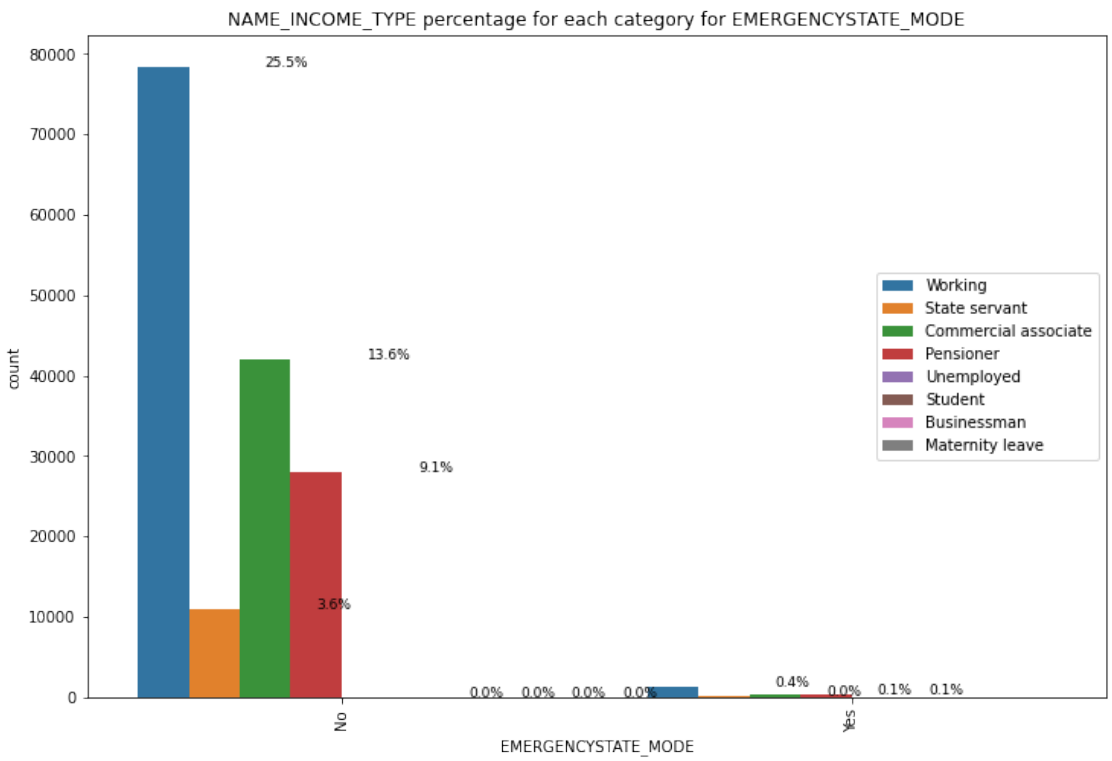


- Most women are living in better states which dont have emergency kind of situation, but it also showing that because of count of each catregories

```
#lets see its distribution with resepect to NAME_INCOME_TYPE variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='EMERGENCYSTATE_MODE',hue='NAME_INCOME_TYPE',data=d
f_train)
plt.xticks(rotation=90)
ax.legend(loc='right')
for p in ax.patches:
```

```
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=9)

ax.set_title("NAME_INCOME_TYPE percentage for each category for
EMERGENCYSTATE_MODE")
```

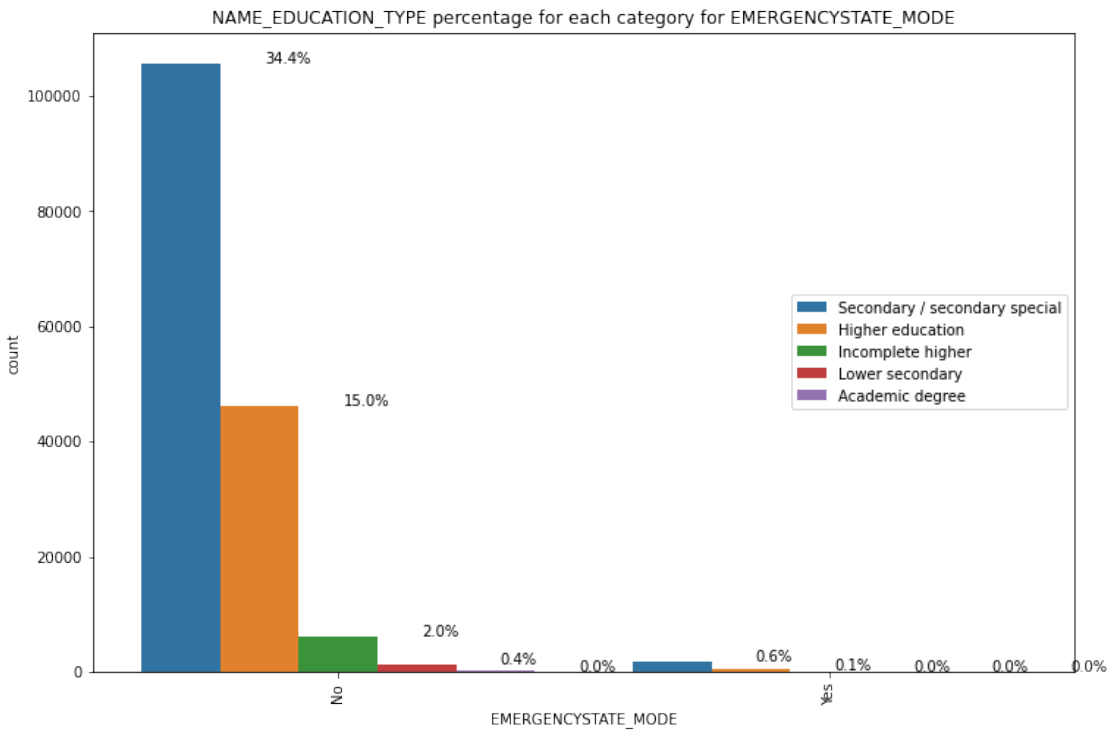
```
Text(0.5, 1.0, 'NAME_INCOME_TYPE percentage for each category for
EMERGENCYSTATE_MODE')
```



- it shows that most of people whose EMERGENCYSTATE_MODE is "yes" mostly have working,state govt,ans some pensioner

```
#lets see its distribution with resepect to NAME_EDUCATION_TYPE
variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='EMERGENCYSTATE_MODE',hue='NAME_EDUCATION_TYPE',dat
a=df_train)
plt.xticks(rotation=90,)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=10)

ax.set_title("NAME_EDUCATION_TYPE percentage for each category for
EMERGENCYSTATE_MODE")
```

```
Text(0.5, 1.0, 'NAME_EDUCATION_TYPE percentage for each category for
EMERGENCYSTATE_MODE')
```



- People having good eductaion seems to living mostly in calm states

So now as we have already studied the Categorical columns,let's get Numerical columns

Continous Varirables

```
#Lets see how many columns are continous or numerical

numerics = ['int16', 'int32', 'int64', 'float16', 'float32',
'float64']
continous_variables=df_train.select_dtypes(include=numerics).columns.t
o_list()
len(continous_variables)
```

```
106
```

CNT_CHILDREN

Number of children the client has

```
df_train['CNT_CHILDREN'].value_counts()
```

```
0     215371
1      61119
2      26749
```

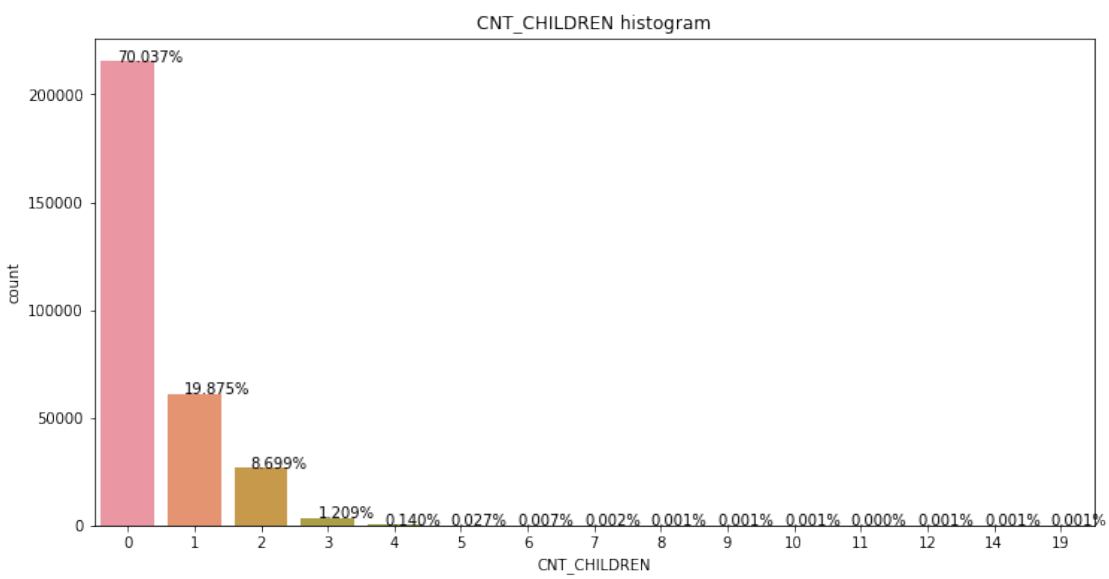
```
3       3717
4        429
5         84
6         21
7          7
14         3
8          2
9          2
10         2
12         2
19         2
11         1
Name: CNT_CHILDREN, dtype: int64
```

```
df_train['CNT_CHILDREN'].isna().sum()
#No NA values
```

```
0
```

```
#lets see its distribution
plt.figure(figsize=(12,6))
ax=sns.countplot(x='CNT_CHILDREN',data=df_train)
plt.xticks(rotation=0)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("CNT_CHILDREN histogram")
```

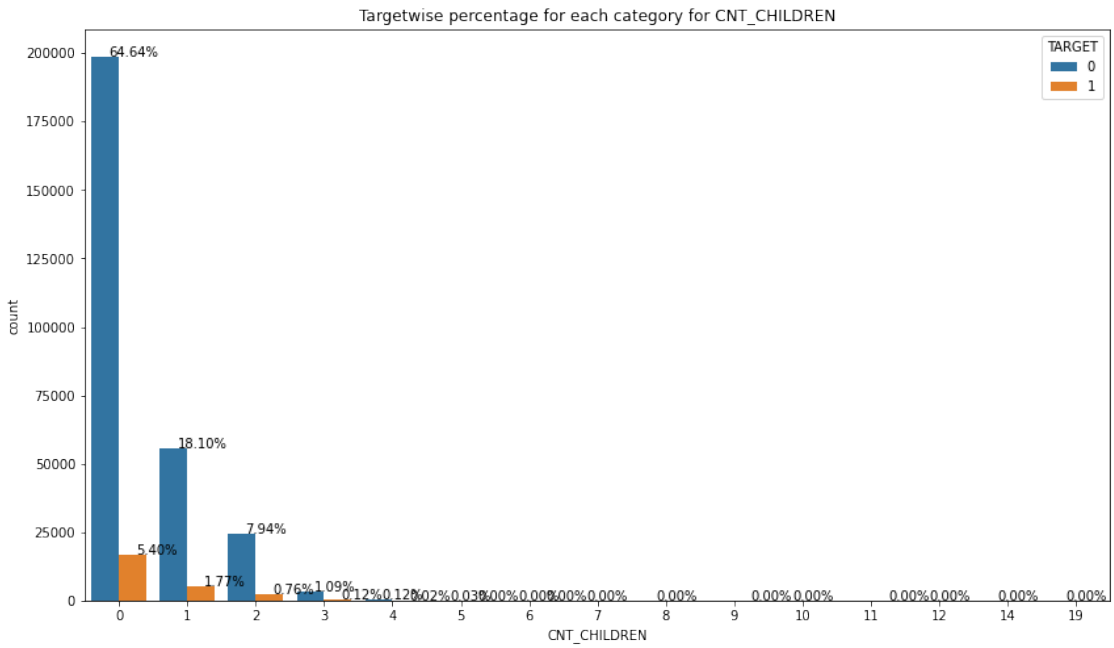
```
Text(0.5, 1.0, 'CNT_CHILDREN histogram')
```

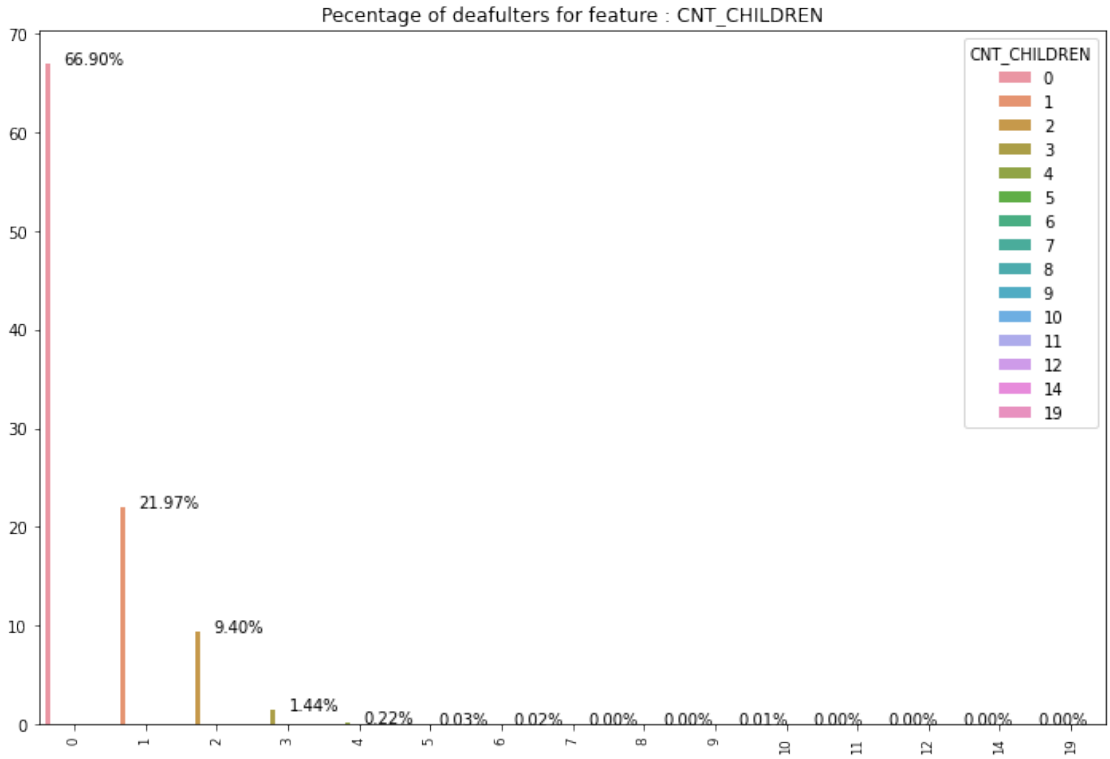



```
#its normal ,it follow power law distribution

#lets see its distribution with resepect to target variables
plt.figure(figsize=(14,8))
```

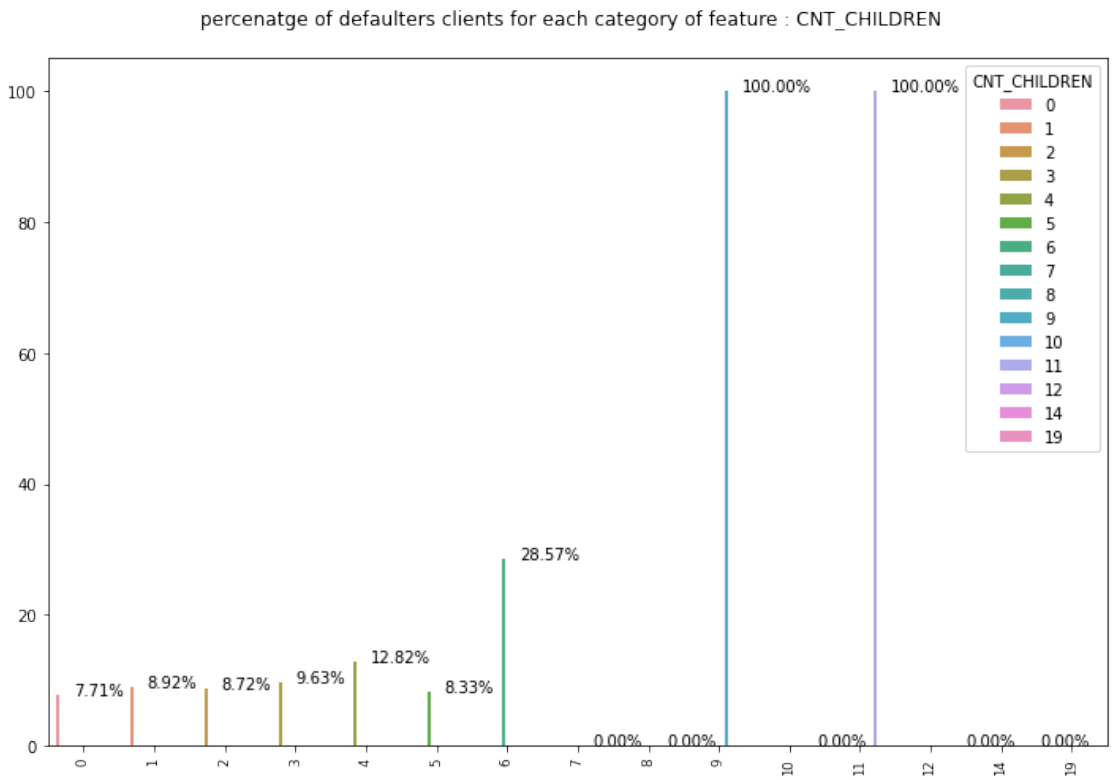
```
ax=sns.countplot(x='CNT_CHILDREN',hue='TARGET',data=df_train)
plt.xticks(rotation=0)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Targetwise percentage for each category for
CNT_CHILDREN")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
CNT_CHILDREN')
```


Targetwise percentage for each category for CNT_CHILDREN
TARGET
0
1
count
CNT_CHILDREN
64.64%
5.40%
18.10%
1.77%
7.94%
0.76%
1.09%
0.12%
0.00%
0.00%
0.00



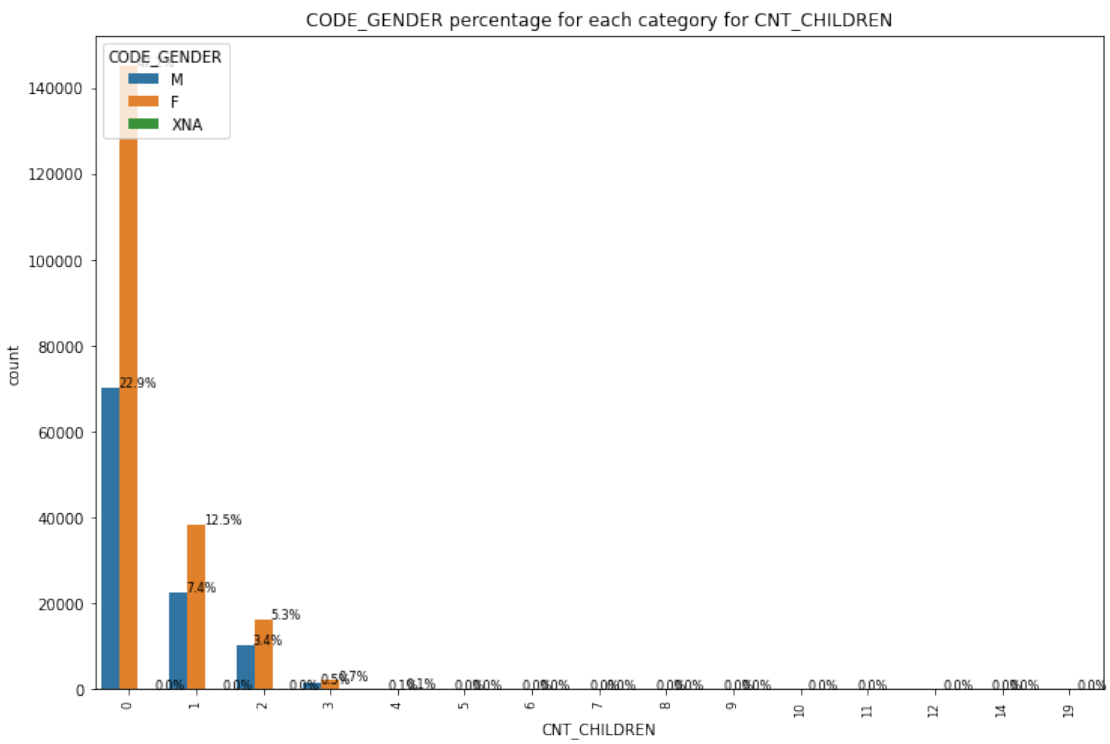

```
plot_for_defaulters_categorywise(df_train,'CNT_CHILDREN')
```



More the kids more the chances of being defaulter

```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='CNT_CHILDREN',hue='CODE_GENDER',data=df_train)
plt.xticks(rotation=90,fontsize='small')
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),fontsize=8)
ax.set_title("CODE_GENDER percentage for each category for
CNT_CHILDREN")
```

```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
CNT_CHILDREN')
```



- most women who are applying for loan are seems to have 0 to 3 kids.Max are without kids

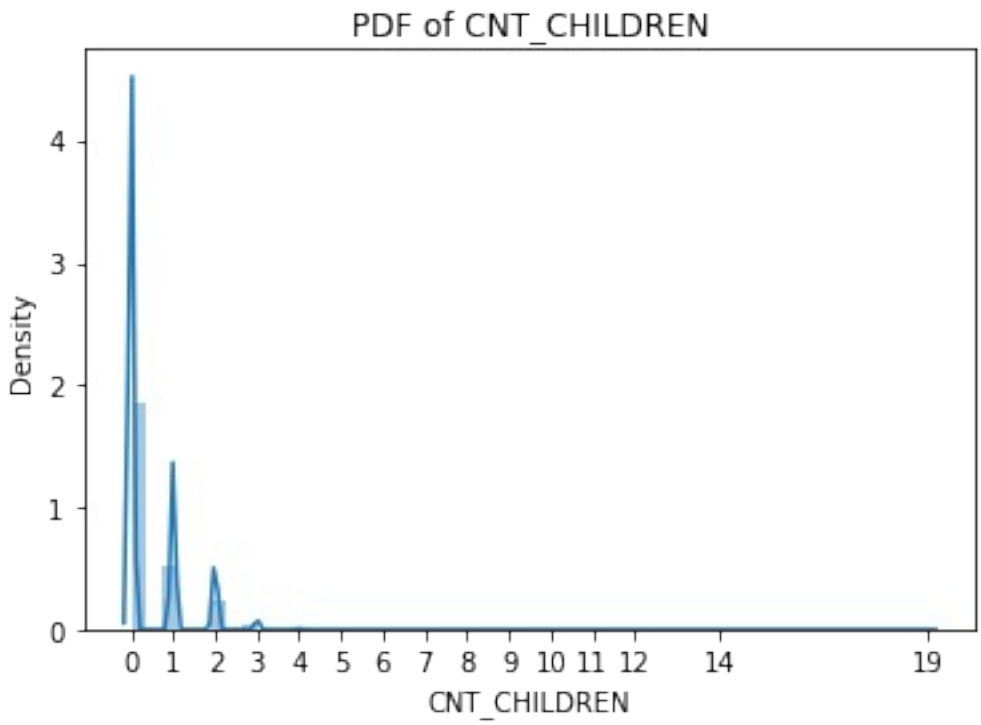
```
df_corr[['CNT_CHILDREN']].sort_values('CNT_CHILDREN',ascending=False)
[:5]
```

```
                    CNT_CHILDREN
CNT_CHILDREN            1.000000
CNT_FAM_MEMBERS         0.879161
DAYS_BIRTH              0.330938
FLAG_EMP_PHONE          0.240714
DAYS_REGISTRATION       0.183395
```

- well this is normal anology ,Size of family increase with kids,So both are higly correlated and DAYS_BIRTH as client age is also much correlated

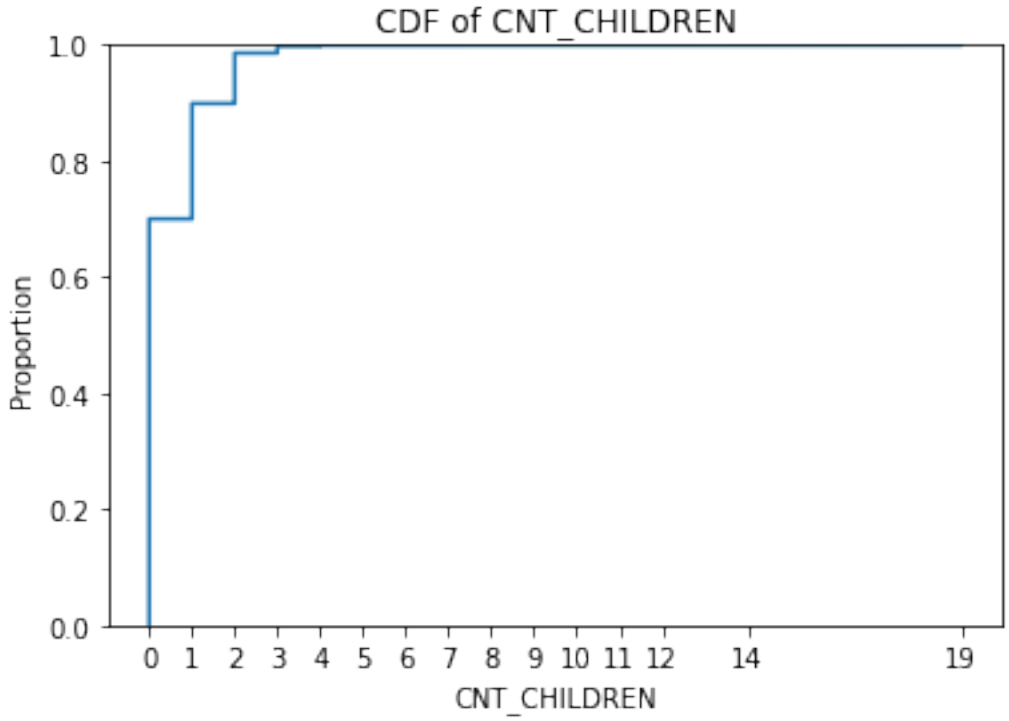
```
#lets see its PDF and CDF

sns.distplot(df_train['CNT_CHILDREN'])
plt.xticks(df_train['CNT_CHILDREN'].value_counts().index.to_list())
plt.title("PDF of CNT_CHILDREN")
```

```
Text(0.5, 1.0, 'PDF of CNT_CHILDREN')
```




```
sns.ecdfplot(df_train['CNT_CHILDREN'])
plt.xticks(df_train['CNT_CHILDREN'].value_counts().index.to_list())
plt.title("CDF of CNT_CHILDREN")
```

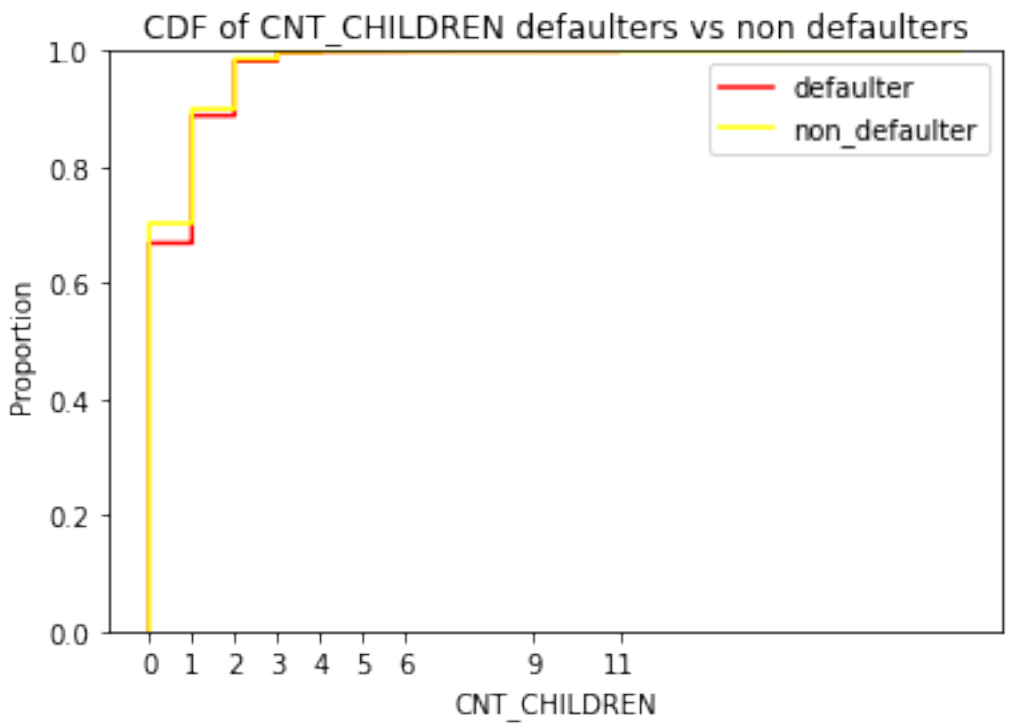
```
Text(0.5, 1.0, 'CDF of CNT_CHILDREN')
```



- 99% client seems to have lesser kids than 4
- 65% without kids

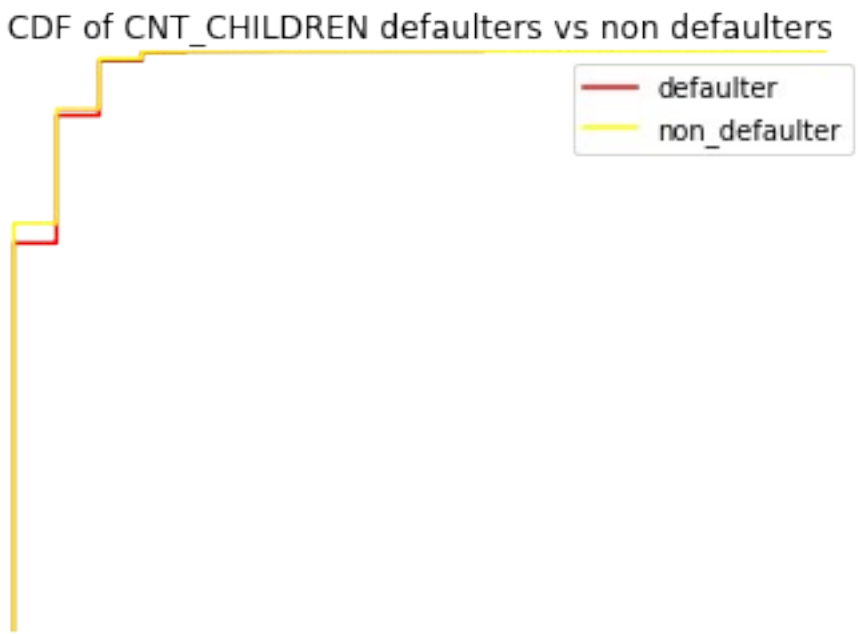
```
sns.ecdfplot(df_defaulter['CNT_CHILDREN'],label='defaulter',color='red
')
sns.ecdfplot(df_non_defaulter['CNT_CHILDREN'],label='non_defaulter',co
lor='yellow')
plt.xticks(df_defaulter['CNT_CHILDREN'].value_counts().index.to_list()
)
plt.title("CDF of CNT_CHILDREN defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1e24e81cf10>
```



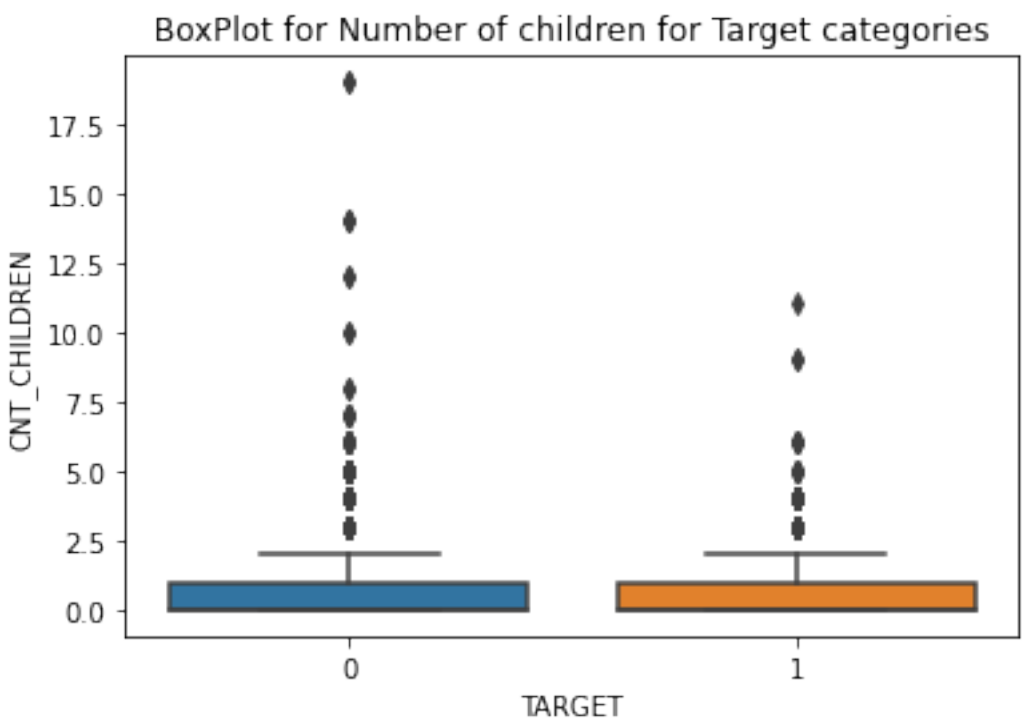

```
sns.ecdfplot(df_defaulter['CNT_CHILDREN'],label='defaulter',color='red
')
sns.ecdfplot(df_non_defaulter['CNT_CHILDREN'],label='non_defaulter',co
lor='yellow')
plt.xticks(df_defaulter['CNT_CHILDREN'].value_counts().index.to_list()
)
plt.title("CDF of CNT_CHILDREN defaulters vs non defaulters")
plt.legend()

plt.axis('off')#JNust for clarity
```

```
(-0.9500000000000001, 19.95, 0.0, 1.0)
```



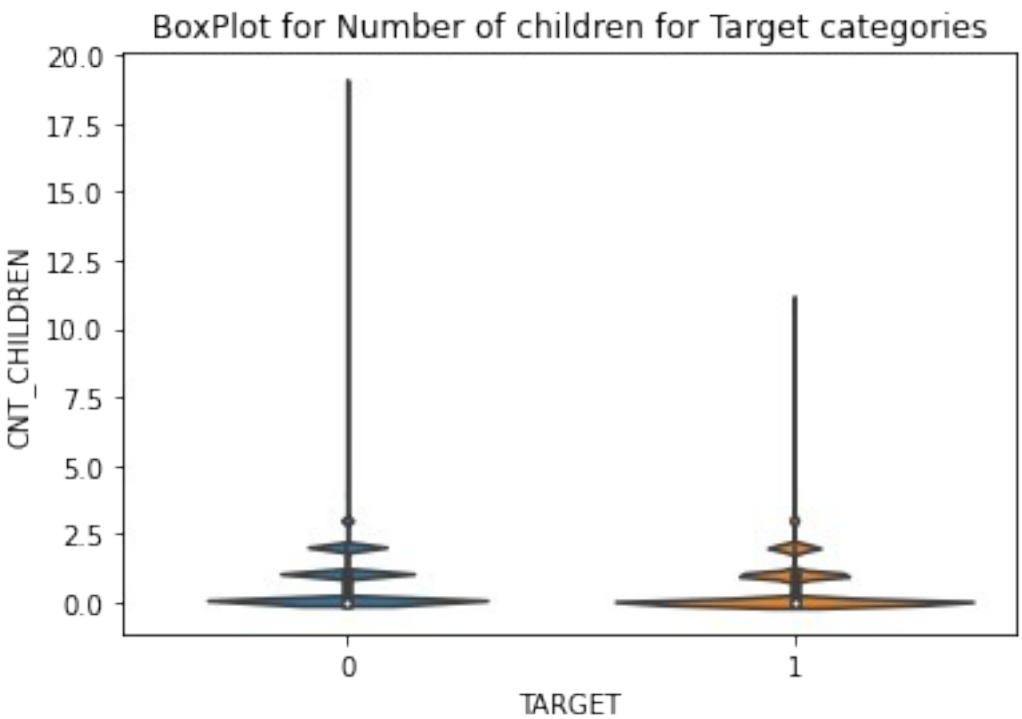

```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="CNT_CHILDREN", data=df_train)
plt.title("BoxPlot for Number of children for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for Number of children for Target categories')
```



```
#lets see the box plot
ax = sns.violinplot(x="TARGET", y="CNT_CHILDREN", data=df_train)
plt.title("BoxPlot for Number of children for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for Number of children for Target categories')
```




```
#Both are similar ,so its raerely matters

#60 % of client having re payment problem are having 1 or zero
kids,more than that mostly no defaulters
```

AMT_INCOME_TOTAL

Income of the client

```
pd.set_option('float_format', '{:f}'.format)
```

```
df_train['AMT_INCOME_TOTAL'].describe()
```

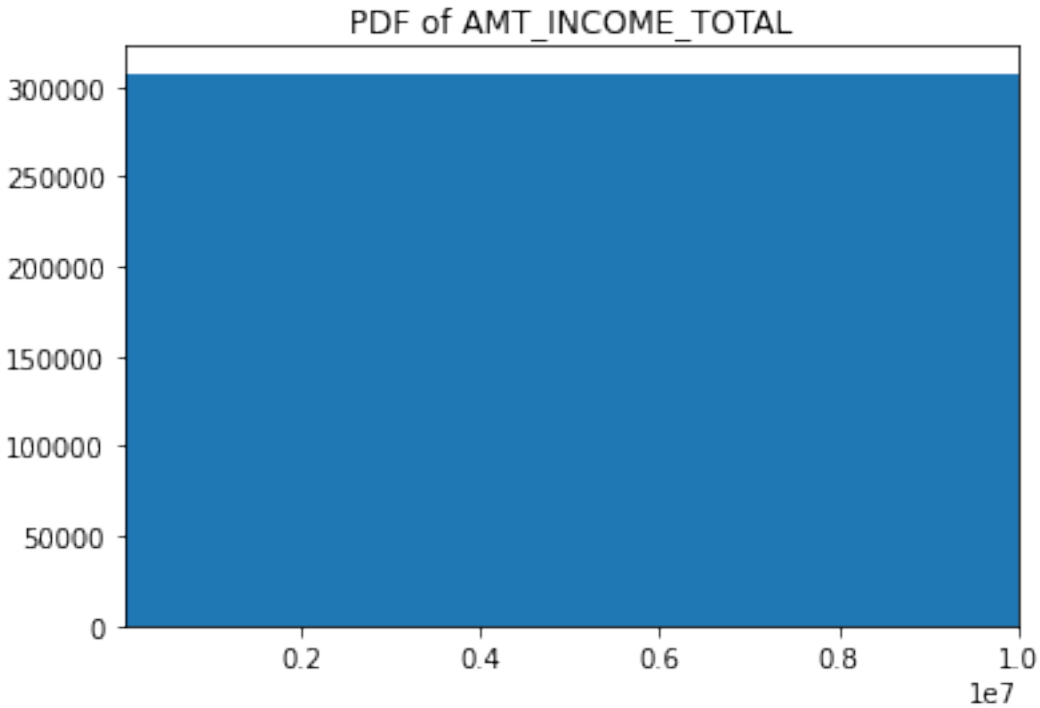
```
count      307511.000000
mean       168797.919297
std        237123.146279
min         25650.000000
25%        112500.000000
50%        147150.000000
75%        202500.000000
max     117000000.000000
Name: AMT_INCOME_TOTAL, dtype: float64
```

```
df_train['AMT_INCOME_TOTAL'].isna().sum()
#No NA values
```

```
0
```

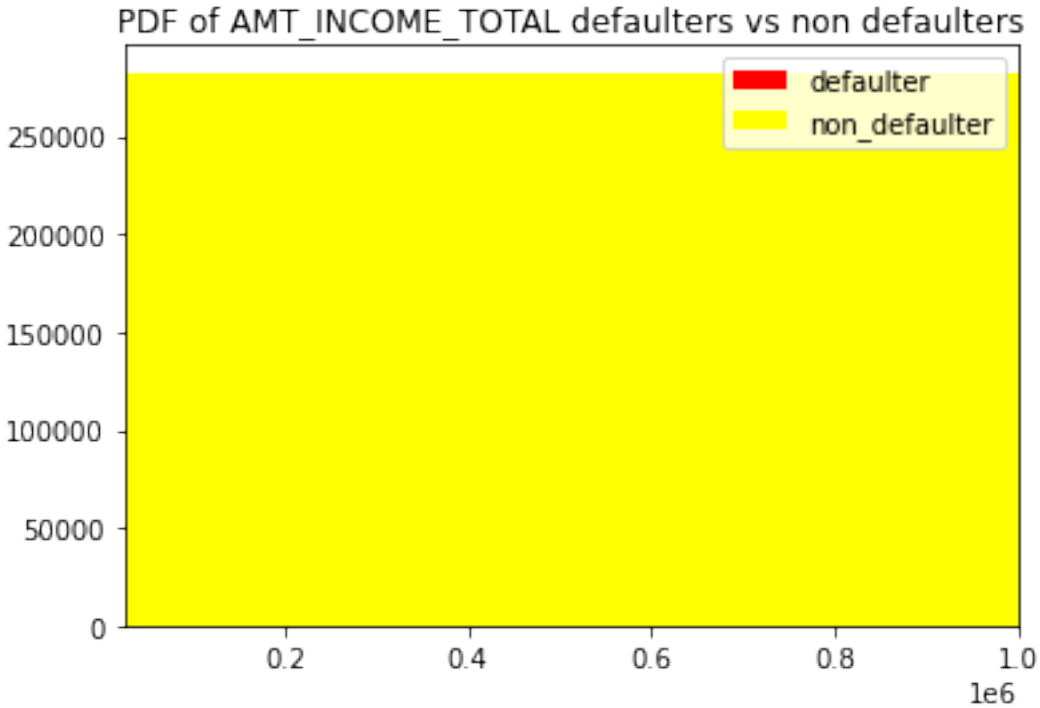
```
#lets see its PDF and CDF

plt.hist(df_train['AMT_INCOME_TOTAL'])
plt.xlim([25000,10000000])
plt.title("PDF of AMT_INCOME_TOTAL")
```

```
Text(0.5, 1.0, 'PDF of AMT_INCOME_TOTAL')
```



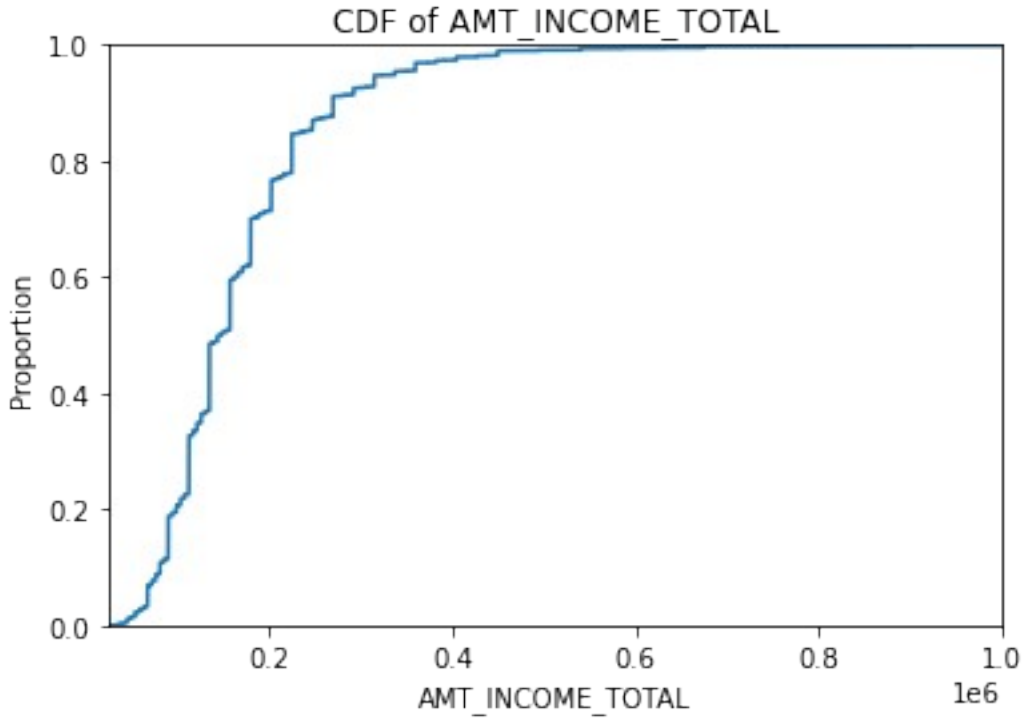

```
plt.hist(df_defaulter['AMT_INCOME_TOTAL'],label='defaulter',color='red
')
plt.hist(df_non_defaulter['AMT_INCOME_TOTAL'],label='non_defaulter',co
lor='yellow')
plt.xlim([25000,1000000])
plt.title("PDF of AMT_INCOME_TOTAL defaulters vs non defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e23696a280>
```




```
#Its seems to be uniform distribution

sns.ecdfplot(df_train['AMT_INCOME_TOTAL'])
plt.xlim([25000,1000000])
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of AMT_INCOME_TOTAL")
```

```
Text(0.5, 1.0, 'CDF of AMT_INCOME_TOTAL')
```




```
np.quantile(df_train['AMT_INCOME_TOTAL'],0.99)
```

```
472500.0
```

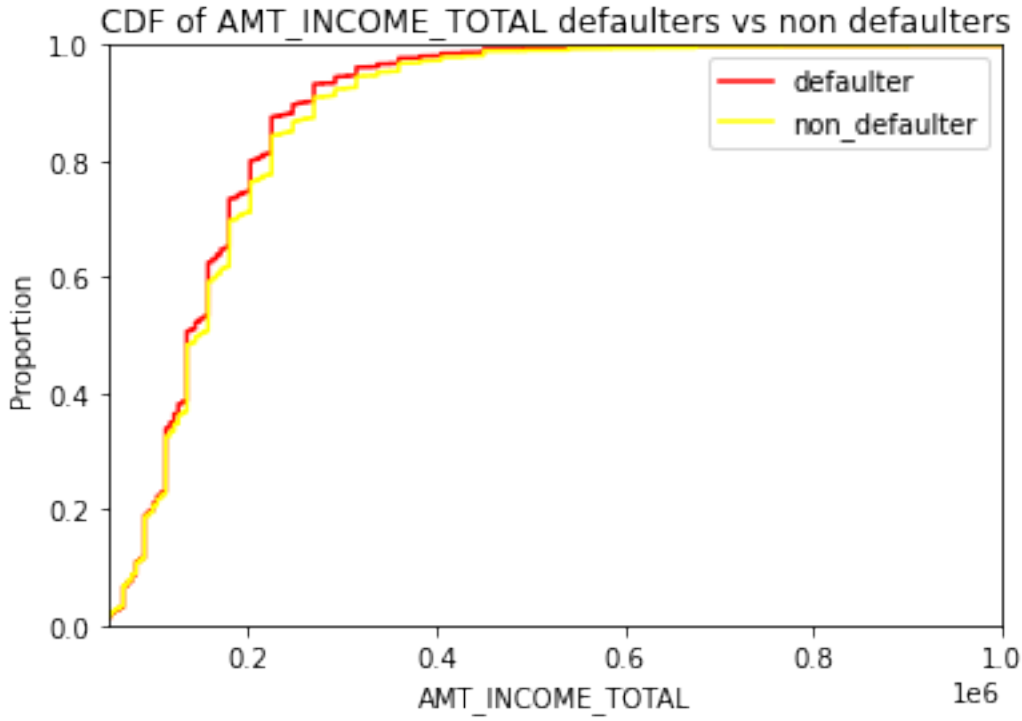
```
np.quantile(df_train['AMT_INCOME_TOTAL'],0.80)
```

```
225000.0
```

- 99% client seems to have income lesser than 472000
- 80% client seems to have income lesser than 225000.0

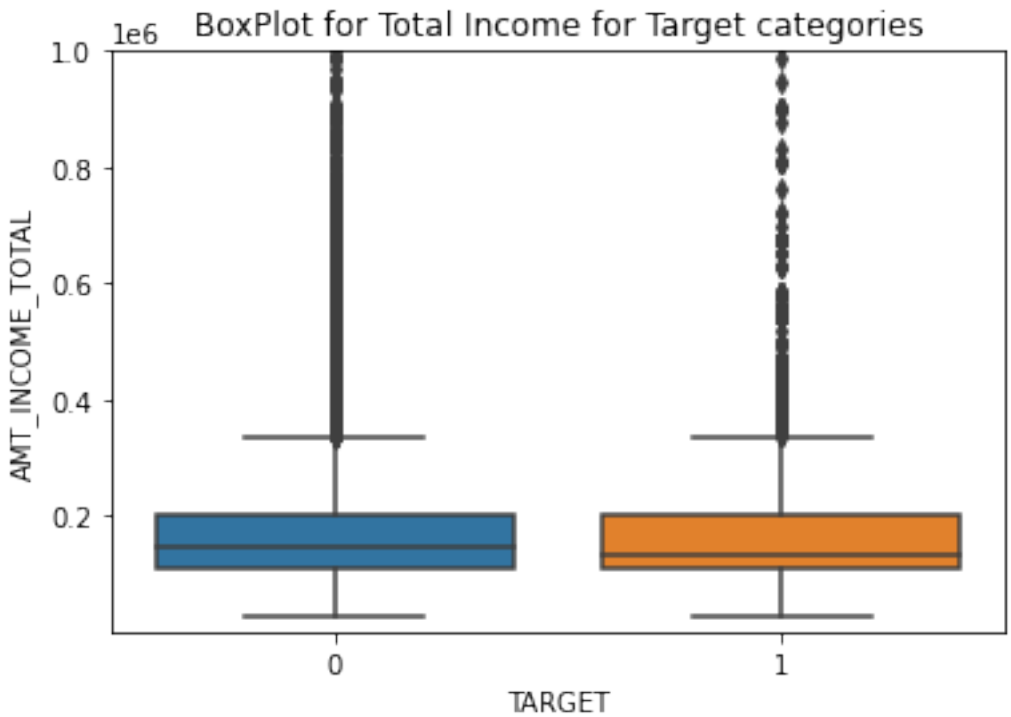
```
sns.ecdfplot(df_defaulter['AMT_INCOME_TOTAL'],label='defaulter',color=
'red')
sns.ecdfplot(df_non_defaulter['AMT_INCOME_TOTAL'],label='non_defaulter
',color='yellow')
plt.xlim([52000,1000000])
plt.title("CDF of AMT_INCOME_TOTAL defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1e24b4fabb0>
```



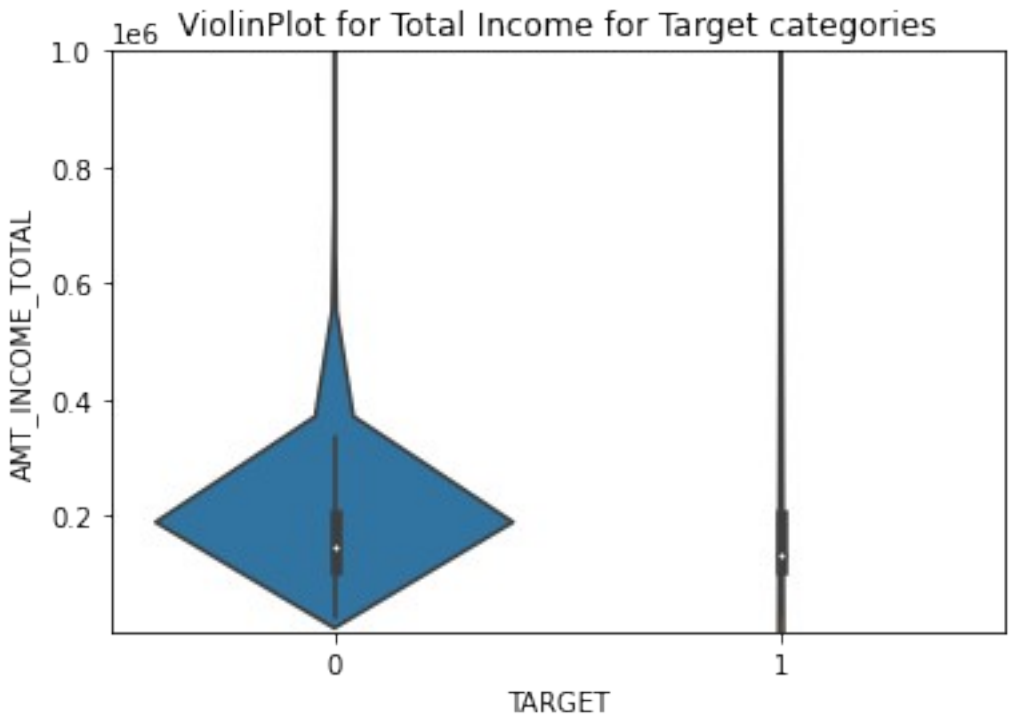

```
#With this plot defaulters seems to have slighly low income but
distribution is almost same

#lets see the box plot
ax = sns.boxplot(x="TARGET", y="AMT_INCOME_TOTAL", data=df_train)
plt.ylim([1000,1000000])
plt.title("BoxPlot for Total Income for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for Total Income for Target categories')
```




```
#seems to have similar dustribution for both categores

#Violin Plot
ax = sns.violinplot(x="TARGET", y="AMT_INCOME_TOTAL", data=df_train)
plt.ylim([1000,1000000])
plt.title("ViolinPlot for Total Income for Target categories")
```

```
Text(0.5, 1.0, 'ViolinPlot for Total Income for Target categories')
```




```
#its okay.. because number of defaults are very less than compare to
number of non defaulters

#top 10 correlated column with AMT_INCOME_TOTAL
df_corr[['AMT_INCOME_TOTAL']].sort_values('AMT_INCOME_TOTAL',ascending
=False)[:10]
```

```
                        AMT_INCOME_TOTAL
AMT_INCOME_TOTAL                1.000000
AMT_ANNUITY                     0.191657
AMT_GOODS_PRICE                 0.159610
AMT_CREDIT                      0.156870
FLOORSMIN_AVG                   0.139860
FLOORSMIN_MEDI                  0.138489
FLOORSMIN_MODE                  0.131800
LIVINGAPARTMENTS_AVG            0.106920
LIVINGAPARTMENTS_MEDI           0.104914
LIVINGAPARTMENTS_MODE           0.092991
```

AMT_CREDIT

Credit amount of the loan: Total Loan amount

```
df_train['AMT_CREDIT'].describe()
```

```
count    307511.000000
mean     599025.999706
```

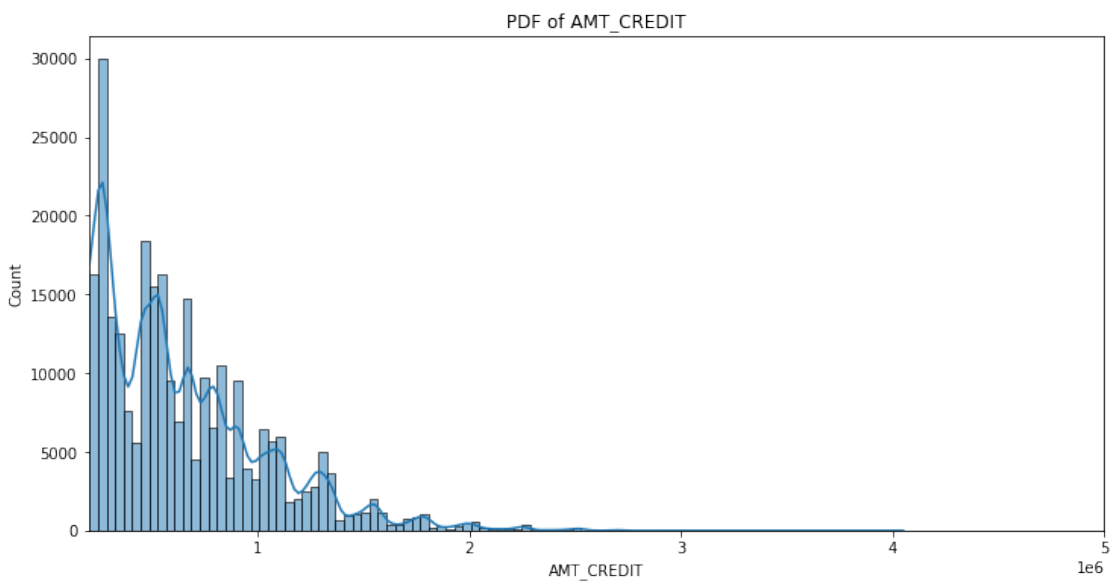
```
std      402490.776996
min       45000.000000
25%      270000.000000
50%      513531.000000
75%      808650.000000
max     4050000.000000
Name: AMT_CREDIT, dtype: float64
```

```
df_train['AMT_CREDIT'].isna().sum()
#No NA values
```

```
0
```

```
#lets see its PDF and CDF
plt.figure(figsize=(12,6))
sns.histplot(df_train['AMT_CREDIT'],bins=100,kde=True)
plt.xlim([200000,5000000])
plt.title("PDF of AMT_CREDIT")
```

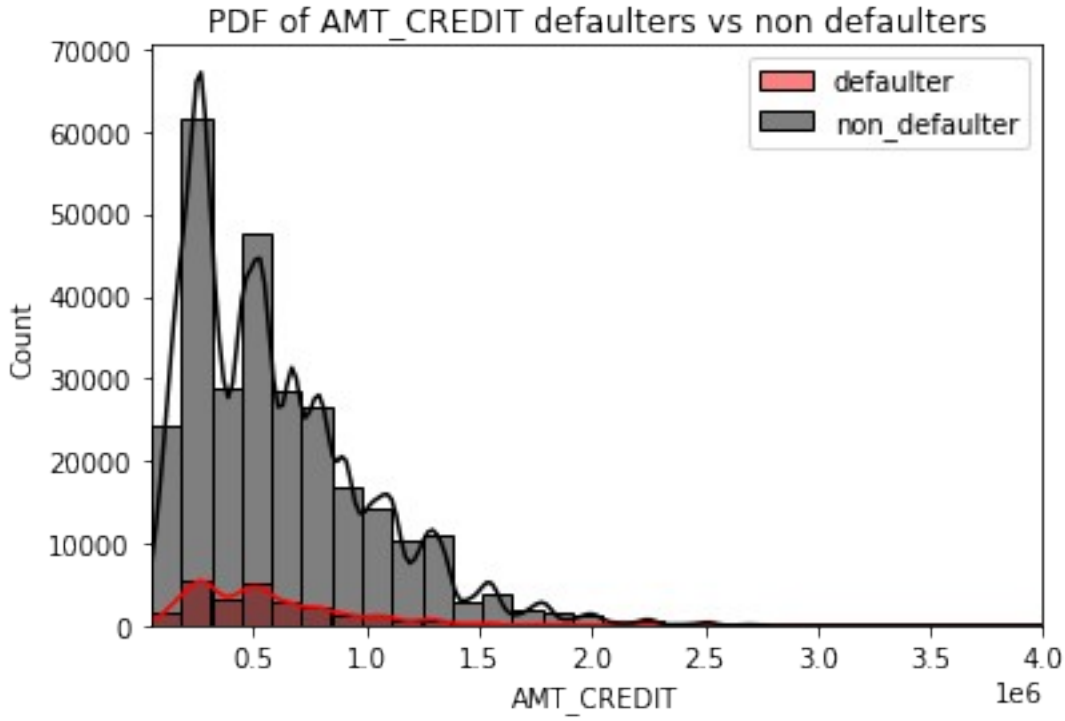
```
Text(0.5, 1.0, 'PDF of AMT_CREDIT')
```




```
# It also follow power law distribution
```

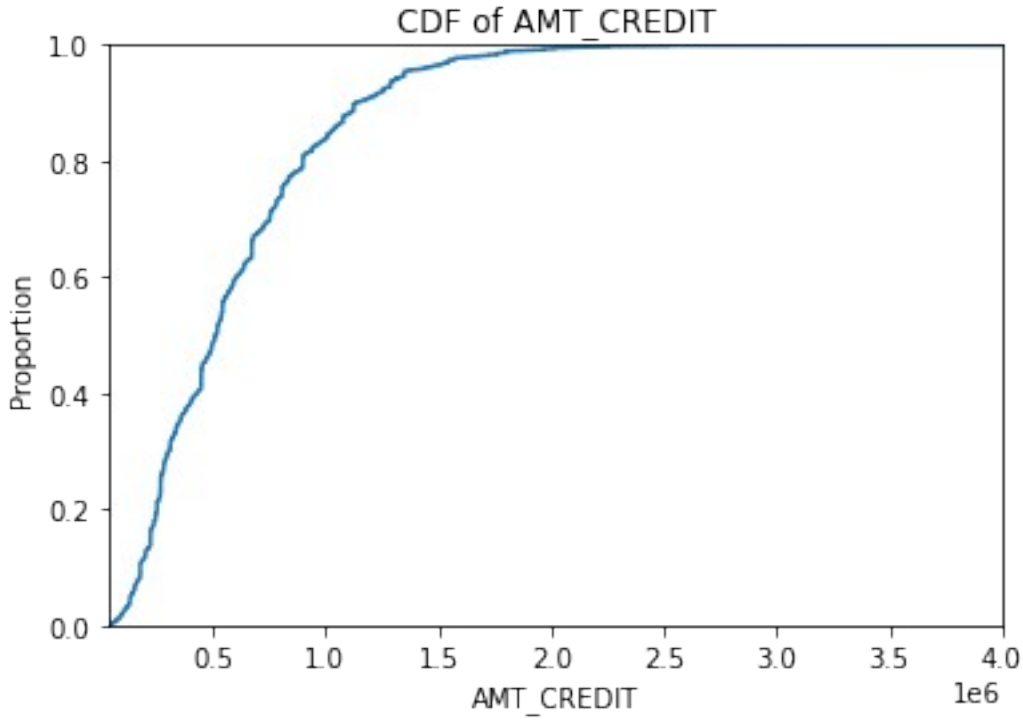
```
sns.histplot(df_defaulter['AMT_CREDIT'],bins=30,kde=True,label='defaul
ter',color='red')
sns.histplot(df_non_defaulter['AMT_CREDIT'],bins=30,kde=True,label='no
n_defaulter',color="#000000")
plt.xlim([40000,4000000])
plt.title("PDF of AMT_CREDIT defaulters vs non defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e24d6938b0>
```




```
sns.ecdfplot(df_train['AMT_CREDIT'])
plt.xlim([40000,4000000])
plt.gca()
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of AMT_CREDIT")
```

```
Text(0.5, 1.0, 'CDF of AMT_CREDIT')
```




```
np.quantile(df_train['AMT_CREDIT'],0.99)
```

```
1854000.0
```

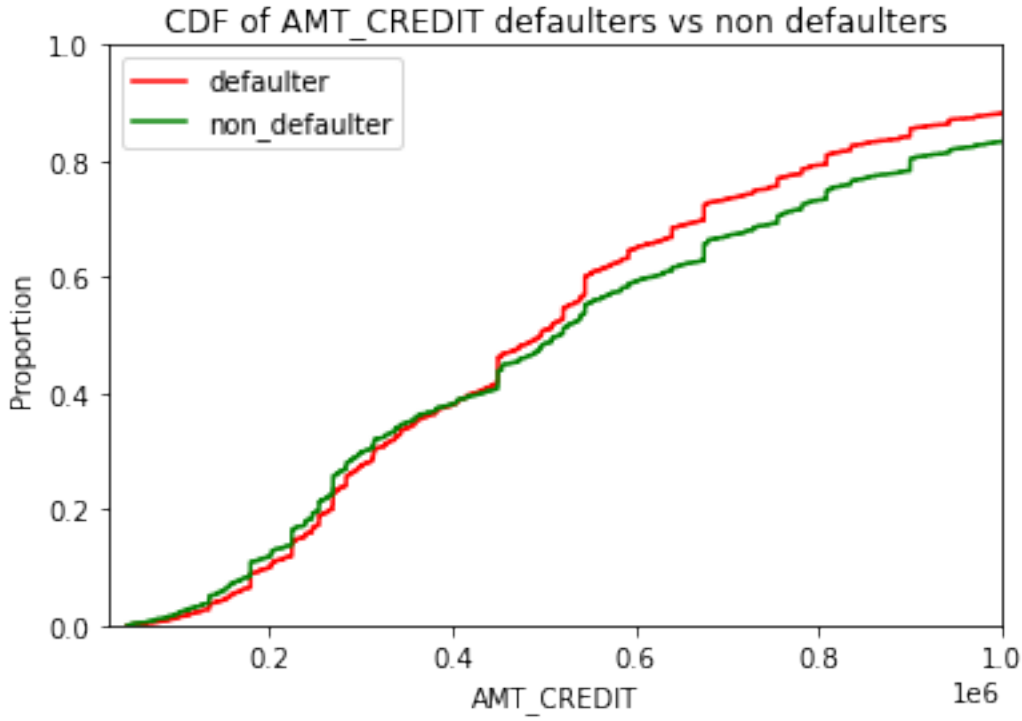
```
np.quantile(df_train['AMT_CREDIT'],0.80)
```

```
900000.0
```

- 99% client seems to have total loan amount lesser than 1854000.0
- 80% client seems to have income lesser than 900000.0

```
sns.ecdfplot(df_defaulter['AMT_CREDIT'],label='defaulter',color='red')
sns.ecdfplot(df_non_defaulter['AMT_CREDIT'],label='non_defaulter',colo
r='green')
plt.xlim([25000,1000000])
plt.title("CDF of AMT_CREDIT defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

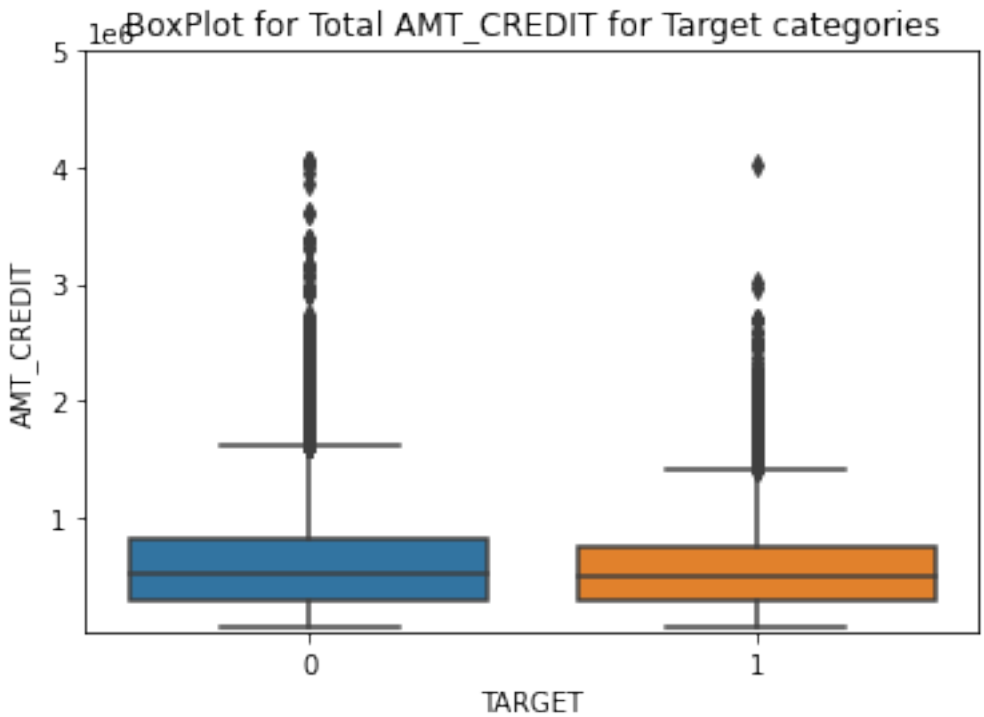
```
<matplotlib.legend.Legend at 0x1e24c733b20>
```



- Below 500000 there are more defaulters after than there are lesser in compare to non defaulters

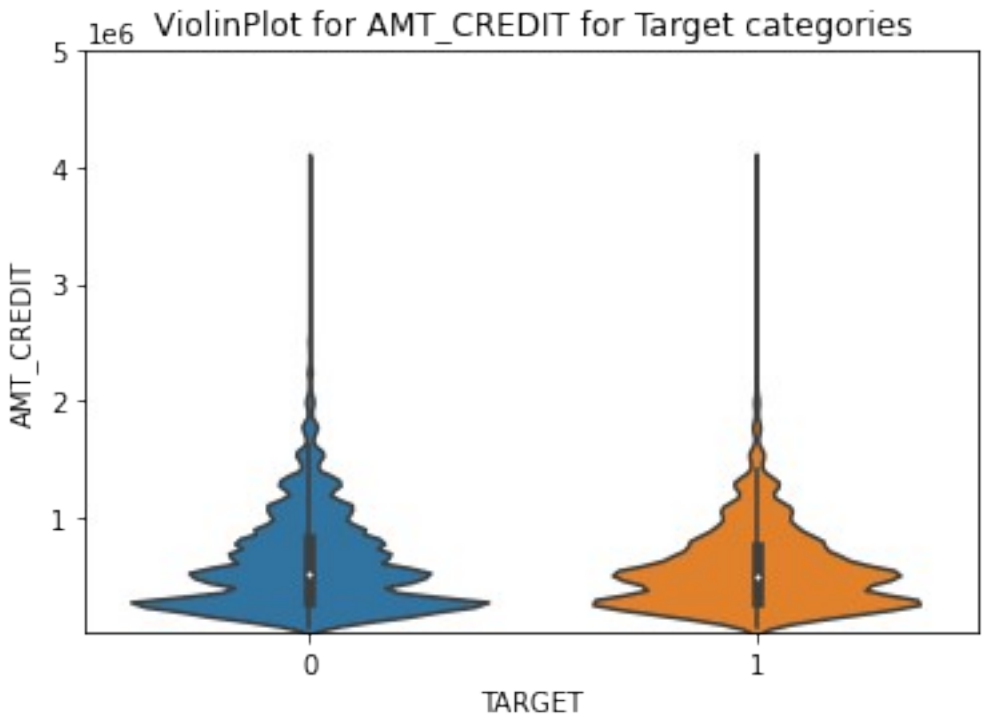
```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="AMT_CREDIT", data=df_train)
plt.ylim([10000,5000000])
plt.title("BoxPlot for Total AMT_CREDIT for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for Total AMT_CREDIT for Target categories')
```




```
#seems to have similar dustribution for both categores,but dafulters
have lesser qunatiles than non defaulters

#Violin Plot
ax = sns.violinplot(x="TARGET", y="AMT_CREDIT", data=df_train)
plt.ylim([10000,5000000])
plt.title("ViolinPlot for AMT_CREDIT for Target categories")
```

```
Text(0.5, 1.0, 'ViolinPlot for AMT_CREDIT for Target categories')
```




```
#its okay.. because number of defaults are very less than compare to
number of non defaulters

#top 10 correlated column with AMT_INCOME_TOTAL
df_corr[['AMT_CREDIT']].sort_values('AMT_CREDIT',ascending=False)[:10]
```

```
                            AMT_CREDIT
AMT_CREDIT                    1.000000
AMT_GOODS_PRICE               0.986968
AMT_ANNUITY                   0.770138
EXT_SOURCE_1                  0.168429
AMT_INCOME_TOTAL              0.156870
EXT_SOURCE_2                  0.131228
FLOORSMAX_AVG                 0.103296
FLOORSMAX_MEDI                0.102770
FLOORSMAX_MODE                0.100418
REGION_POPULATION_RELATIVE    0.099738
```

- AMT_GOODS_PRICE has very high correlation with Total loan amount because loan amount will defintely increase price of the goods for which the loan is given
- AMT_ANNUITY has very high correlation with Loan ANuunity that is anunal amount for loan Repayment willl increase with Total loan amount

AMT_GOODS_PRICE

For consumer loans it is the price of the goods for which the loan is given

```
df_train['AMT_GOODS_PRICE'].describe()
```

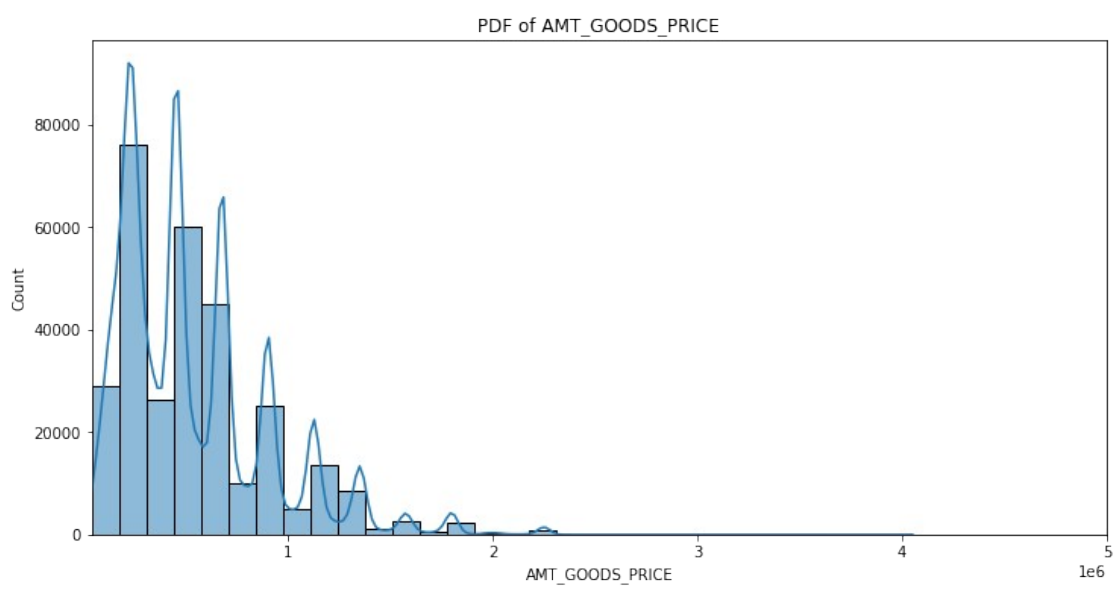
```
count    307233.000000
mean     538396.207429
std      369446.460540
min       40500.000000
25%      238500.000000
50%      450000.000000
75%      679500.000000
max     4050000.000000
Name: AMT_GOODS_PRICE, dtype: float64
```

```
df_train['AMT_GOODS_PRICE'].isna().sum()
# have NA values
```

```
278
```

```
#df_train[df_train['AMT_GOODS_PRICE'].isna()]['AMT_GOODS_PRICE']

#df_train[df_train['AMT_GOODS_PRICE'].isna()]['AMT_CREDIT']

#lets see its PDF and CDF
plt.figure(figsize=(12,6))
sns.histplot(df_train['AMT_GOODS_PRICE'],bins=30,kde=True)
plt.xlim([40000,5000000])
plt.title("PDF of AMT_GOODS_PRICE")
```

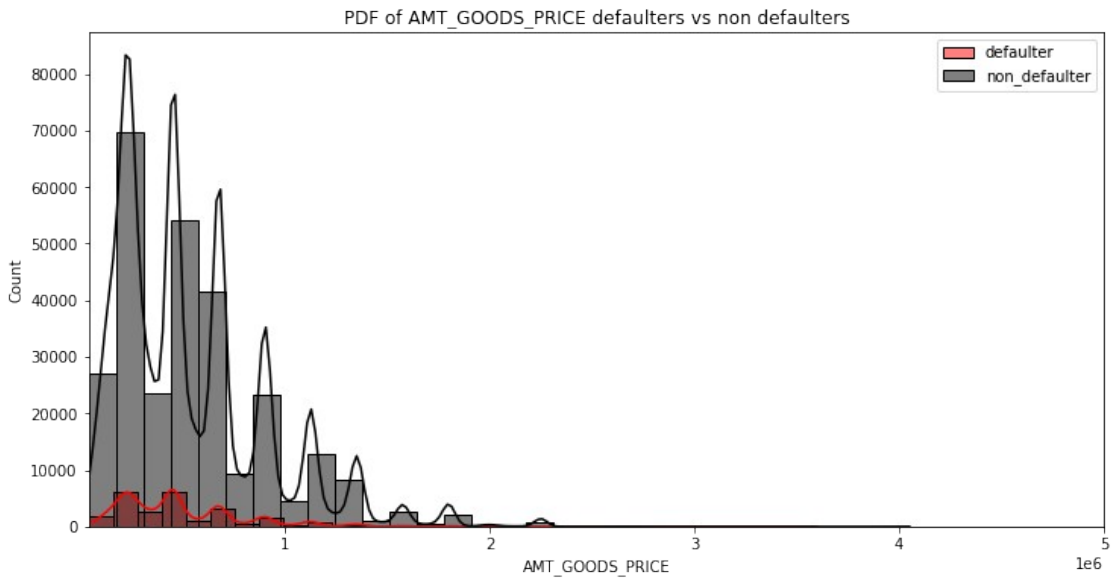
```
Text(0.5, 1.0, 'PDF of AMT_GOODS_PRICE')
```




```
# It also follow power law distribution something like
```

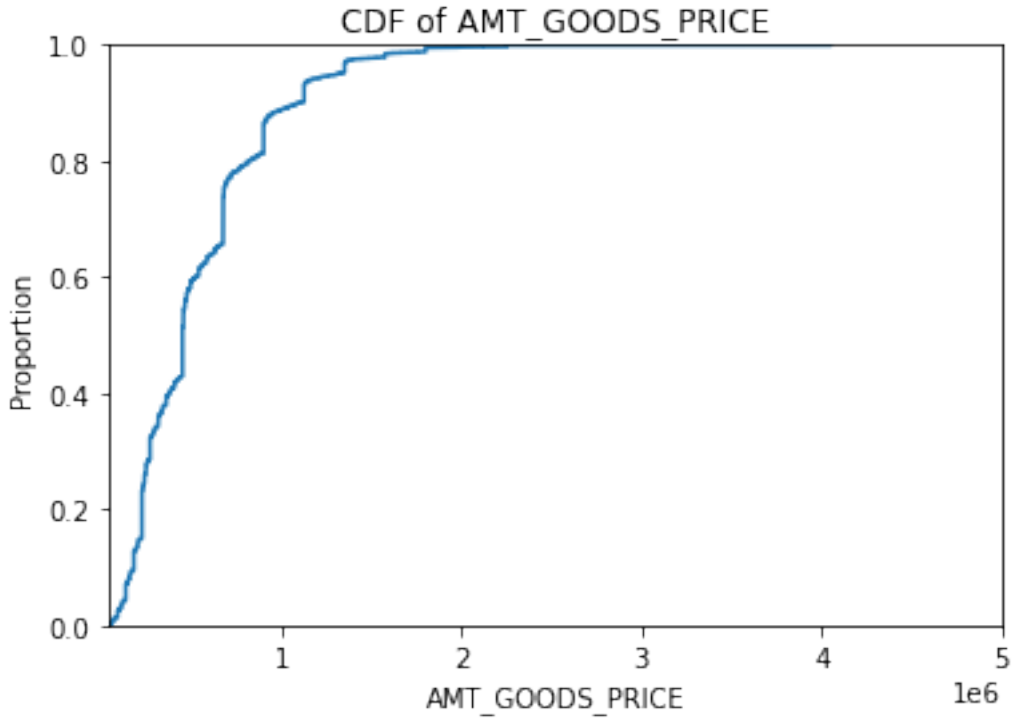
```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['AMT_GOODS_PRICE'],bins=30,kde=True,label='d
efaulter',color='red')
sns.histplot(df_non_defaulter['AMT_GOODS_PRICE'],bins=30,kde=True,labe
l='non_defaulter',color="#000000")
plt.xlim([40000,5000000])
plt.title("PDF of AMT_GOODS_PRICE defaulters vs non defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e24e822b20>
```




```
sns.ecdfplot(df_train['AMT_GOODS_PRICE'])
plt.xlim([40000,5000000])
plt.gca()
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of AMT_GOODS_PRICE")
```

```
Text(0.5, 1.0, 'CDF of AMT_GOODS_PRICE')
```




```
np.quantile(df_train['AMT_GOODS_PRICE'].dropna(),0.99)
```

```
1800000.0
```

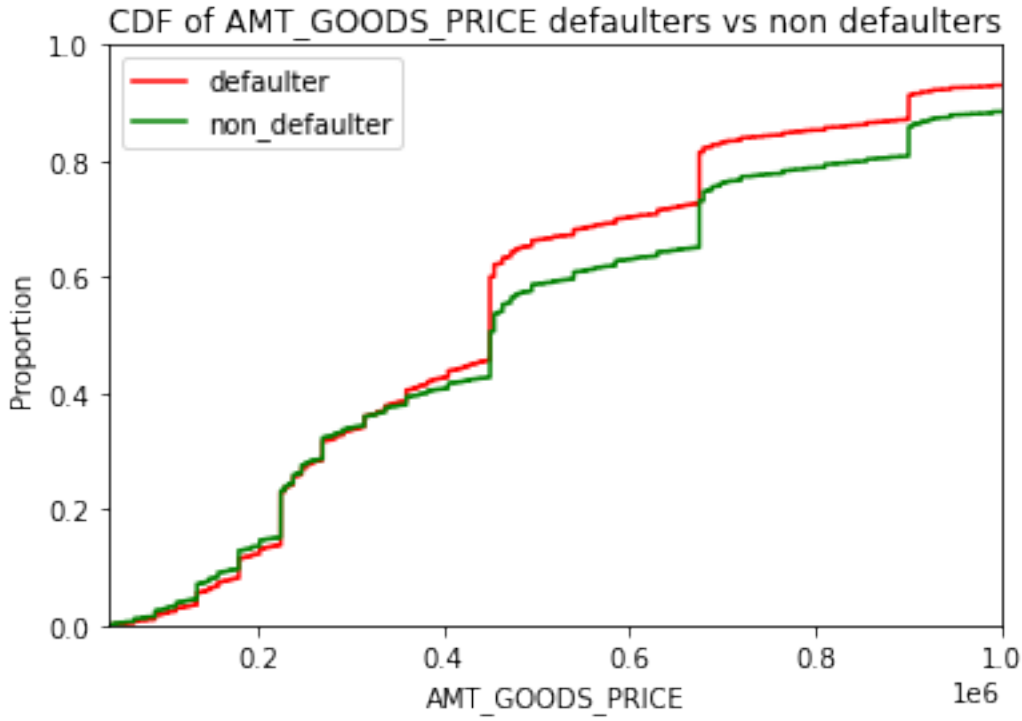
```
np.quantile(df_train['AMT_GOODS_PRICE'].dropna(),0.80)
```

```
814500.0
```

- 99% client seems to have AMT_GOODS_PRICE lesser than 1800000.0
- 80% client seems to have AMT_GOODS_PRICE lesser than 8145000.0

```
sns.ecdfplot(df_defaulter['AMT_GOODS_PRICE'],label='defaulter',color='
red')
sns.ecdfplot(df_non_defaulter['AMT_GOODS_PRICE'],label='non_defaulter'
,color='green')
plt.xlim([40000,1000000])
plt.title("CDF of AMT_GOODS_PRICE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

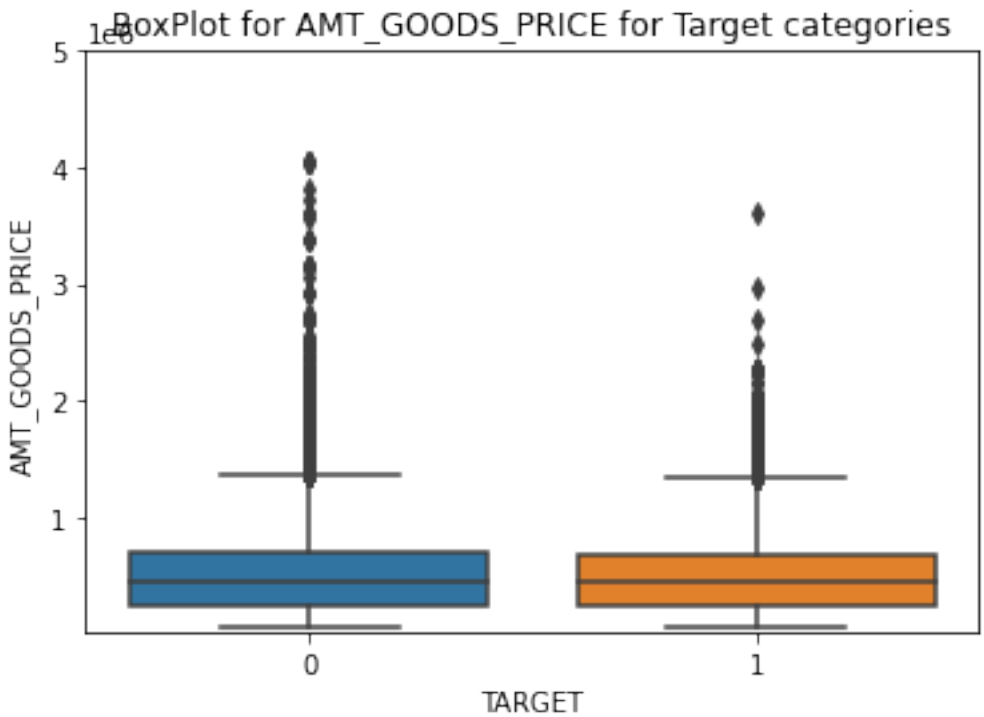
```
<matplotlib.legend.Legend at 0x1e24d3095e0>
```



- Below 500000 there are more defaulters after than there are lesser in compare to non defaulters

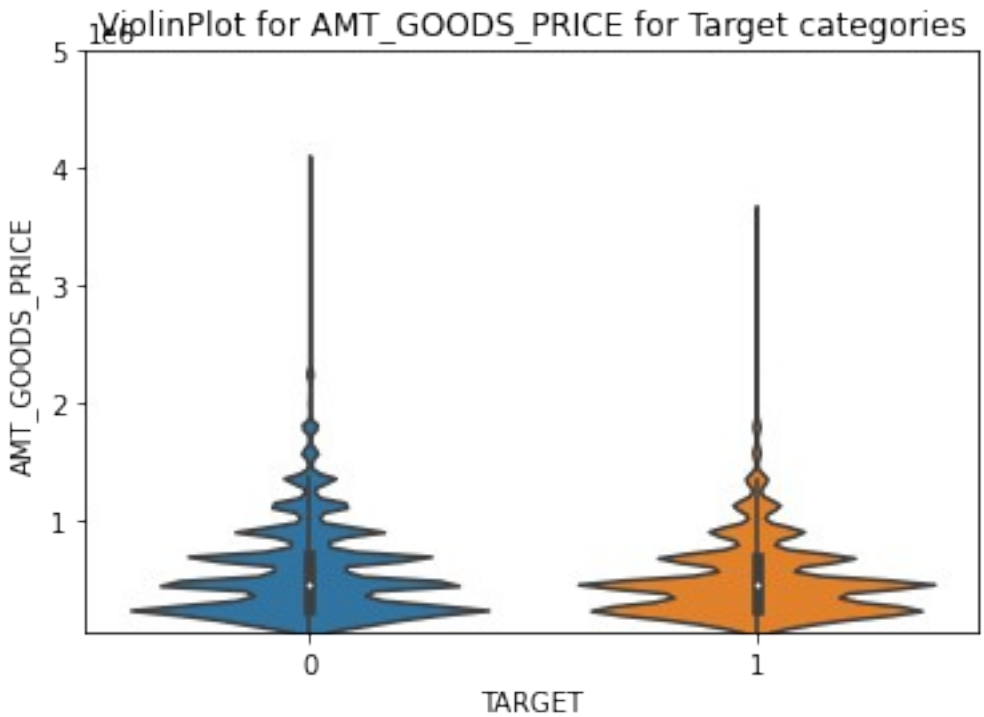
```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="AMT_GOODS_PRICE", data=df_train)
plt.ylim([10000,5000000])
plt.title("BoxPlot for AMT_GOODS_PRICE for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for AMT_GOODS_PRICE for Target categories')
```




```
#seems to have similar dustribution for both categores,but dafulters
have lesser qunatiles than non defaulters

#Violin Plot
ax = sns.violinplot(x="TARGET", y="AMT_GOODS_PRICE", data=df_train)
plt.ylim([40000,5000000])
plt.title("ViolinPlot for AMT_GOODS_PRICE for Target categories")
```

```
Text(0.5, 1.0, 'ViolinPlot for AMT_GOODS_PRICE for Target categories')
```




```
#its okay.. because number of defaults are very less than compare to
number of non defaulters

#top 10 correlated column with AMT_INCOME_TOTAL
df_corr[['AMT_GOODS_PRICE']].sort_values('AMT_GOODS_PRICE',ascending=F
alse)[:10]
```

```
                            AMT_GOODS_PRICE
AMT_GOODS_PRICE                    1.000000
AMT_CREDIT                         0.986968
AMT_ANNUITY                        0.775109
EXT_SOURCE_1                       0.175502
AMT_INCOME_TOTAL                   0.159610
EXT_SOURCE_2                       0.139367
FLOORSMAX_AVG                      0.108512
FLOORSMAX_MEDI                     0.107936
FLOORSMAX_MODE                     0.105532
REGION_POPULATION_RELATIVE         0.103520
```

- AMT_GOODS_PRICE has very high correlation with Total loan amount because loan amount will defintely increase price of the goods for which the loan is given
- AMT_ANNUITY has very high correlation with AMT_GOODS_PRICE that is anunal amount for loan Repayment willl increase with Total loan amount

REGION_POPULATION_RELATIVE

Normalized population of region where client lives (higher number means the client lives in more populated region)

```
df_train['REGION_POPULATION_RELATIVE'].describe()
```

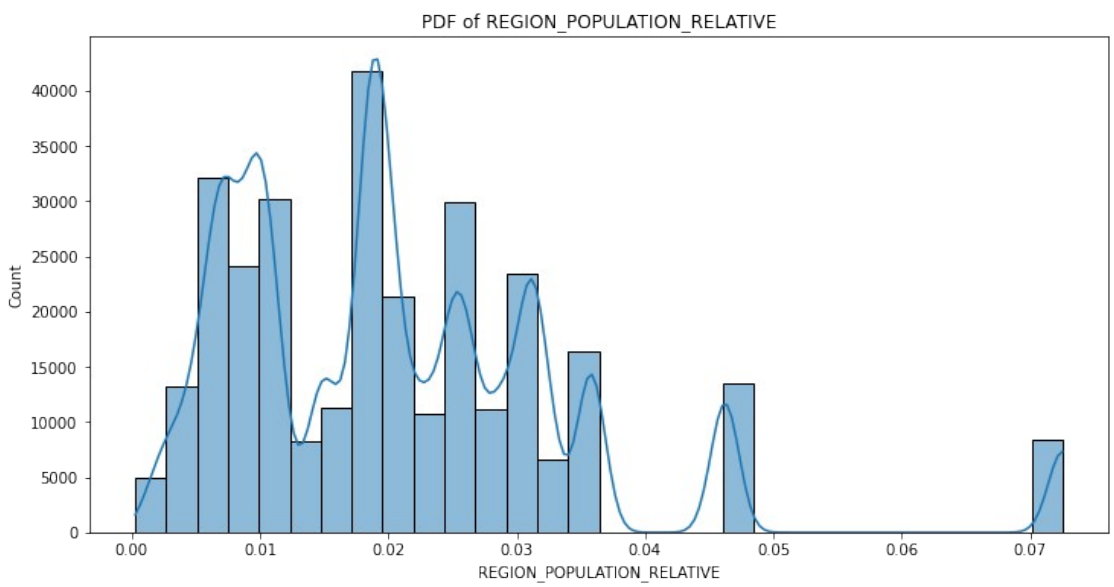
```
count   307511.000000
mean         0.020868
std          0.013831
min          0.000290
25%          0.010006
50%          0.018850
75%          0.028663
max          0.072508
Name: REGION_POPULATION_RELATIVE, dtype: float64
```

```
df_train['REGION_POPULATION_RELATIVE'].isna().sum()
# dont have NA values
```

```
0
```

```
#lets see its PDF and CDF
plt.figure(figsize=(12,6))
sns.histplot(df_train['REGION_POPULATION_RELATIVE'],bins=30,kde=True)
#plt.xlim([40000,5000000])
plt.title("PDF of REGION_POPULATION_RELATIVE")
```

```
Text(0.5, 1.0, 'PDF of REGION_POPULATION_RELATIVE')
```

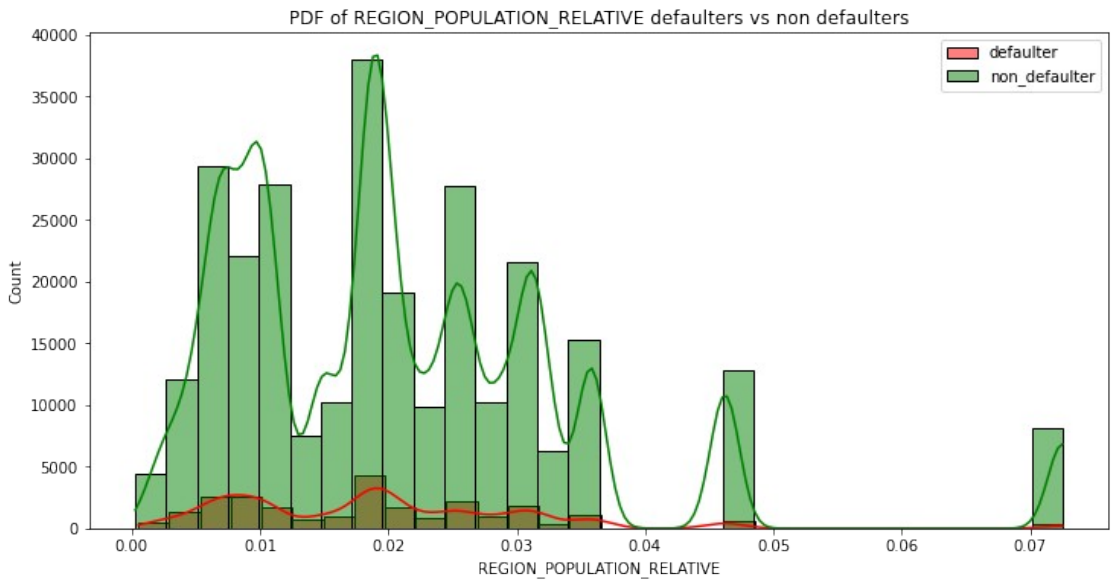



```
# It looks like follows normal but right skewed distribution
```

```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['REGION_POPULATION_RELATIVE'],bins=30,kde=Tr
```

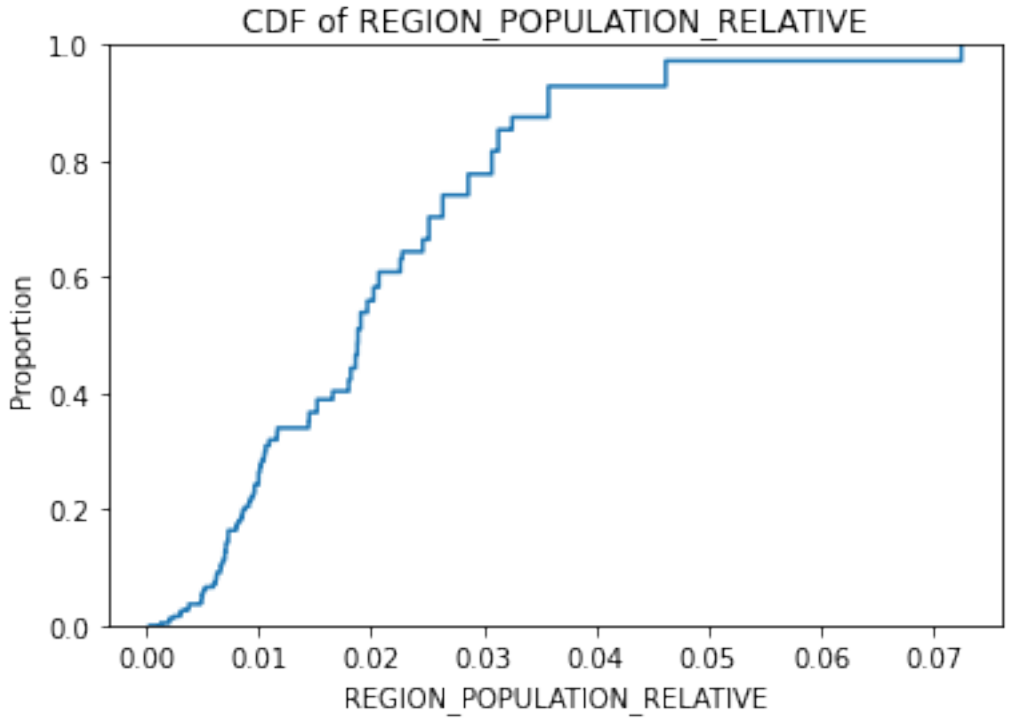
```
ue,label='defaulter',color='red')
sns.histplot(df_non_defaulter['REGION_POPULATION_RELATIVE'],bins=30,kd
e=True,label='non_defaulter',color="green")
#plt.xlim([40000,5000000])
plt.title("PDF of REGION_POPULATION_RELATIVE defaulters vs non
defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e24cb30340>
```




```
sns.ecdfplot(df_train['REGION_POPULATION_RELATIVE'])
#plt.xlim([40000,5000000])
plt.gca()
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of REGION_POPULATION_RELATIVE")
```

```
Text(0.5, 1.0, 'CDF of REGION_POPULATION_RELATIVE')
```




```
np.quantile(df_train['REGION_POPULATION_RELATIVE'].dropna(),0.99)
```

```
0.072508
```

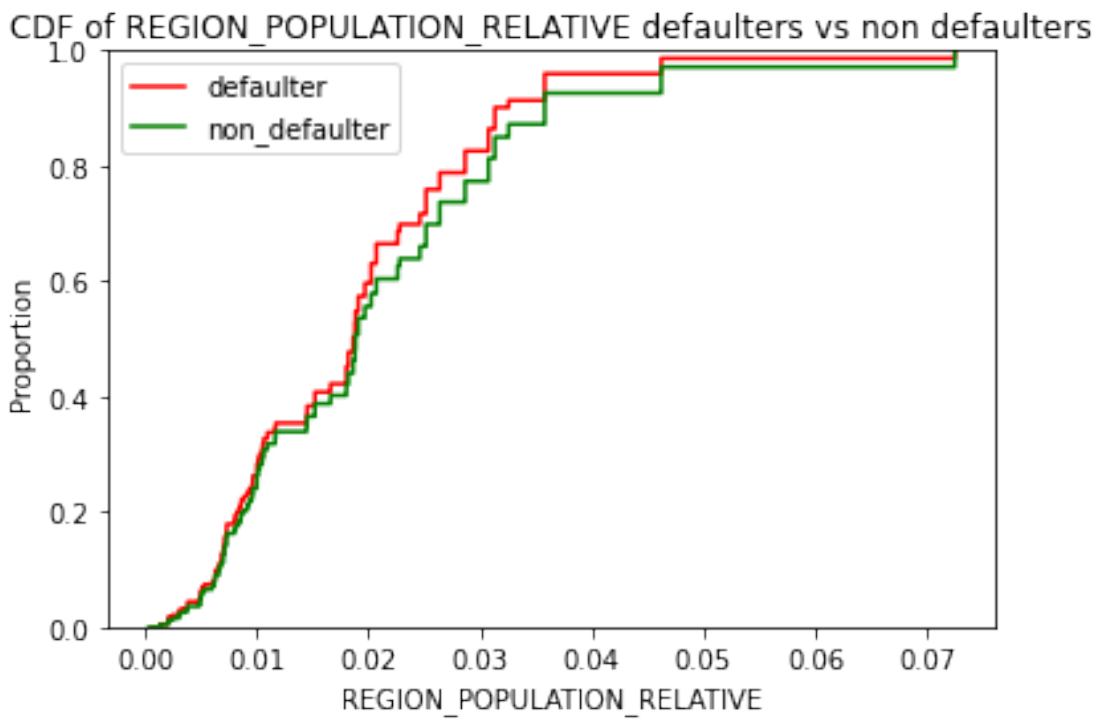
```
np.quantile(df_train['REGION_POPULATION_RELATIVE'].dropna(),0.80)
```

```
0.030755
```

- 99% client seems to have REGION_POPULATION_RELATIVE lesser than 0.07
- 80% client seems to have REGION_POPULATION_RELATIVE lesser than 0.03

```
sns.ecdfplot(df_defaulter['REGION_POPULATION_RELATIVE'],label='default
er',color='red')
sns.ecdfplot(df_non_defaulter['REGION_POPULATION_RELATIVE'],label='non
_defaulter',color='green')
#plt.xlim([40000,1000000])
plt.title("CDF of REGION_POPULATION_RELATIVE defaulters vs non
defaulters")
plt.legend()
#plt.axis('off')
```

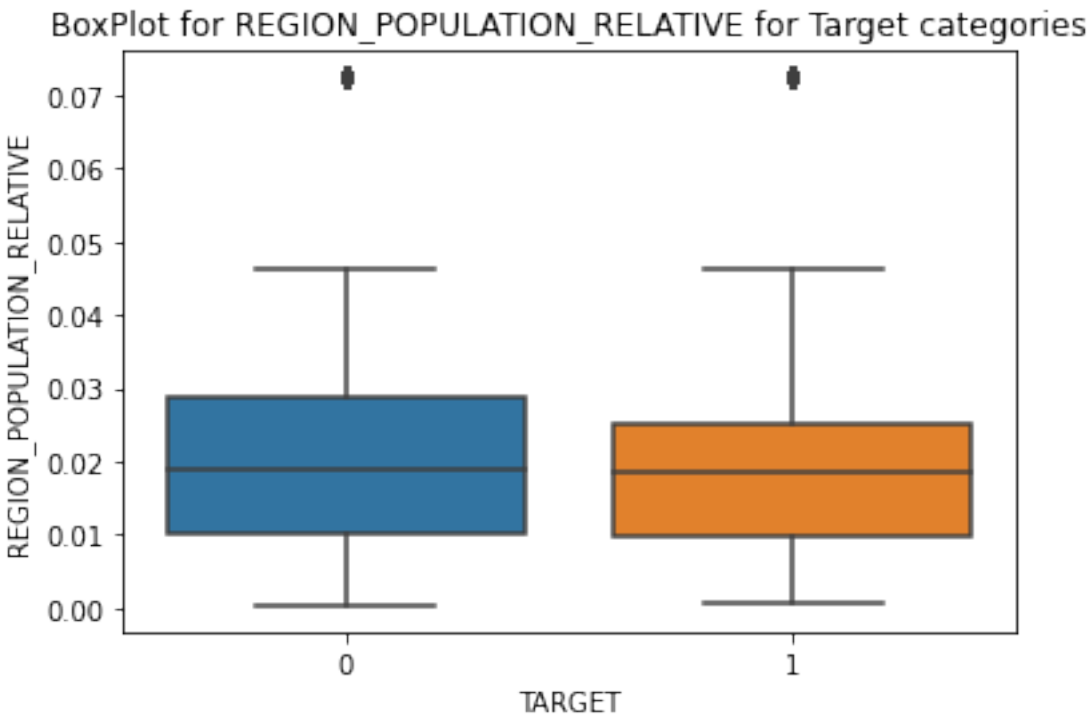
```
<matplotlib.legend.Legend at 0x1e24d852b50>
```



- Above 0.2 polpulation density there are lesser defaulters after than there are lesser in compare to non defaulters

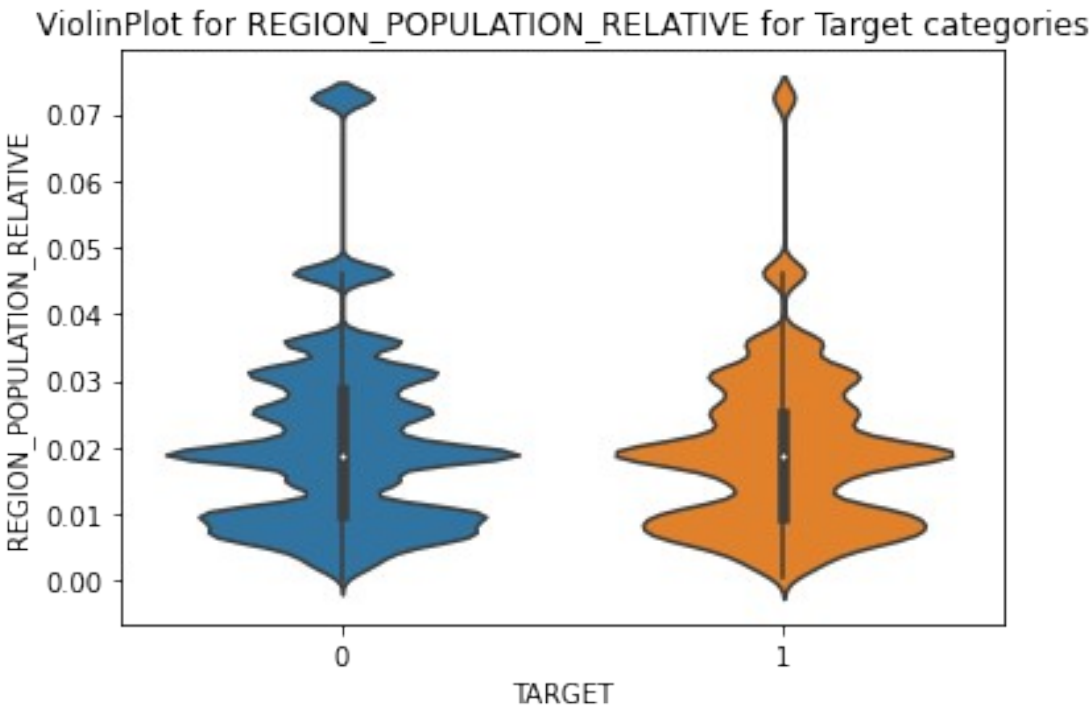
```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="REGION_POPULATION_RELATIVE",
data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for REGION_POPULATION_RELATIVE for Target
categories")
```

```
Text(0.5, 1.0, 'BoxPlot for REGION_POPULATION_RELATIVE for Target
categories')
```




```
#seems to have similar dustribution for both categores,but dafulters
have lesser qunatiles than non defaulters

#Violin Plot
ax = sns.violinplot(x="TARGET", y="REGION_POPULATION_RELATIVE",
data=df_train)
#.ylim([40000,5000000])
plt.title("ViolinPlot for REGION_POPULATION_RELATIVE for Target
categories")
```

```
Text(0.5, 1.0, 'ViolinPlot for REGION_POPULATION_RELATIVE for Target
categories')
```




```
#top 10 correlated column with AMT_INCOME_TOTAL
df_corr[['REGION_POPULATION_RELATIVE']].sort_values('REGION_POPULATION
_RELATIVE',ascending=False)[:10]
```

```
                            REGION_POPULATION_RELATIVE
REGION_POPULATION_RELATIVE                    1.000000
FLOORSMAX_AVG                                 0.322652
FLOORSMAX_MEDI                                0.318319
FLOORSMAX_MODE                                0.303992
FLOORSMIN_AVG                                 0.294349
FLOORSMIN_MEDI                                0.290744
ELEVATORS_AVG                                 0.280685
FLOORSMIN_MODE                                0.275656
ELEVATORS_MEDI                                0.274390
ELEVATORS_MODE                                0.251539
```

DAYS_BIRTH

Client's age in days at the time of application

```
df_train['DAYS_BIRTH'].describe()
```

```
count   307511.000000
mean    -16036.995067
std       4363.988632
min     -25229.000000
25%     -19682.000000
```

```
50%     -15750.000000
75%     -12413.000000
max      -7489.000000
Name: DAYS_BIRTH, dtype: float64
```

```
df_train['DAYS_BIRTH'].isna().sum()
# dont have NA values
```

```
0
```

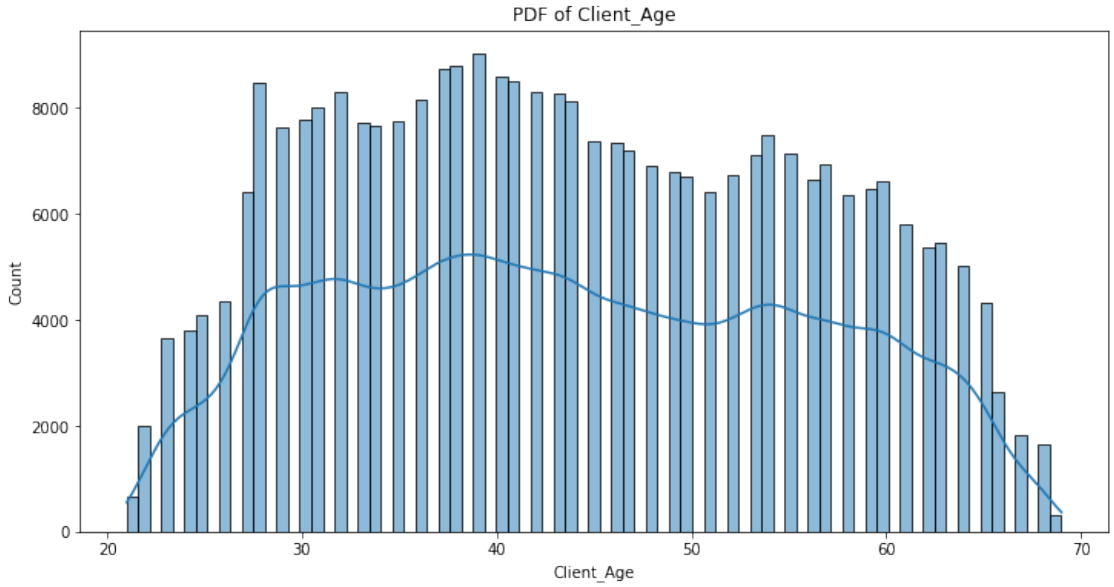
```
#we can turn it into Age in months,so we dont losse all info

df_train['Client_Age']=df_train['DAYS_BIRTH'].apply(lambda x :
int(np.round((-1*x/365))))
df_defaulter['Client_Age']=df_defaulter['DAYS_BIRTH'].apply(lambda x :
int(np.round((-1*x/365))))
df_non_defaulter['Client_Age']=df_non_defaulter['DAYS_BIRTH'].apply(la
mbda x : int(np.round((-1*x/365))))
df_test['Client_Age']=df_test['DAYS_BIRTH'].apply(lambda x :
int(np.round((-1*x/365))))
```

```
df_train['Client_Age'].describe()
```

```
count   307511.000000
mean        43.938646
std         11.964047
min         21.000000
25%         34.000000
50%         43.000000
75%         54.000000
max         69.000000
Name: Client_Age, dtype: float64
```

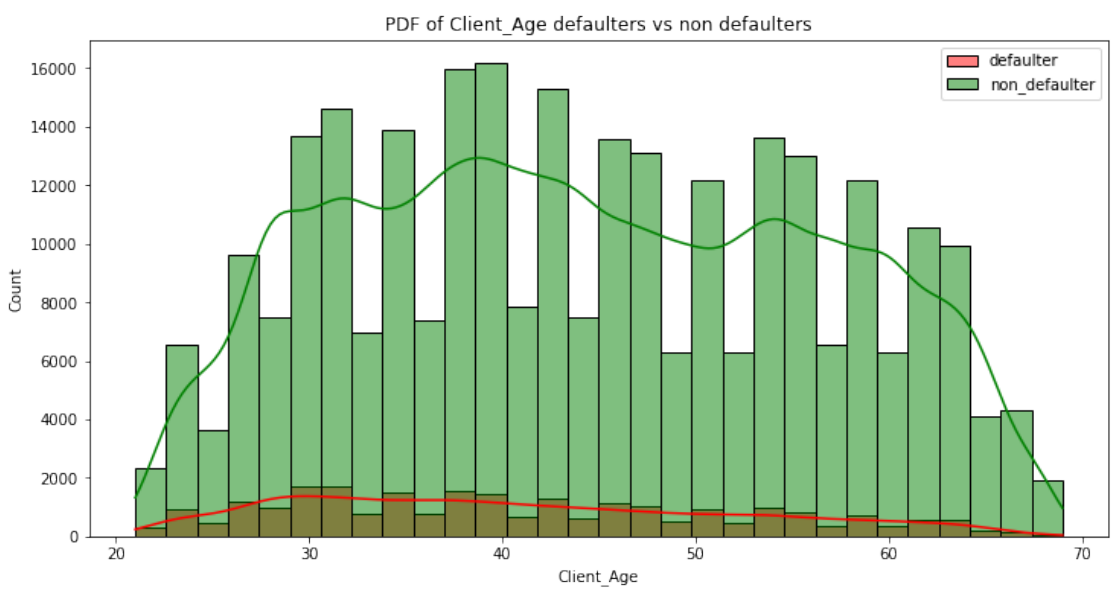
```
#lets see its PDF and CDF
plt.figure(figsize=(12,6))
sns.histplot(df_train['Client_Age'],kde=True)
#plt.xlim([40000,5000000])
plt.title("PDF of Client_Age")
```

```
Text(0.5, 1.0, 'PDF of Client_Age')
```




```
# It looks like follows normal but right skewed distribution

plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['Client_Age'],bins=30,kde=True,label='defaul
ter',color='red')
sns.histplot(df_non_defaulter['Client_Age'],bins=30,kde=True,label='no
n_defaulter',color="green")
#plt.xlim([40000,5000000])
plt.title("PDF of Client_Age defaulters vs non defaulters")
plt.legend()
```

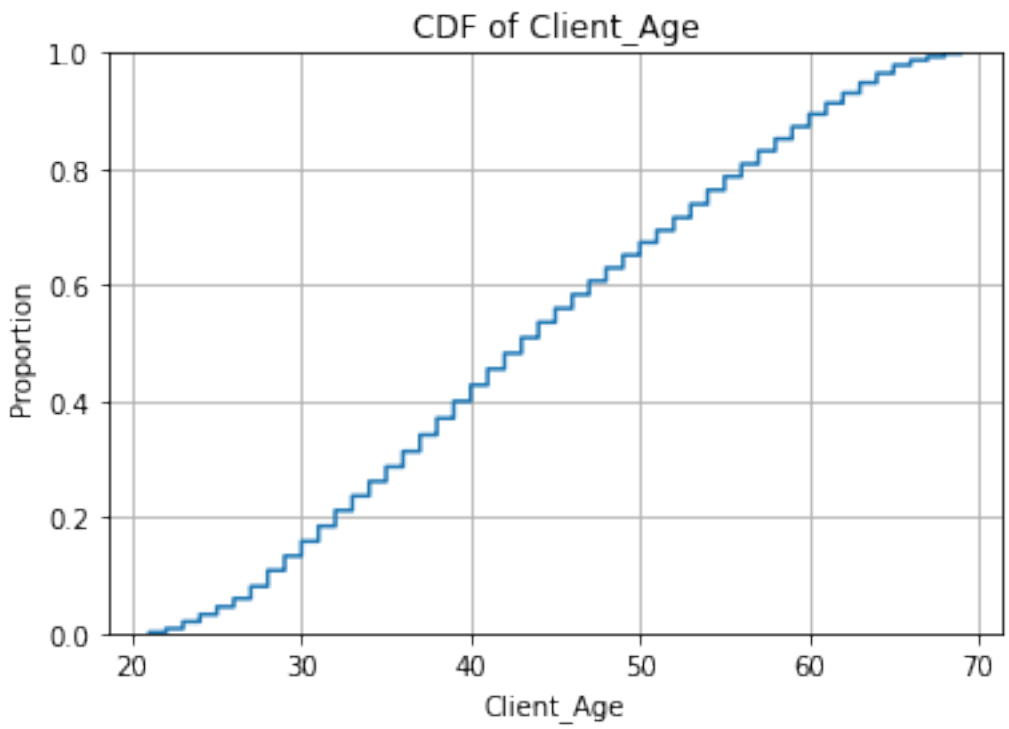
```
<matplotlib.legend.Legend at 0x1e24e307100>
```




```
sns.ecdfplot(df_train['Client_Age'])
#plt.xlim([40000,5000000])
```

```
plt.grid()
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of Client_Age")
```

```
Text(0.5, 1.0, 'CDF of Client_Age')
```




```
np.quantile(df_train['Client_Age'].dropna(),0.99)
```

```
67.0
```

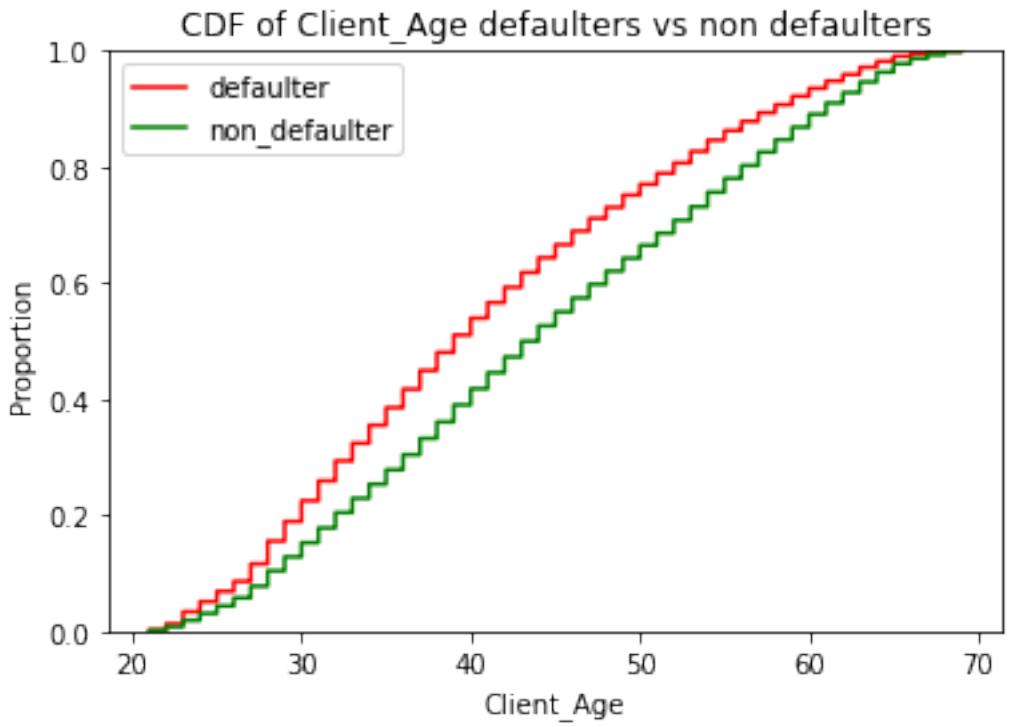
```
np.quantile(df_train['Client_Age'].dropna(),0.80)
```

```
56.0
```

- 99% client seems to have Client_Age lesser than 67 years
- 80% client seems to have Client_Age lesser than 56

```
sns.ecdfplot(df_defaulter['Client_Age'],label='defaulter',color='red')
sns.ecdfplot(df_non_defaulter['Client_Age'],label='non_defaulter',colo
r='green')
#plt.xlim([40000,1000000])
plt.title("CDF of Client_Age defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

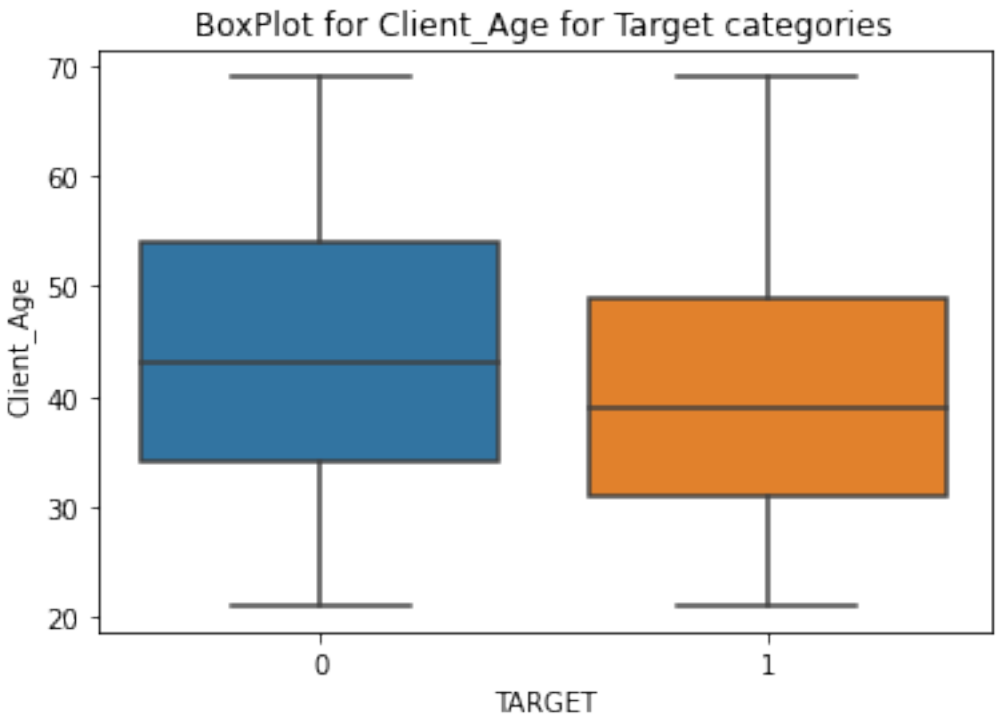
```
<matplotlib.legend.Legend at 0x1e24e01ee20>
```



- Avergae Age of defaulters are lesser in compare to non defaulters

```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="Client_Age", data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for Client_Age for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for Client_Age for Target categories')
```




```
#  deafaulters are supposed to be averagely younger than Non
defaulters

#Violin Plot
ax = sns.violinplot(x="TARGET", y="Client_Age", data=df_train)
#.ylim([40000,5000000])
plt.title("ViolinPlot for Client_Age for Target categories")
```

```
Text(0.5, 1.0, 'ViolinPlot for Client_Age for Target categories')
```

```
#top 10 correlated column with AMT_INCOME_TOTAL
df_train.corr()
[['Client_Age']].sort_values('Client_Age',ascending=False)[:10]
```

```
                              Client_Age
Client_Age                      1.000000
DAYS_EMPLOYED                   0.615569
EXT_SOURCE_1                    0.600333
FLAG_DOCUMENT_6                 0.406997
EXT_SOURCE_3                    0.205492
EXT_SOURCE_2                    0.091993
AMT_REQ_CREDIT_BUREAU_YEAR      0.072011
AMT_CREDIT                      0.055482
AMT_GOODS_PRICE                 0.053473
FLAG_PHONE                      0.042346
```

- Client_Age has very high correlation with DAYS_EMPLOYED because it defintely increase with Age of clients
- EXT_SOURCE_1 has also high correlation with Client_Age because as age increase ,Its not tendency to have a extra income

DAYS_EMPLOYED

How many days before the application the person started current employment

```
df_train['DAYS_EMPLOYED'].describe()
```

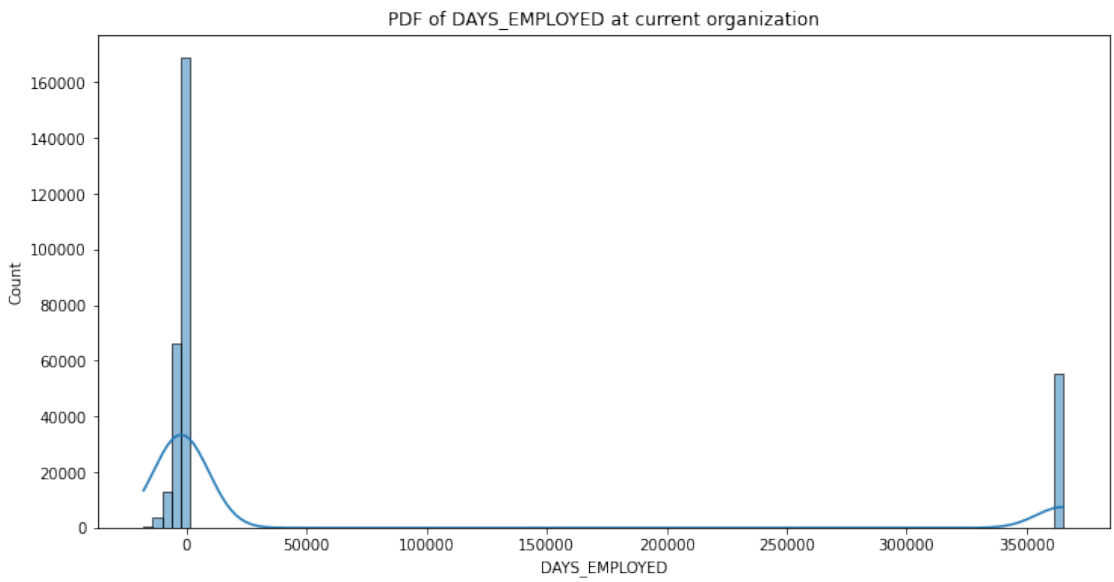
```
count   307511.000000
mean     63815.045904
std     141275.766519
min     -17912.000000
25%      -2760.000000
50%      -1213.000000
75%       -289.000000
max     365243.000000
Name: DAYS_EMPLOYED, dtype: float64
```

```
df_train['DAYS_EMPLOYED'].isna().sum()
# dont have NA values
```

```
0
```

```
#lets see its PDF and CDF
plt.figure(figsize=(12,6))
sns.histplot(df_train['DAYS_EMPLOYED'],bins=100,kde=True)
#plt.xlim([40000,5000000])
plt.title("PDF of DAYS_EMPLOYED at current organization")
```

```
Text(0.5, 1.0, 'PDF of DAYS_EMPLOYED at current organization')
```




```
# It looks like mostly are negative values but some are postive
ones,lets see positive ones
```

```
len(df_train[df_train['DAYS_EMPLOYED']>0])
```

```
55374
```

```
df_train[df_train['DAYS_EMPLOYED']>0]['TARGET'].value_counts()
```

```
0    52384
1     2990
Name: TARGET, dtype: int64
```

```
df_train[df_train['DAYS_EMPLOYED']>0]['CODE_GENDER'].value_counts()
```

```
F    45271
M    10103
Name: CODE_GENDER, dtype: int64
```

```
#So its normal because they have valid entries,lets see distribution
for non negative
```

```
df_train[df_train['DAYS_EMPLOYED']>0]['DAYS_EMPLOYED'].describe()
```

```
count    55374.000000
mean    365243.000000
std          0.000000
min     365243.000000
25%     365243.000000
50%     365243.000000
75%     365243.000000
max     365243.000000
Name: DAYS_EMPLOYED, dtype: float64
```

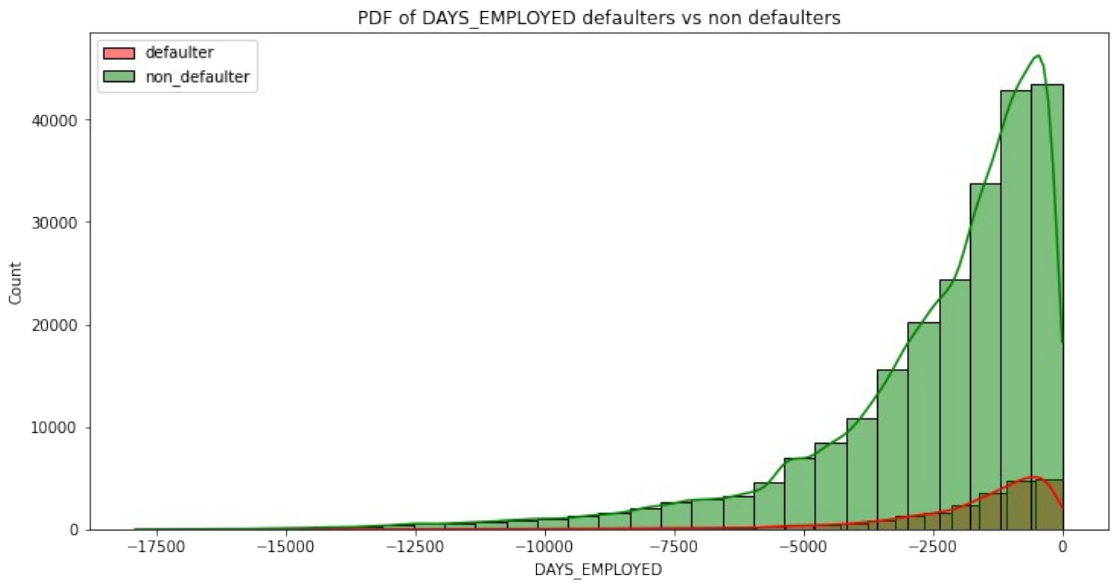
```
df_train[df_train['DAYS_EMPLOYED']<0]['DAYS_EMPLOYED'].describe()
```

```
count   252135.000000
mean     -2384.188236
std       2338.359795
min     -17912.000000
25%      -3175.000000
50%      -1648.000000
75%       -767.000000
max         -1.000000
Name: DAYS_EMPLOYED, dtype: float64
```

- Wow this looks like a data entry error because no body can have 110 years of employment and non employment
- Also all values are same
- i think we can impute those value in relation with Clients_Age

```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter[df_defaulter['DAYS_EMPLOYED']<0]
['DAYS_EMPLOYED'],bins=30,kde=True,label='defaulter',color='red')
sns.histplot(df_non_defaulter[df_non_defaulter['DAYS_EMPLOYED']<0]
['DAYS_EMPLOYED'],bins=30,kde=True,label='non_defaulter',color="green"
)
#plt.xlim([40000,5000000])
plt.title("PDF of DAYS_EMPLOYED defaulters vs non defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e24cc31910>
```




```
sns.ecdfplot(-1*df_train[df_train['DAYS_EMPLOYED']<0]
['DAYS_EMPLOYED'])
#plt.xlim([40000,5000000])
plt.grid()
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of client's DAYS_EMPLOYED")
```

```
Text(0.5, 1.0, "CDF of client's DAYS_EMPLOYED")
```

```
np.quantile(-1*df_train[df_train['DAYS_EMPLOYED']<0]
['DAYS_EMPLOYED'].dropna(),0.99)/365
```

```
31.063890410958923
```

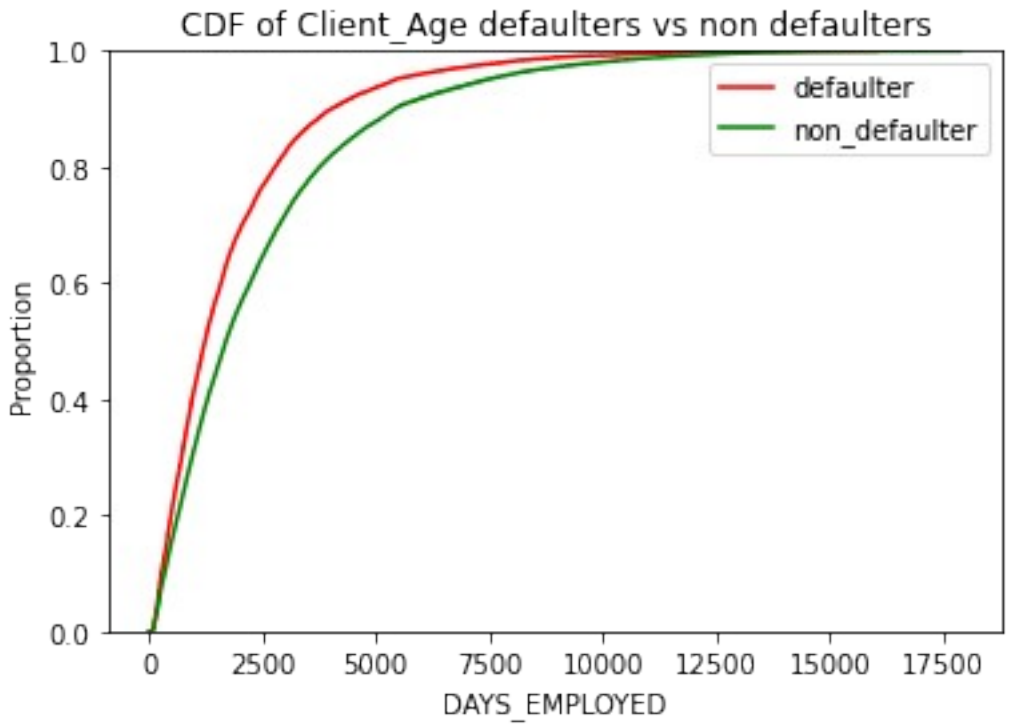
```
np.quantile(-1*df_train[df_train['DAYS_EMPLOYED']<0]
['DAYS_EMPLOYED'].dropna(),0.80)/365
```

```
10.07123287671233
```

- 99% client seems to have Days of employment more than 31 years
- 80% client seems to have have Days of employment more than 10 years

```
sns.ecdfplot(-1 *df_defaulter[df_defaulter['DAYS_EMPLOYED']<0]
['DAYS_EMPLOYED'],label='defaulter',color='red')
sns.ecdfplot(-1* df_non_defaulter[df_non_defaulter['DAYS_EMPLOYED']<0]
['DAYS_EMPLOYED'],label='non_defaulter',color='green')
#plt.xlim([40000,1000000])
plt.title("CDF of Client_Age defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

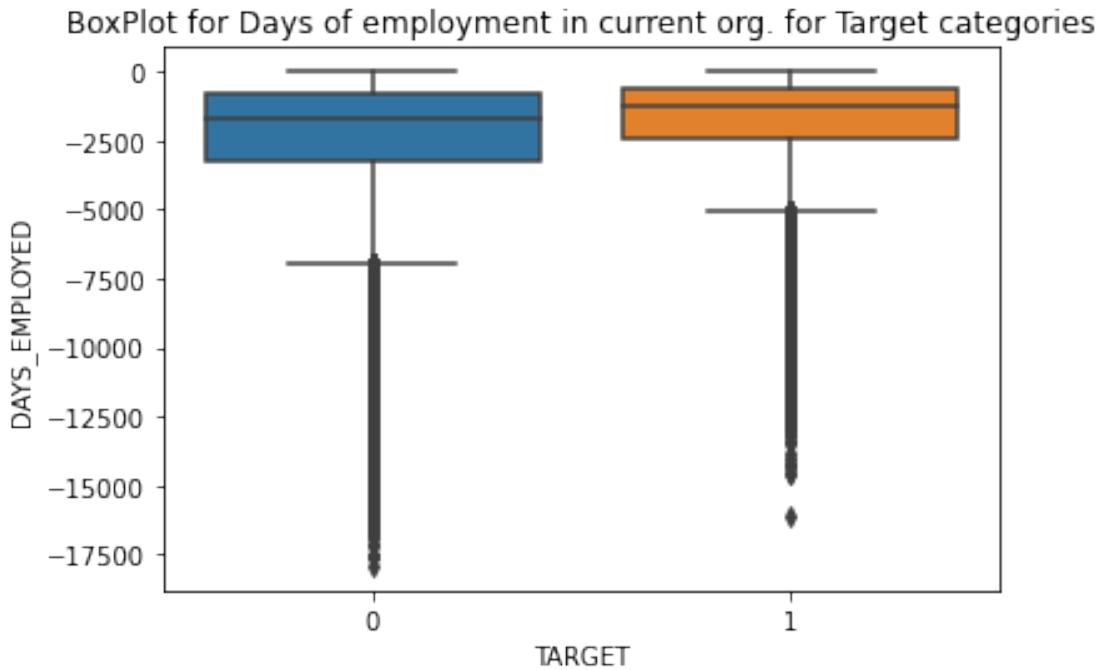
```
<matplotlib.legend.Legend at 0x1e231704e20>
```



- Avergae days of employment in current org. of defaulters are lesser in compare to non defaulters

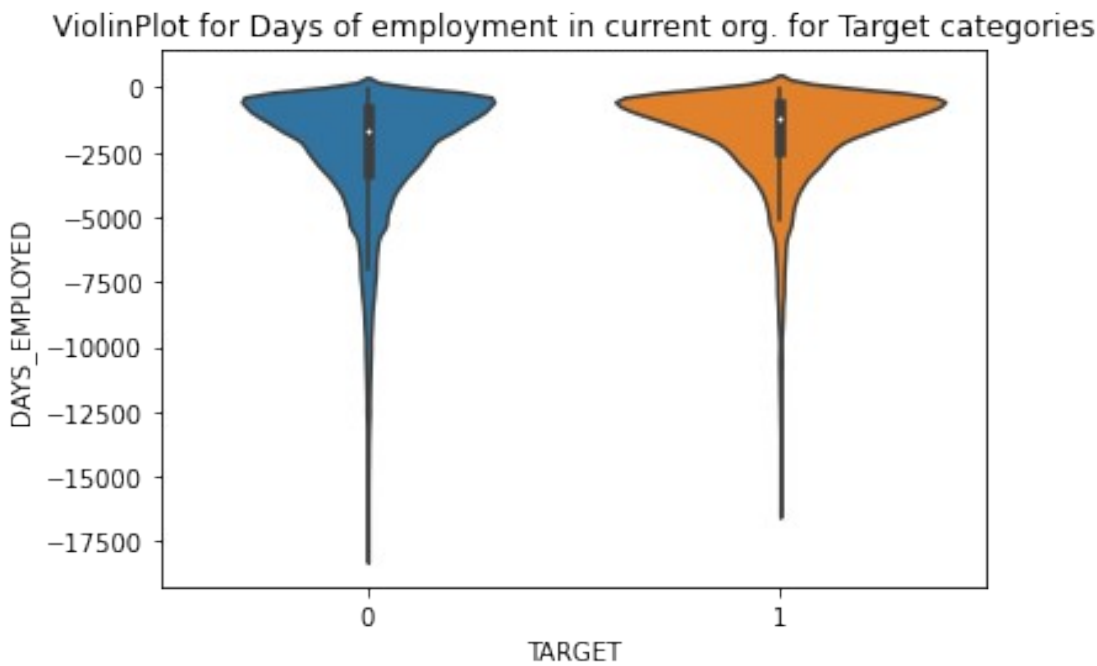
```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="DAYS_EMPLOYED", data=
df_train[df_train['DAYS_EMPLOYED']<0])
#plt.ylim([10,5000000])
plt.title("BoxPlot for Days of employment in current org. for Target
categories")
```

```
Text(0.5, 1.0, 'BoxPlot for Days of employment in current org. for
Target categories')
```




```
#  deafaulters are supposed to be averagely lesser day working in same
org. than Non defaulters

#Violin Plot
ax = sns.violinplot(x="TARGET", y="DAYS_EMPLOYED",
data=df_train[df_train['DAYS_EMPLOYED']<0])
#.ylim([40000,5000000])
plt.title("ViolinPlot for Days of employment in current org. for
Target categories")
```

```
Text(0.5, 1.0, 'ViolinPlot for Days of employment in current org. for
Target categories')
```




```
#top 10 correlated column
df_train[df_train['DAYS_EMPLOYED']<0].corr()
[['DAYS_EMPLOYED']].sort_values('DAYS_EMPLOYED',ascending=False)[:10]
```

```
                              DAYS_EMPLOYED
DAYS_EMPLOYED                      1.000000
DAYS_BIRTH                         0.352321
DAYS_REGISTRATION                  0.171751
DAYS_LAST_PHONE_CHANGE             0.141682
REG_CITY_NOT_WORK_CITY             0.128803
REG_CITY_NOT_LIVE_CITY             0.114092
DAYS_ID_PUBLISH                    0.089878
REG_REGION_NOT_WORK_REGION         0.081691
LIVE_CITY_NOT_WORK_CITY            0.075780
TARGET                             0.074948
```

- Days of employment in current org. has very good correlation with DAYS_BIRTH because it defintely increase with Age of clients

DAYS_REGISTRATION

How many days before the application did client change his registration

```
df_train['DAYS_REGISTRATION'].describe()
```

```
count   307511.000000
mean     -4986.120328
std       3522.886321
min     -24672.000000
```

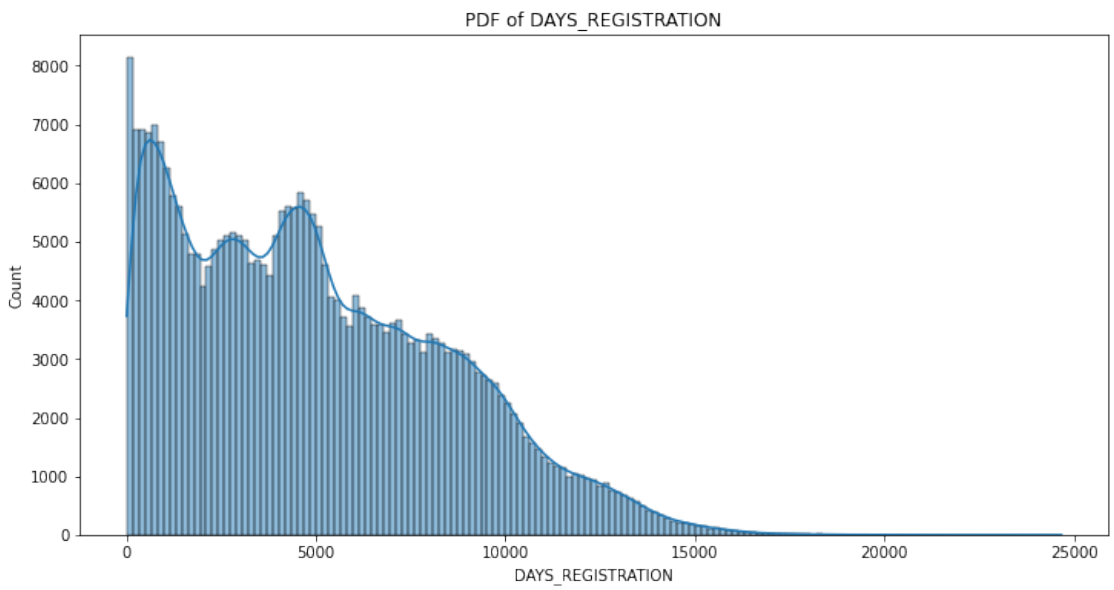
```
25%      -7479.500000
50%      -4504.000000
75%      -2010.000000
max          0.000000
Name: DAYS_REGISTRATION, dtype: float64
```

```
df_train['DAYS_REGISTRATION'].isna().sum()
# dont have NA values
```

```
0
```

```
#lets see its PDF and CDF
plt.figure(figsize=(12,6))
sns.histplot(-1*df_train['DAYS_REGISTRATION'],kde=True)
#plt.xlim([40000,5000000])
plt.title("PDF of DAYS_REGISTRATION")
```

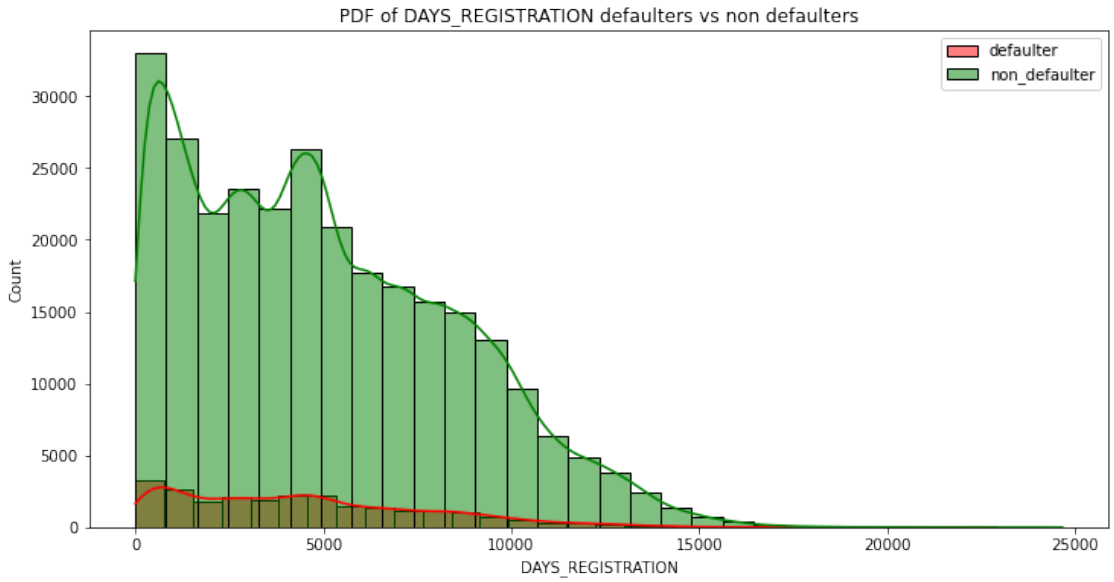
```
Text(0.5, 1.0, 'PDF of DAYS_REGISTRATION')
```




```
# It looks like  with values frequnecy is decreasing (because we have
just taken -1*)
```

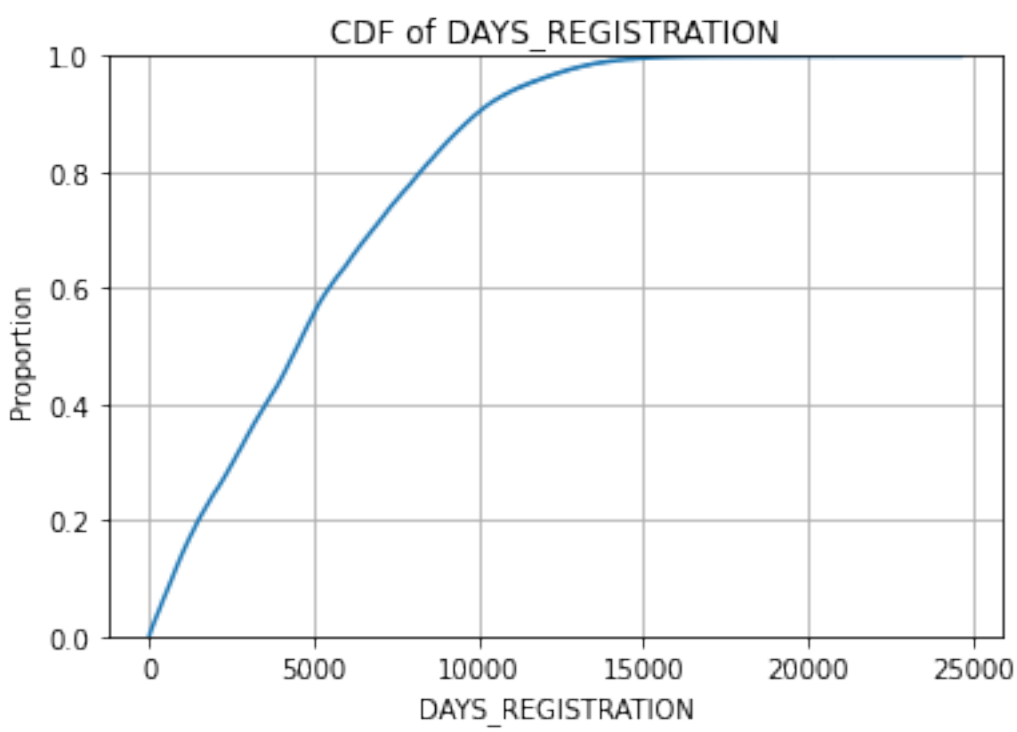
```
plt.figure(figsize=(12,6))
sns.histplot(-
1*df_defaulter['DAYS_REGISTRATION'],bins=30,kde=True,label='defaulter'
,color='red')
sns.histplot(-
1*df_non_defaulter['DAYS_REGISTRATION'],bins=30,kde=True,label='non_de
faulter',color="green")
#plt.xlim([40000,5000000])
plt.title("PDF of DAYS_REGISTRATION defaulters vs non defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e2124e1940>
```




```
sns.ecdfplot(-1*df_train['DAYS_REGISTRATION'])
#plt.xlim([40000,5000000])
plt.grid()
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of DAYS_REGISTRATION")
```

```
Text(0.5, 1.0, 'CDF of DAYS_REGISTRATION')
```




```
np.quantile(-1*df_train['DAYS_REGISTRATION'].dropna(),0.99)/365
```

```
38.02465753424657
```

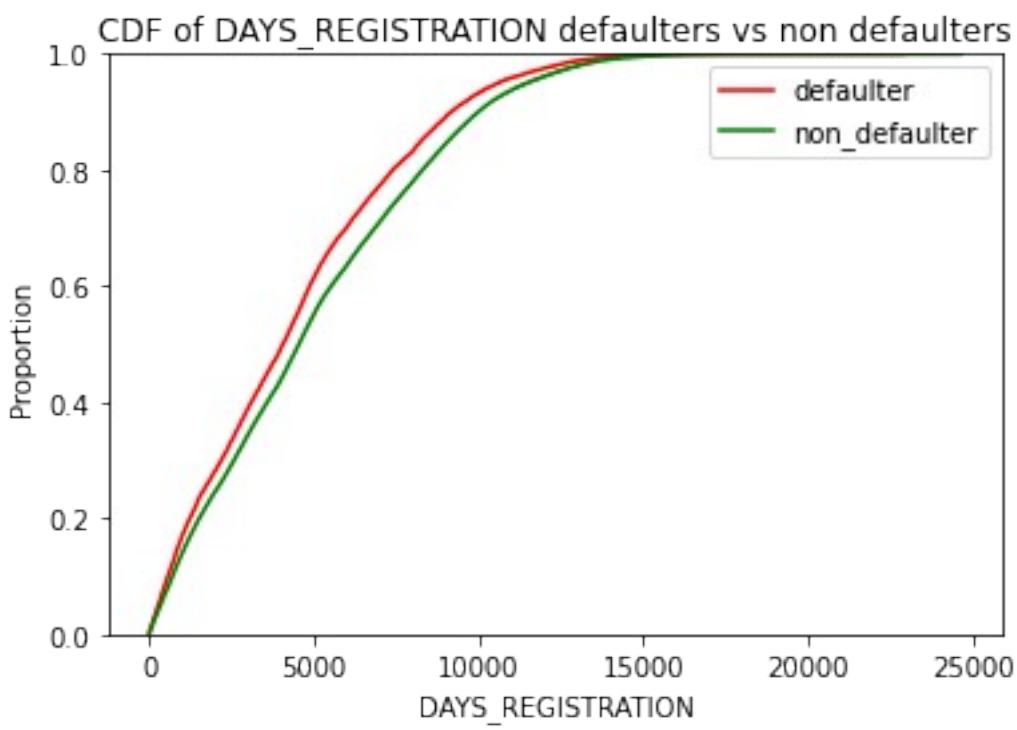
```
np.quantile(-1*df_train['DAYS_REGISTRATION'].dropna(),0.80)/365
```

```
22.54794520547945
```

- 99% client seems to have DAYS_REGISTRATION lesser than 38 years
- 80% client seems to have DAYS_REGISTRATION lesser than 22 years

```
sns.ecdfplot(-
1*df_defaulter['DAYS_REGISTRATION'],label='defaulter',color='red')
sns.ecdfplot(-
1*df_non_defaulter['DAYS_REGISTRATION'],label='non_defaulter',color='g
reen')
#plt.xlim([40000,1000000])
plt.title("CDF of DAYS_REGISTRATION defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

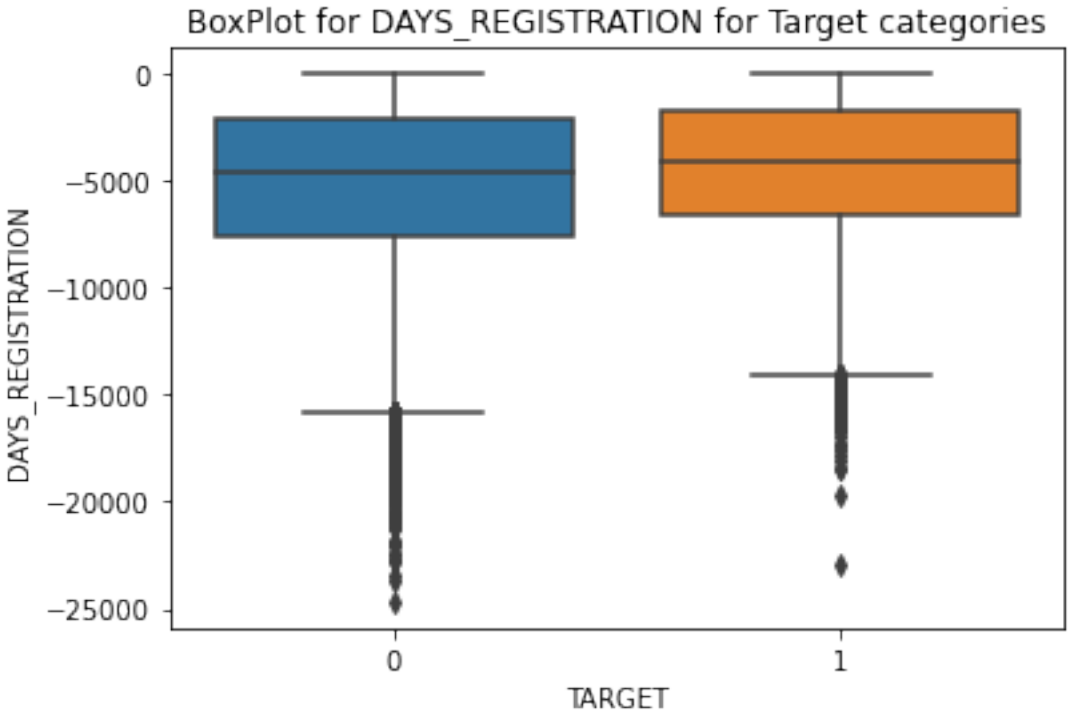
```
<matplotlib.legend.Legend at 0x1e2121eae80>
```



- Avergae DAYS_REGISTRATION of defaulters are lesser in compare to non defaulters

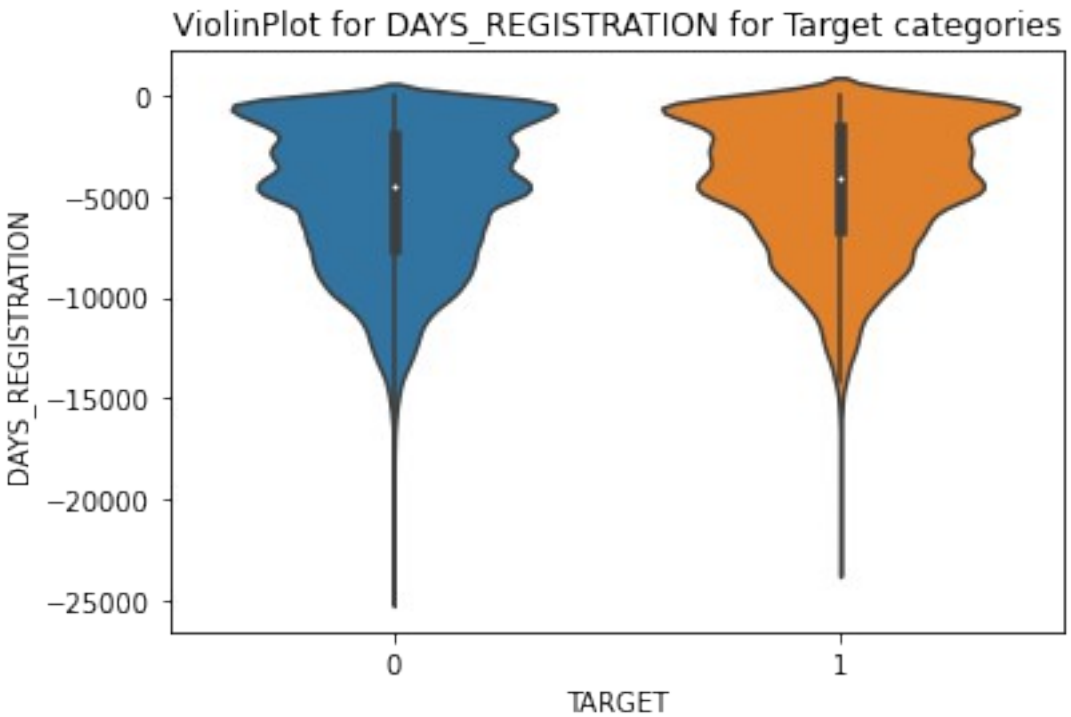
```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="DAYS_REGISTRATION", data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for DAYS_REGISTRATION for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for DAYS_REGISTRATION for Target categories')
```




```
#  deafaulters are supposed to be averagely lesser DAYS_REGISTRATION
than Non defaulters

#Violin Plot
ax = sns.violinplot(x="TARGET", y="DAYS_REGISTRATION", data=df_train)
#.ylim([40000,5000000])
plt.title("ViolinPlot for DAYS_REGISTRATION for Target categories")
```

```
Text(0.5, 1.0, 'ViolinPlot for DAYS_REGISTRATION for Target
categories')
```




```
#top 10 correlated column with DAYS_REGISTRATION
df_train.corr()
[['DAYS_REGISTRATION']].sort_values('DAYS_REGISTRATION',ascending=Fals
e)[:10]
```

```
                        DAYS_REGISTRATION
DAYS_REGISTRATION                1.000000
DAYS_BIRTH                       0.331912
FLAG_EMP_PHONE                   0.212361
CNT_CHILDREN                     0.183395
CNT_FAM_MEMBERS                  0.173416
YEARS_BUILD_AVG                  0.164184
YEARS_BUILD_MEDI                 0.164034
YEARS_BUILD_MODE                 0.162744
DAYS_ID_PUBLISH                  0.101896
REG_CITY_NOT_WORK_CITY           0.099874
```

- DAYS_REGISTRATION has good correlation with client Age because it defintely increase with Age of clients
- FLAG_EMP_PHONE has also good correlation with DAYS_REGISTRATION

DAYS_ID_PUBLISH

How many days before the application did client change the identity document with which he applied for the loan

```
df_train['DAYS_ID_PUBLISH'].describe()
```

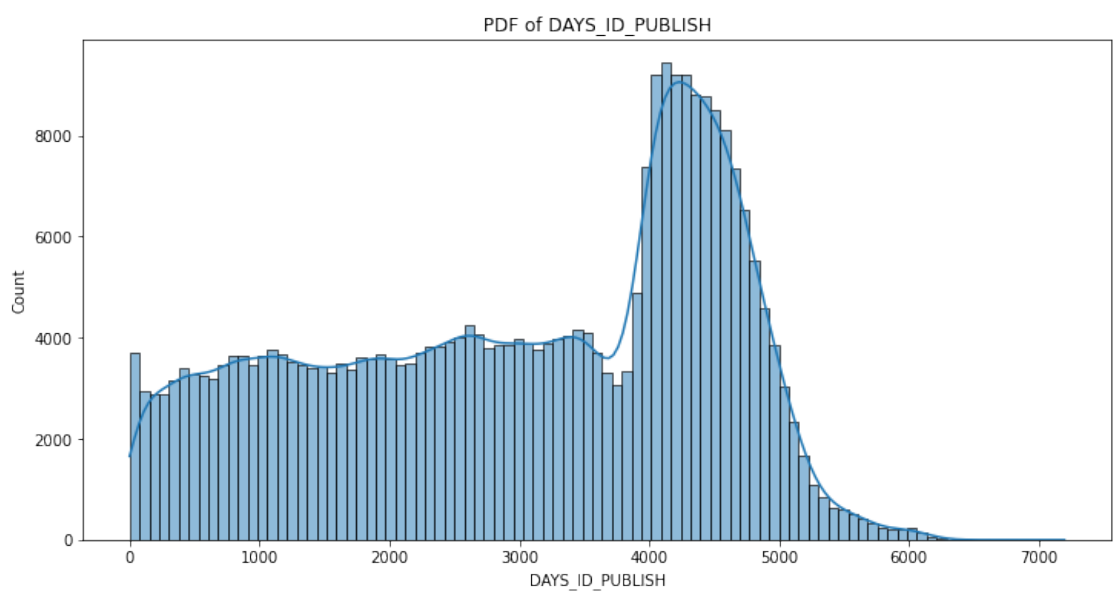
```
count    307511.000000
mean      -2994.202373
std        1509.450419
min       -7197.000000
25%       -4299.000000
50%       -3254.000000
75%       -1720.000000
max           0.000000
Name: DAYS_ID_PUBLISH, dtype: float64
```

```
df_train['DAYS_ID_PUBLISH'].isna().sum()
# dont have NA values
```

```
0
```

```
#lets see its PDF and CDF
plt.figure(figsize=(12,6))
sns.histplot(-1*df_train['DAYS_ID_PUBLISH'],kde=True)
#plt.xlim([40000,5000000])
plt.title("PDF of DAYS_ID_PUBLISH")
```

```
Text(0.5, 1.0, 'PDF of DAYS_ID_PUBLISH')
```

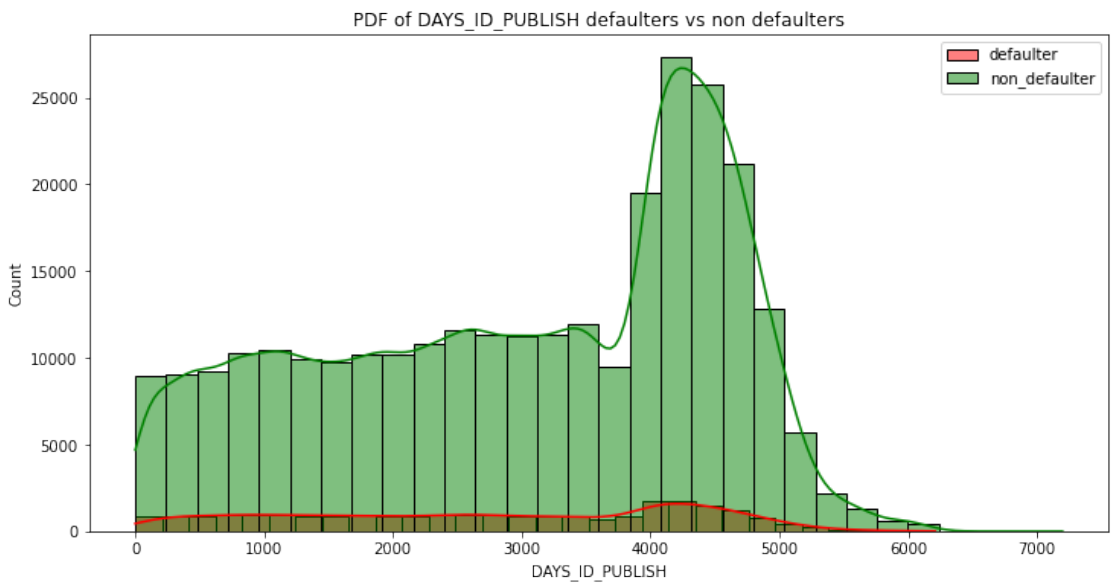



```
#So there are so many people who have older ids
```

```
plt.figure(figsize=(12,6))
sns.histplot(-
1*df_defaulter['DAYS_ID_PUBLISH'],bins=30,kde=True,label='defaulter',c
olor='red')
sns.histplot(-
1*df_non_defaulter['DAYS_ID_PUBLISH'],bins=30,kde=True,label='non_defa
ulter',color="green")
```

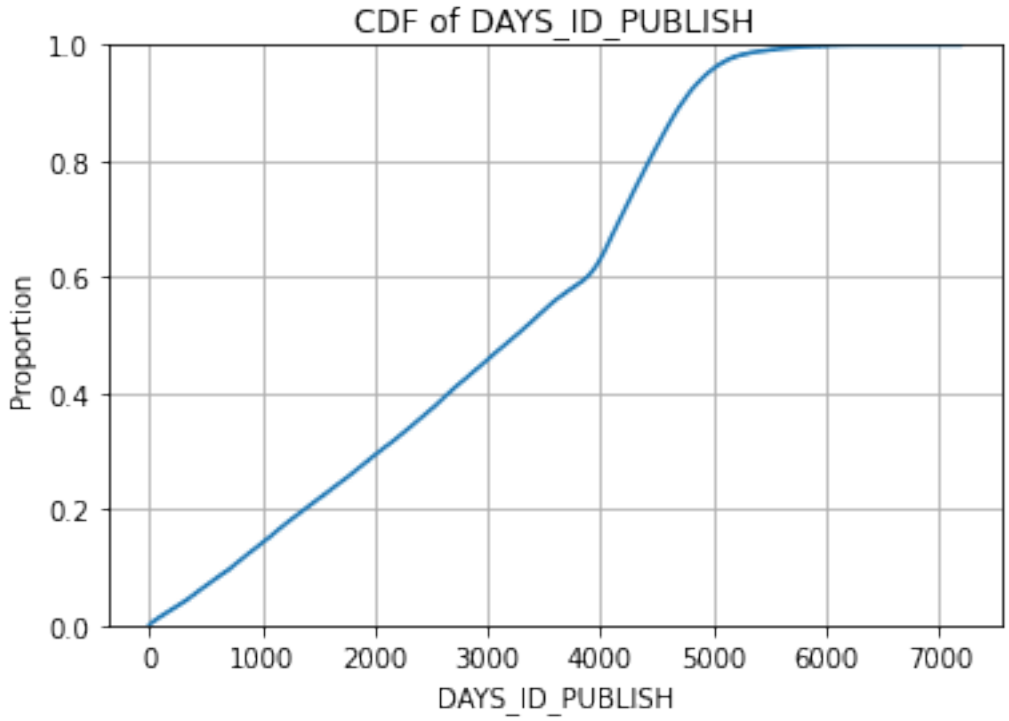
```
#plt.xlim([40000,5000000])
plt.title("PDF of DAYS_ID_PUBLISH defaulters vs non defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d111b95820>
```




```
sns.ecdfplot(-1*df_train['DAYS_ID_PUBLISH'])
#plt.xlim([40000,5000000])
plt.grid()
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of DAYS_ID_PUBLISH")
```

```
Text(0.5, 1.0, 'CDF of DAYS_ID_PUBLISH')
```




```
np.quantile(-1*df_train['DAYS_ID_PUBLISH'].dropna(),0.99)/365
```

```
14.923287671232877
```

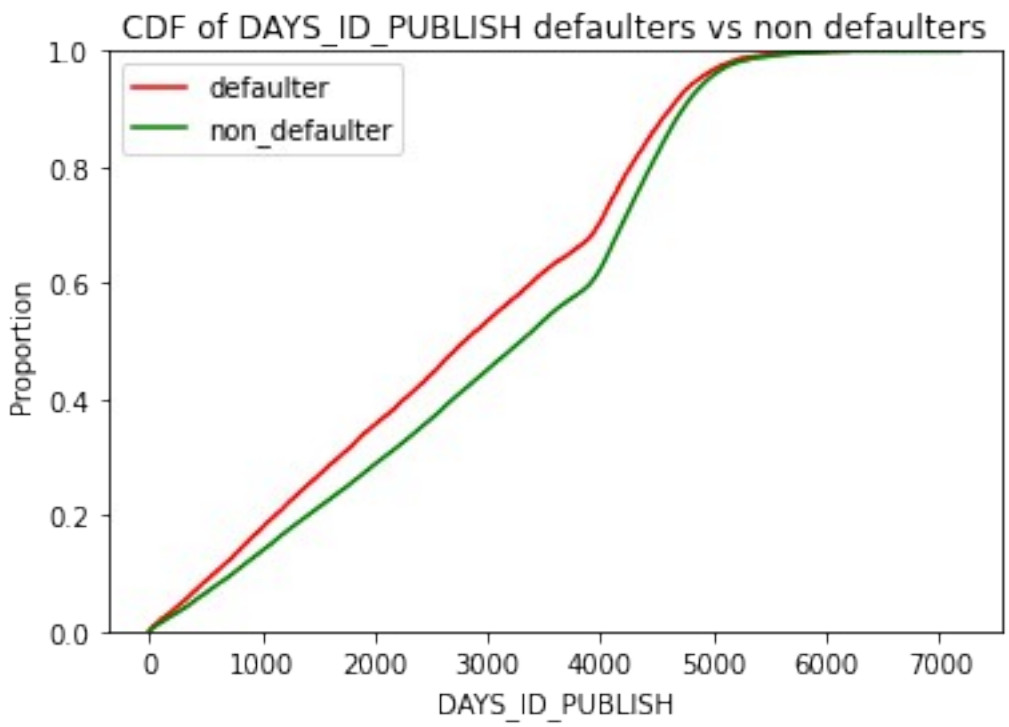
```
np.quantile(-1*df_train['DAYS_ID_PUBLISH'].dropna(),0.80)/365
```

```
12.136986301369863
```

- 99% client seems to have DAYS_ID_PUBLISH lesser than 14 years
- 80% client seems to have DAYS_ID_PUBLISH lesser than 12 years

```
sns.ecdfplot(-
1*df_defaulter['DAYS_ID_PUBLISH'],label='defaulter',color='red')
sns.ecdfplot(-
1*df_non_defaulter['DAYS_ID_PUBLISH'],label='non_defaulter',color='gre
en')
#plt.xlim([40000,1000000])
plt.title("CDF of DAYS_ID_PUBLISH defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

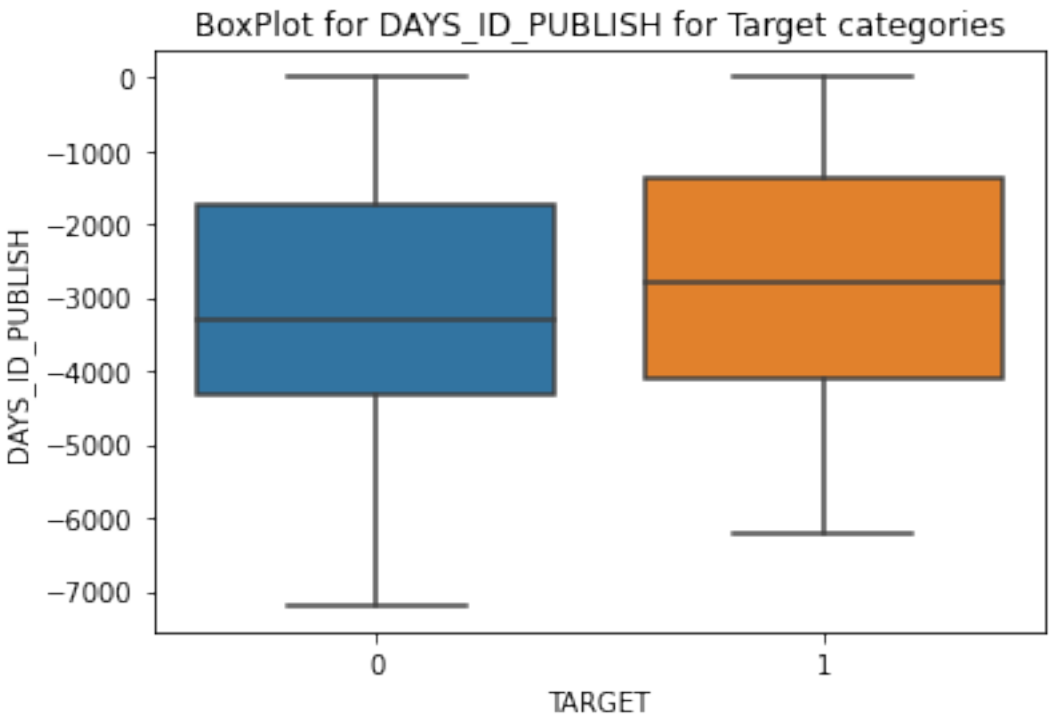
```
<matplotlib.legend.Legend at 0x1d12a76c310>
```



- Avergae DAYS_ID_PUBLISH of defaulters are lesser in compare to non defaulters

```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="DAYS_ID_PUBLISH", data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for DAYS_ID_PUBLISH for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for DAYS_ID_PUBLISH for Target categories')
```




```
#The DAYS_ID_PUBLISH seem to be lower for defaulters than non
defaulters
```

OWN_CAR_AGE

How old client's car is

```
df_train['OWN_CAR_AGE'].describe()
```

```
count    104582.000000
mean         12.061091
std          11.944812
min           0.000000
25%           5.000000
50%           9.000000
75%          15.000000
max          91.000000
Name: OWN_CAR_AGE, dtype: float64
```

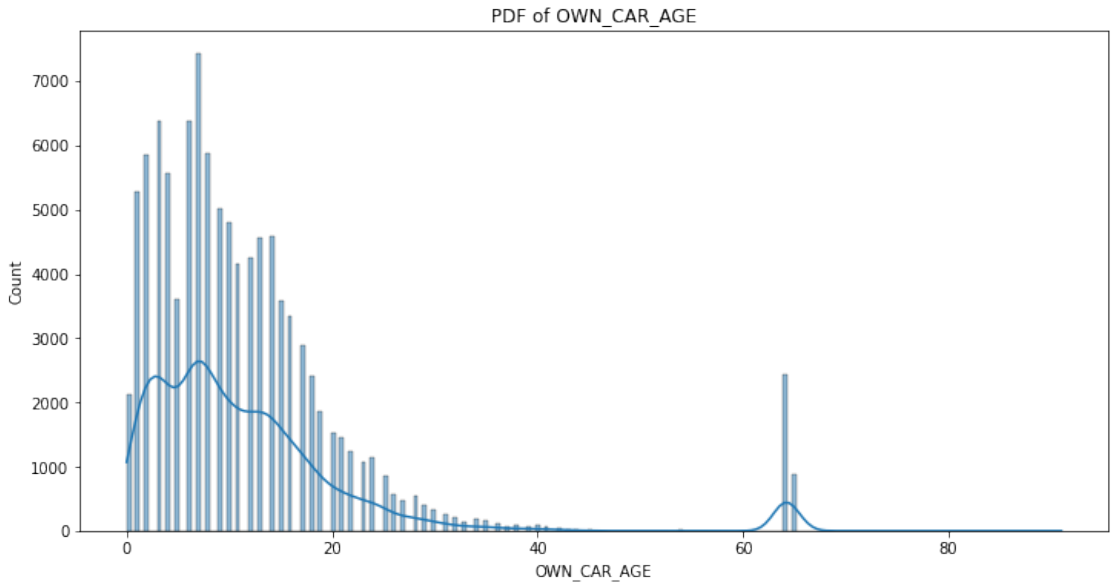
```
df_train['OWN_CAR_AGE'].isna().sum()
# have NA values,because client many not have a car
```

```
202929
```

```
#lets see its PDF and CDF
plt.figure(figsize=(12,6))
sns.histplot(df_train['OWN_CAR_AGE'],kde=True)
```

```
#plt.xlim([40000,5000000])
plt.title("PDF of OWN_CAR_AGE")
```

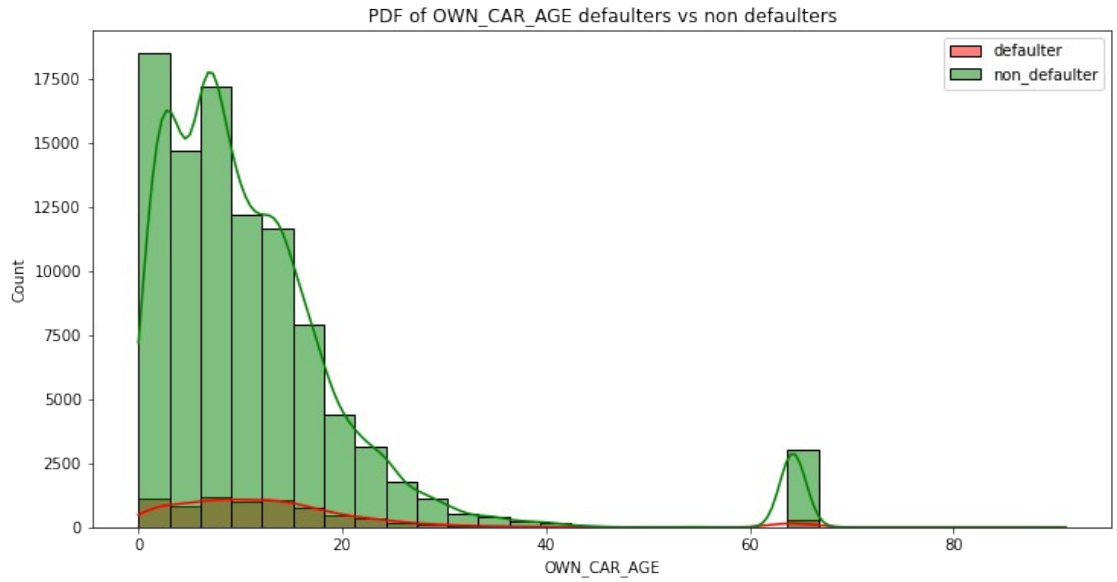
```
Text(0.5, 1.0, 'PDF of OWN_CAR_AGE')
```




```
#So there are so many people whose car age is 0,Mostly cars are new, I
think these age are in years
```

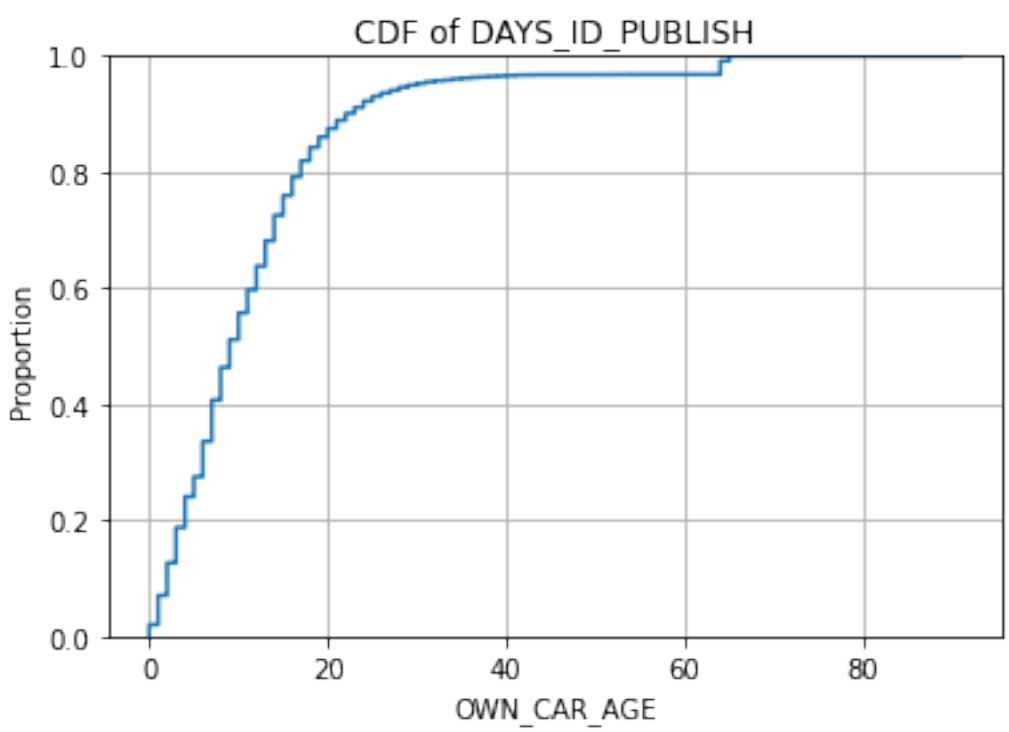
```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['OWN_CAR_AGE'],bins=30,kde=True,label='defau
lter',color='red')
sns.histplot(df_non_defaulter['OWN_CAR_AGE'],bins=30,kde=True,label='n
on_defaulter',color="green")
#plt.xlim([40000,5000000])
plt.title("PDF of OWN_CAR_AGE defaulters vs non defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d1293be040>
```




```
sns.ecdfplot(df_train['OWN_CAR_AGE'])
#plt.xlim([40000,5000000])
plt.grid()
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of DAYS_ID_PUBLISH")
```

```
Text(0.5, 1.0, 'CDF of DAYS_ID_PUBLISH')
```




```
np.quantile(df_train['OWN_CAR_AGE'].dropna(),0.90)
```

```
22.0
```

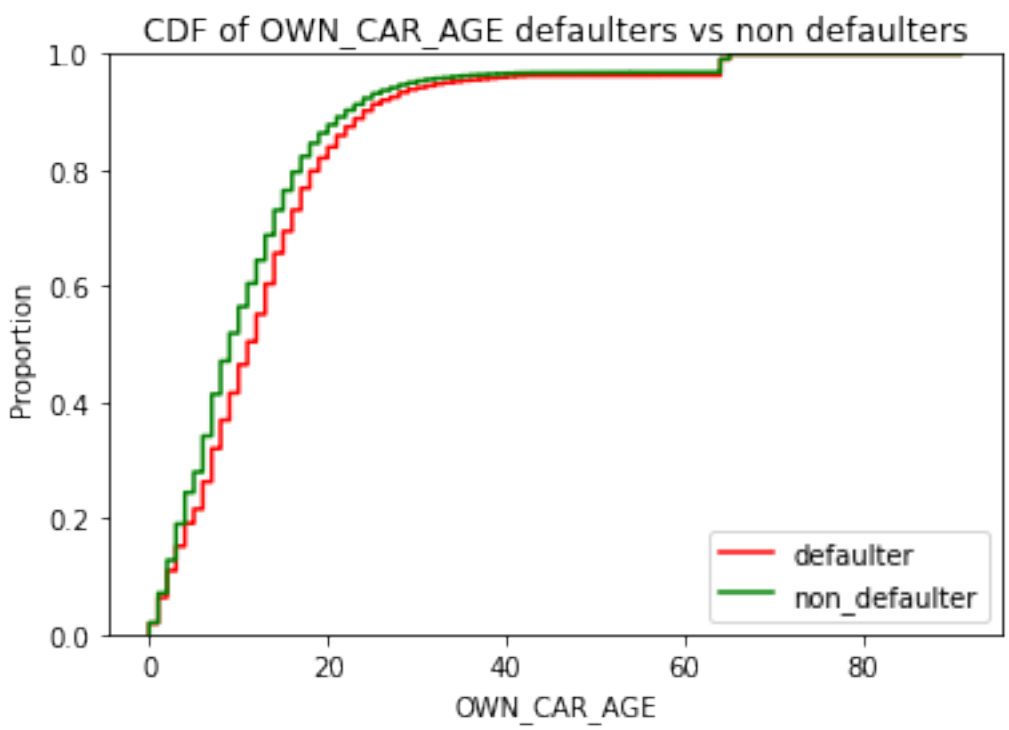
```
np.quantile(df_train['OWN_CAR_AGE'].dropna(),0.70)
```

```
14.0
```

- 90% client seems to have OWN_CAR_AGE lesser than 22 years
- 70% client seems to have DAYS_ID_PUBLISH lesser than 14 years

```
sns.ecdfplot(df_defaulter['OWN_CAR_AGE'],label='defaulter',color='red'
)
sns.ecdfplot(df_non_defaulter['OWN_CAR_AGE'],label='non_defaulter',col
or='green')
#plt.xlim([40000,1000000])
plt.title("CDF of OWN_CAR_AGE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

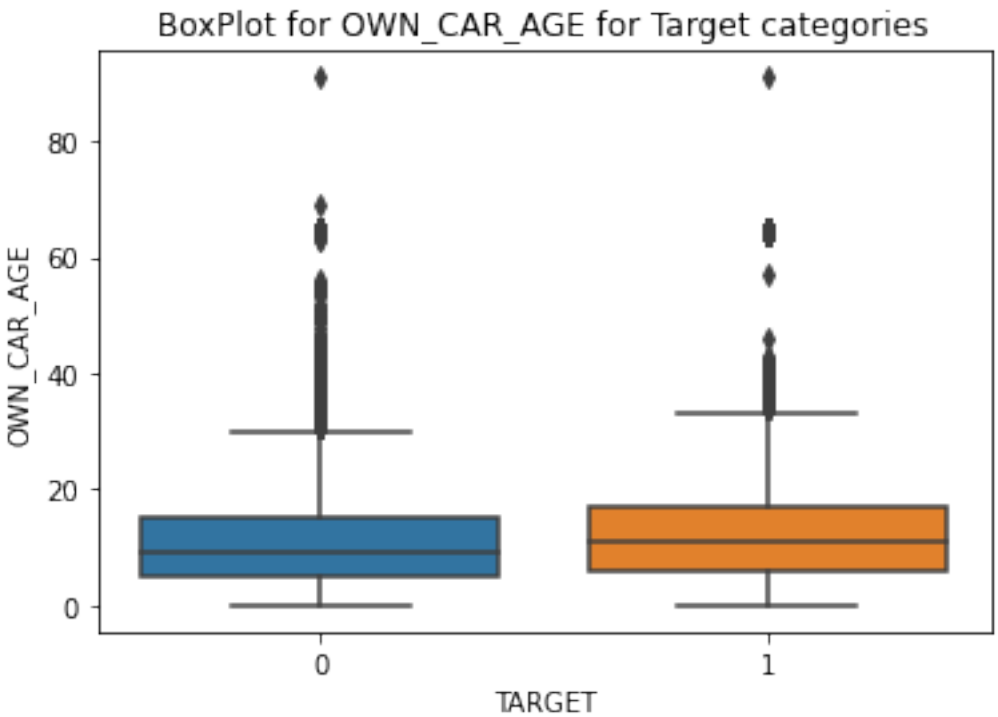
```
<matplotlib.legend.Legend at 0x1d12990deb0>
```



- Avergae OWN_CAR_AGE of defaulters are more in compare to non defaulters

```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="OWN_CAR_AGE", data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for OWN_CAR_AGE for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for OWN_CAR_AGE for Target categories')
```




```
#The OWN_CAR_AGE seem to be little higher for defaulters than non
defaulters
```

- 'FLAG_MOBIL',
- 'FLAG_EMP_PHONE',
- 'FLAG_WORK_PHONE',
- 'FLAG_CONT_MOBILE',
- 'FLAG_PHONE',
- 'FLAG_EMAIL',

Those seems to be just caregorical varibales as its a 1= YES and 0= NO

FLAG_MOBIL

Did client provide mobile phone

```
df_train['FLAG_MOBIL'].value_counts()
```

- So all peope provided mobile phone,we can drop this column as it has a very low variance

FLAG_EMP_PHONE

Did client provide work phone

```
df_train['FLAG_EMP_PHONE'].value_counts()
```

```
1    252125
0     55386
Name: FLAG_EMP_PHONE, dtype: int64
```

```
df_train['FLAG_EMP_PHONE'].isna().sum()
#have NA values
```

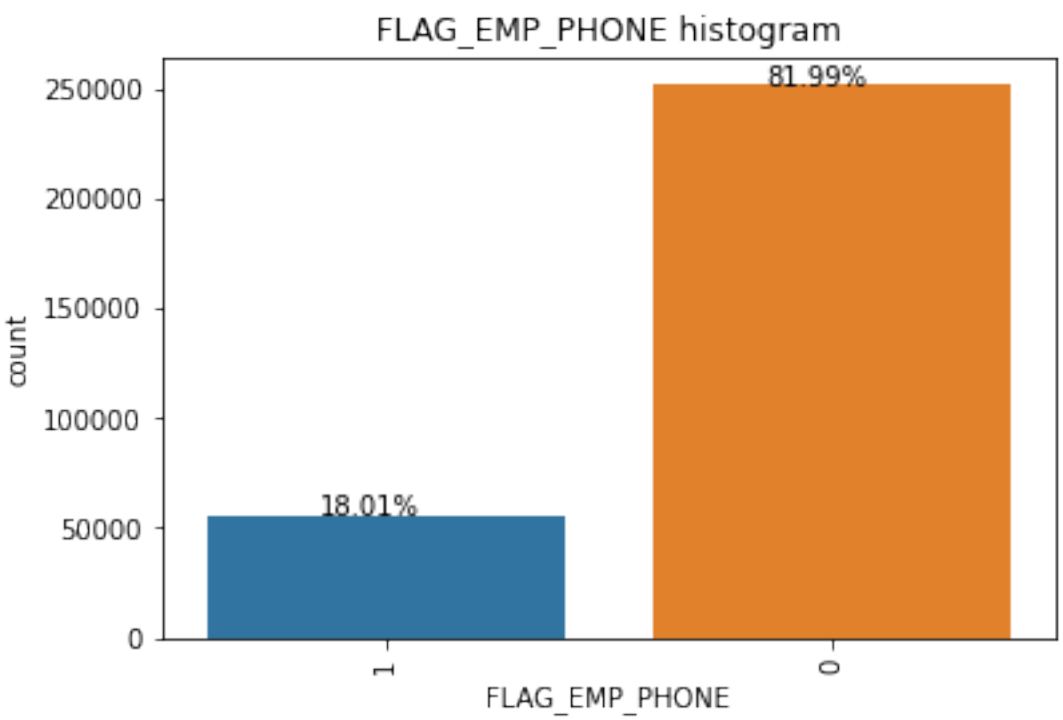
```
0
```

```
df_train['FLAG_EMP_PHONE'].isna().sum()*100/df_train.shape[0]
```

```
0.0
```

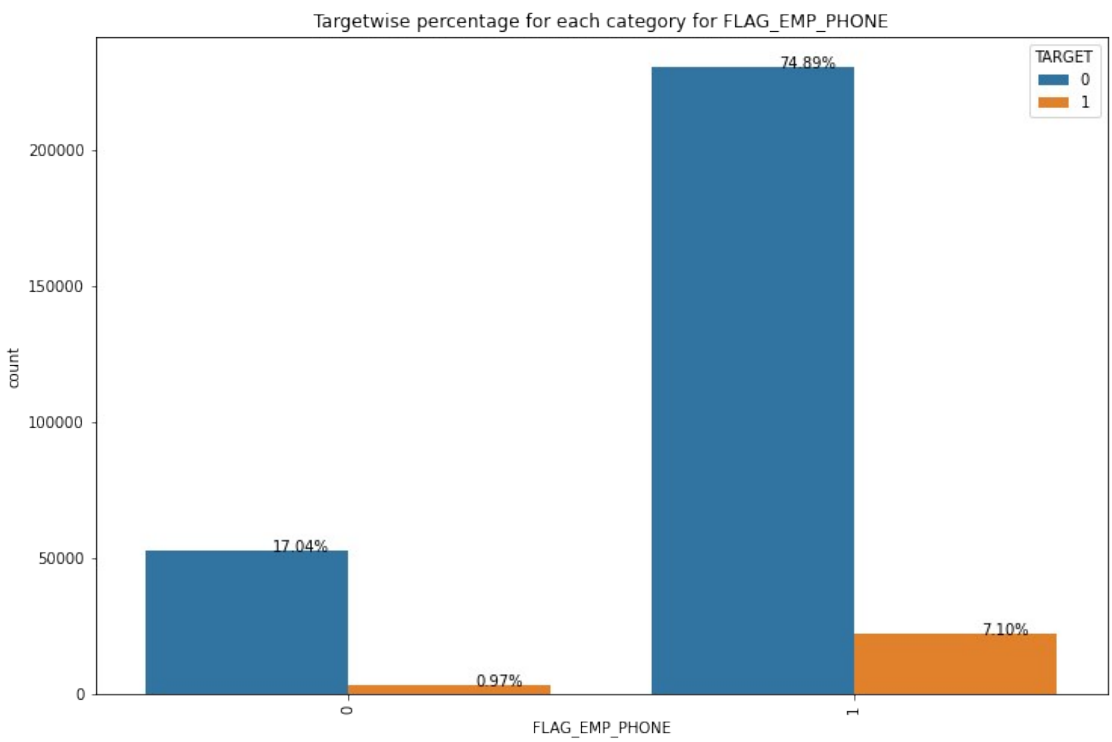
```
#lets see its distribution

ax=sns.countplot(x='FLAG_EMP_PHONE',data=df_train)
ax.set_xticklabels(df_train['FLAG_EMP_PHONE'].value_counts().index.to_
list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("FLAG_EMP_PHONE histogram")
```

```
Text(0.5, 1.0, 'FLAG_EMP_PHONE histogram')
```



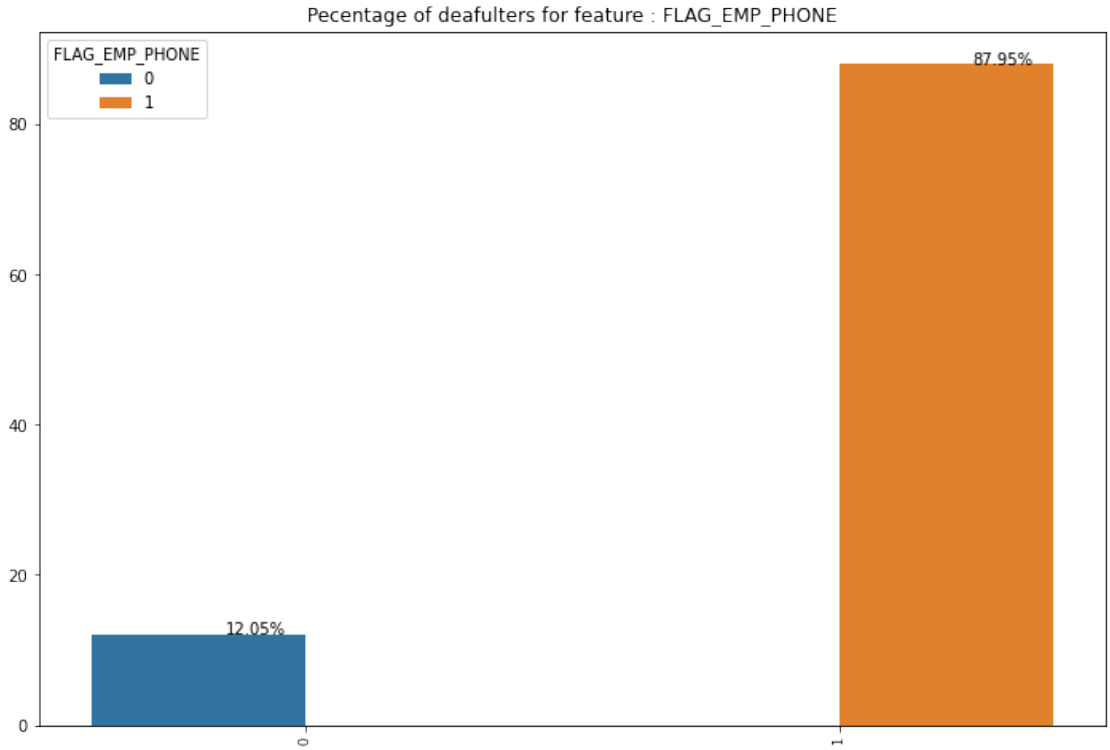
```
#very less people have EMP Phone

#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FLAG_EMP_PHONE',hue='TARGET',data=df_train)
plt.xticks(rotation=90)
#ax.set_xticklabels(df_train['FLAG_EMP_PHONE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Targetwise percentage for each category for
FLAG_EMP_PHONE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FLAG_EMP_PHONE')
```




```
plot_for_categorical_defaulters(df_train,'FLAG_EMP_PHONE')
```

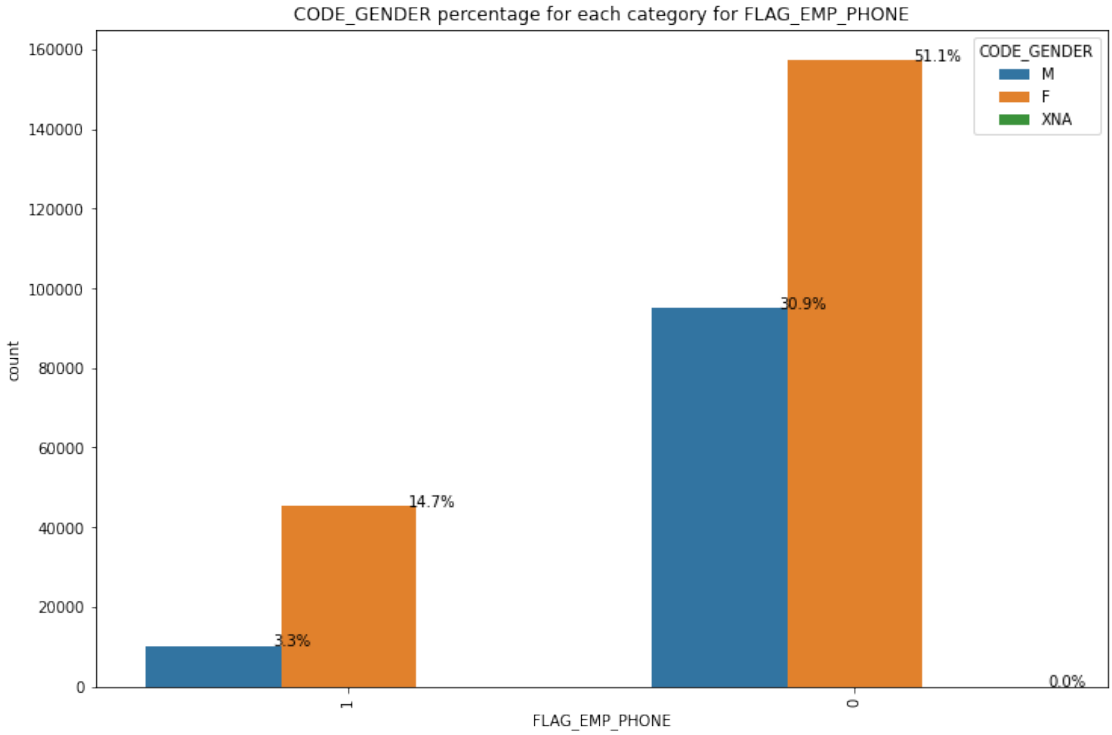
```
{1: 87.95166163141994, 0: 12.04833836858006}
```



it seems people provide Work phone number are more Deafulters

```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FLAG_EMP_PHONE',hue='CODE_GENDER',data=df_train)
ax.set_xticklabels(df_train['FLAG_EMP_PHONE'].value_counts().index.to_
list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("CODE_GENDER percentage for each category for
FLAG_EMP_PHONE")
```

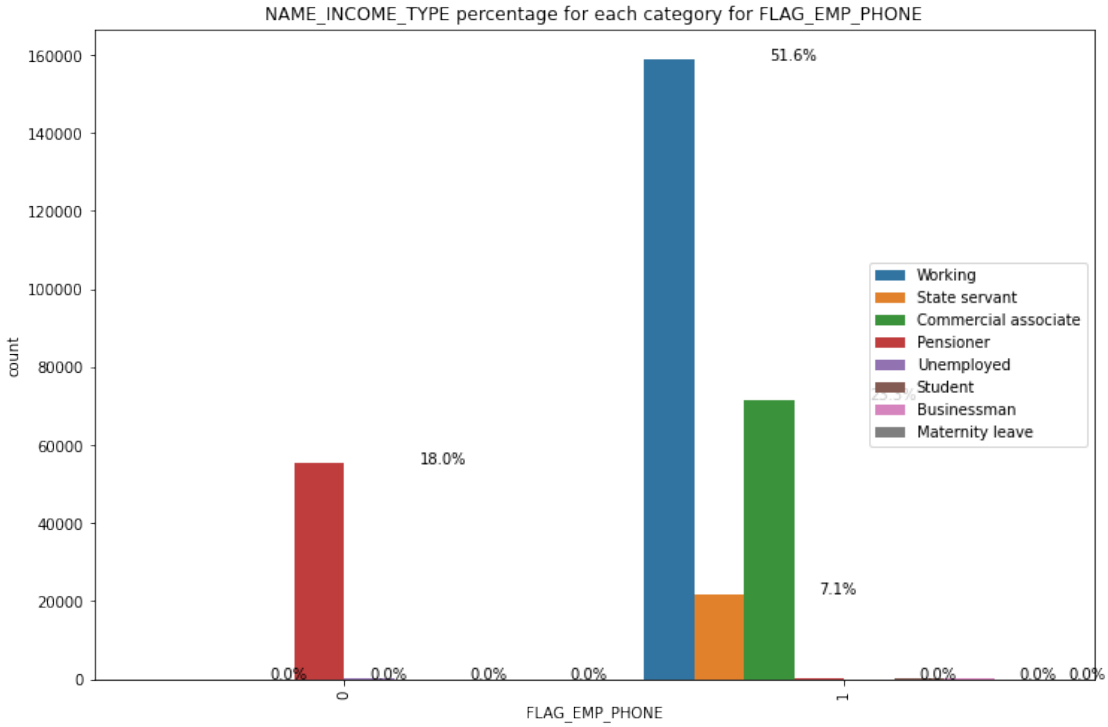
```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
FLAG_EMP_PHONE')
```



- Most non working and pensioner woman cant give work phone

```
#lets see its distribution with resepect to NAME_INCOME_TYPE variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FLAG_EMP_PHONE',hue='NAME_INCOME_TYPE',data=df_tra
in)
plt.xticks(rotation=90)
ax.legend(loc='right')
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("NAME_INCOME_TYPE percentage for each category for
FLAG_EMP_PHONE")
```

```
Text(0.5, 1.0, 'NAME_INCOME_TYPE percentage for each category for
FLAG_EMP_PHONE')
```



- it shows that most of client who has provided the Work Phone are Wokring or commercial associate and state govt employees
- Pensioner cant give work Phone obiviously,So we cant have providing phone will affect repayment performance

FLAG_WORK_PHONE

Did client Working phone

```
df_train['FLAG_WORK_PHONE'].value_counts()
```

```
0    246203
1     61308
Name: FLAG_WORK_PHONE, dtype: int64
```

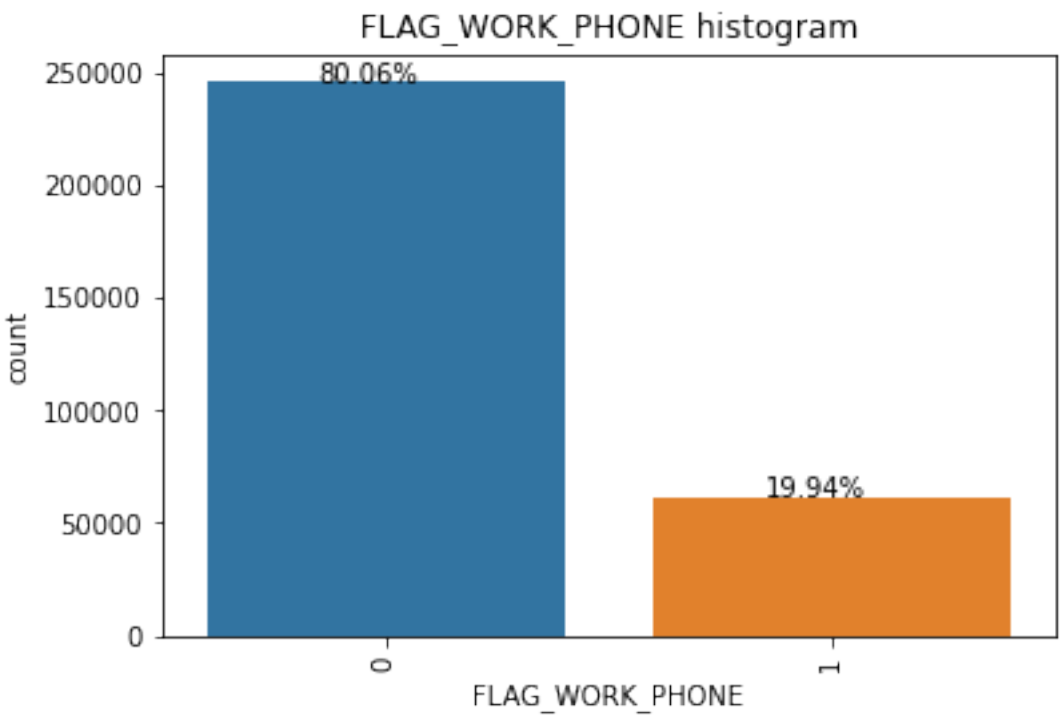
```
df_train['FLAG_WORK_PHONE'].isna().sum()
#have NA values
```

```
0
```

```
#lets see its distribution

ax=sns.countplot(x='FLAG_WORK_PHONE',data=df_train)
ax.set_xticklabels(df_train['FLAG_WORK_PHONE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
```

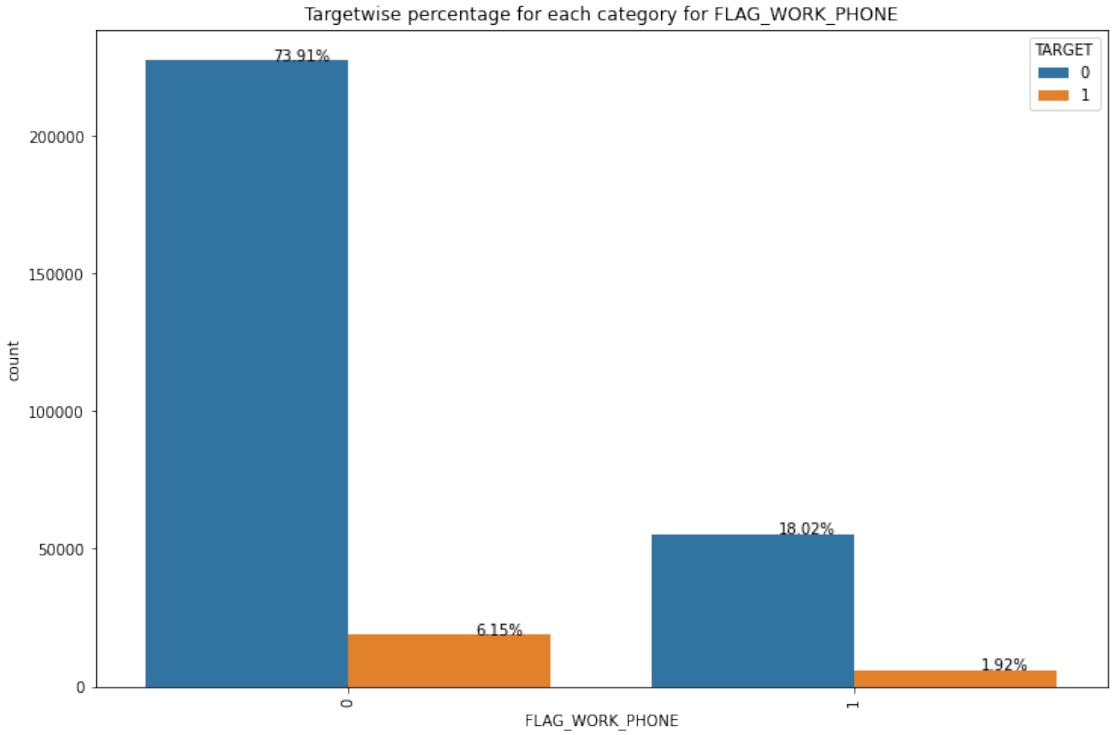
```
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("FLAG_WORK_PHONE histogram")
```

```
Text(0.5, 1.0, 'FLAG_WORK_PHONE histogram')
```

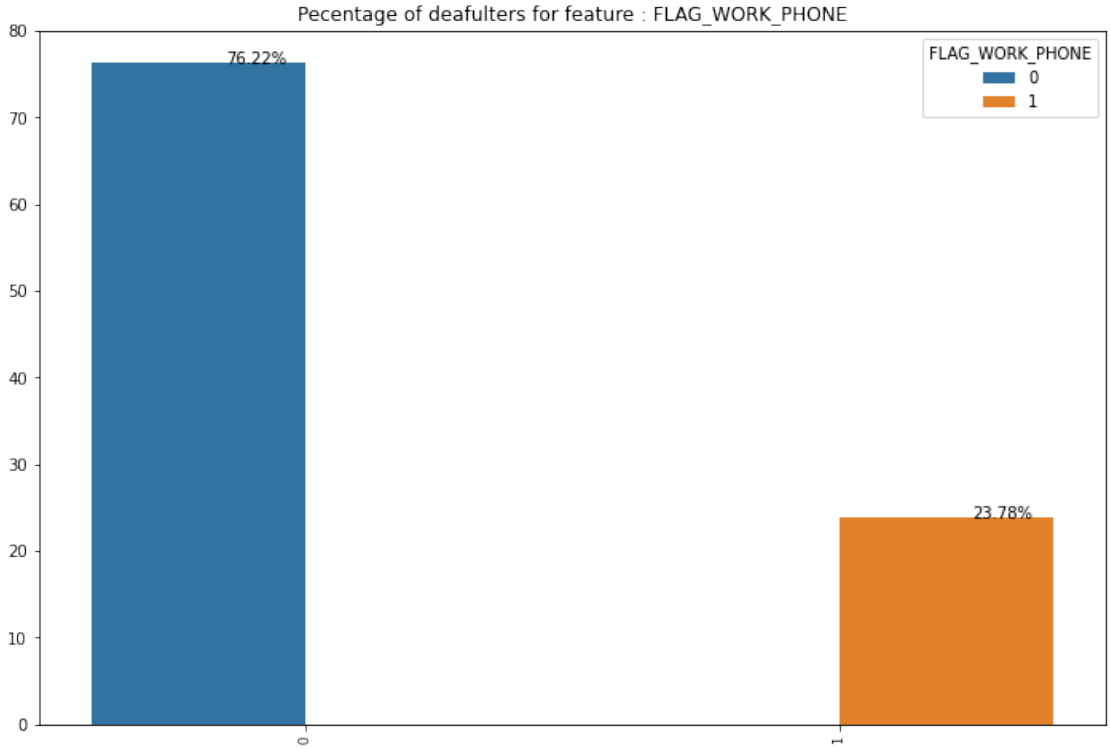



```
# less people have Working place phone

#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FLAG_WORK_PHONE',hue='TARGET',data=df_train)
plt.xticks(rotation=90)
#ax.set_xticklabels(df_train['FLAG_EMP_PHONE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
FLAG_WORK_PHONE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FLAG_WORK_PHONE')
```




```
plot_for_categorical_defaulters(df_train,'FLAG_WORK_PHONE')
```



- More defaulters dont have Working place Phone

```
#top 10 correlated column with DAYS_REGISTRATION
df_train.corr()
[['FLAG_WORK_PHONE']].sort_values('FLAG_WORK_PHONE',ascending=False)
[:10]
```

```
                            FLAG_WORK_PHONE
FLAG_WORK_PHONE                    1.000000
FLAG_PHONE                         0.293105
FLAG_EMP_PHONE                     0.233801
DAYS_BIRTH                         0.172457
FLAG_DOCUMENT_11                   0.123500
REG_CITY_NOT_WORK_CITY             0.121000
LIVE_CITY_NOT_WORK_CITY            0.109890
REG_REGION_NOT_WORK_REGION         0.068202
CNT_FAM_MEMBERS                    0.067191
REG_REGION_NOT_LIVE_REGION         0.064484
```

FLAG_PHONE

Did client provide home phone (1=YES, 0=NO)

```
df_train['FLAG_PHONE'].value_counts()
```

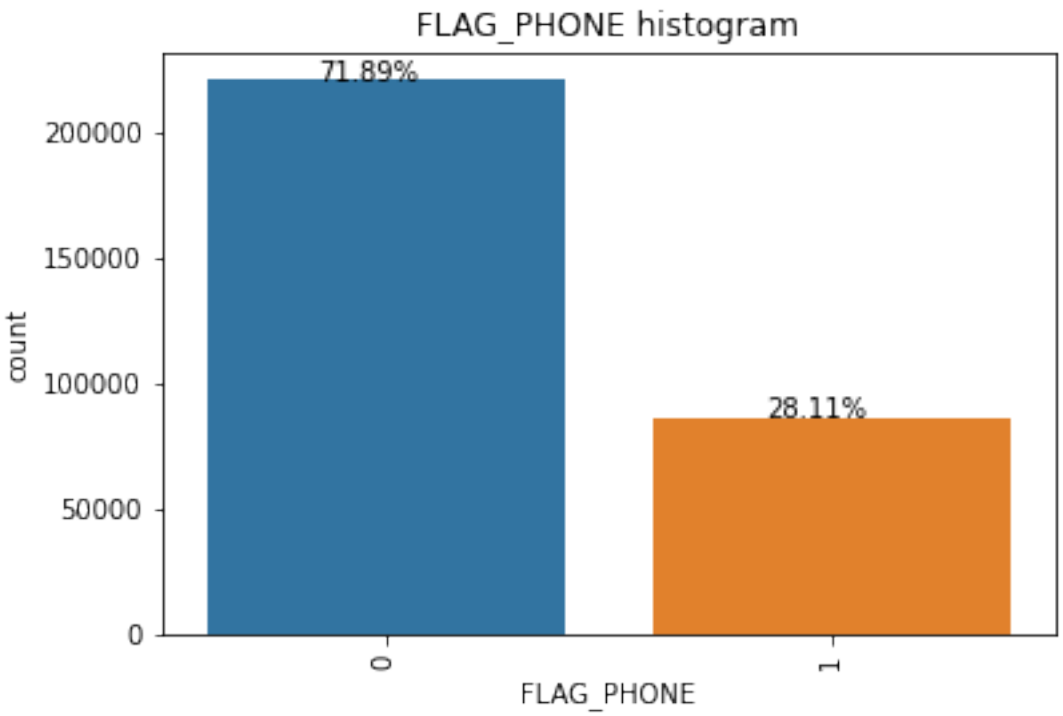
```
0    221080
1     86431
Name: FLAG_PHONE, dtype: int64
```

```
df_train['FLAG_PHONE'].isna().sum()
#have NA values
```

```
0
```

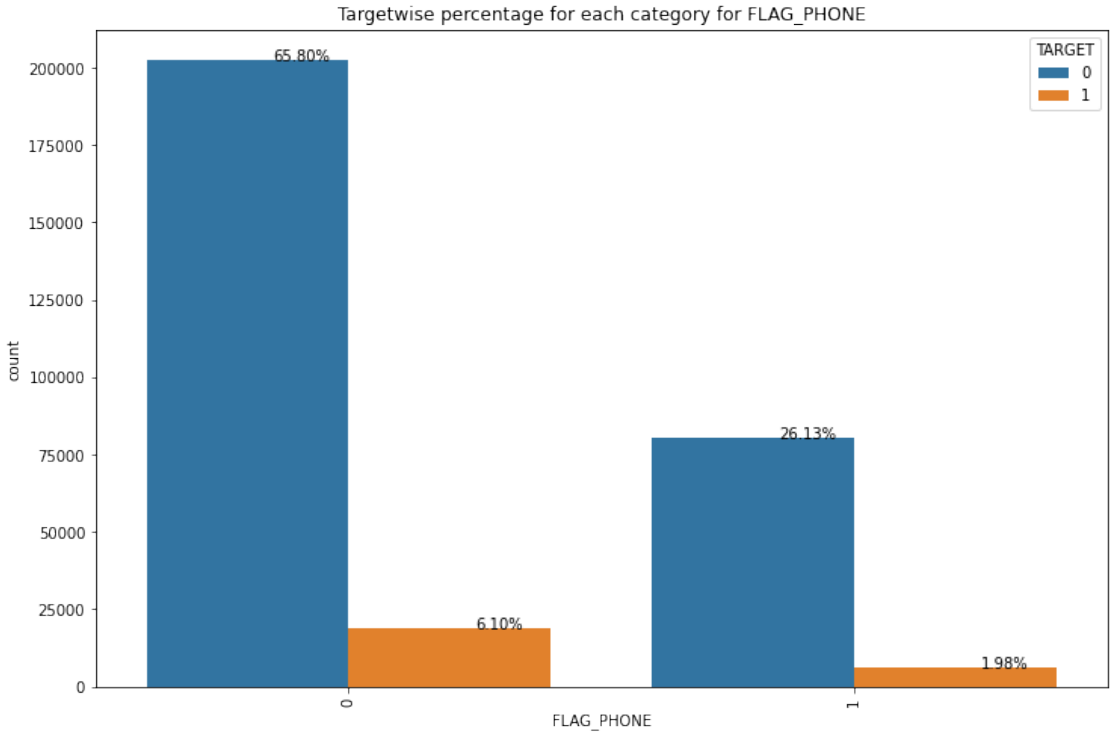
```
#lets see its distribution

ax=sns.countplot(x='FLAG_PHONE',data=df_train)
ax.set_xticklabels(df_train['FLAG_PHONE'].value_counts().index.to_list
(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("FLAG_PHONE histogram")
```

```
Text(0.5, 1.0, 'FLAG_PHONE histogram')
```

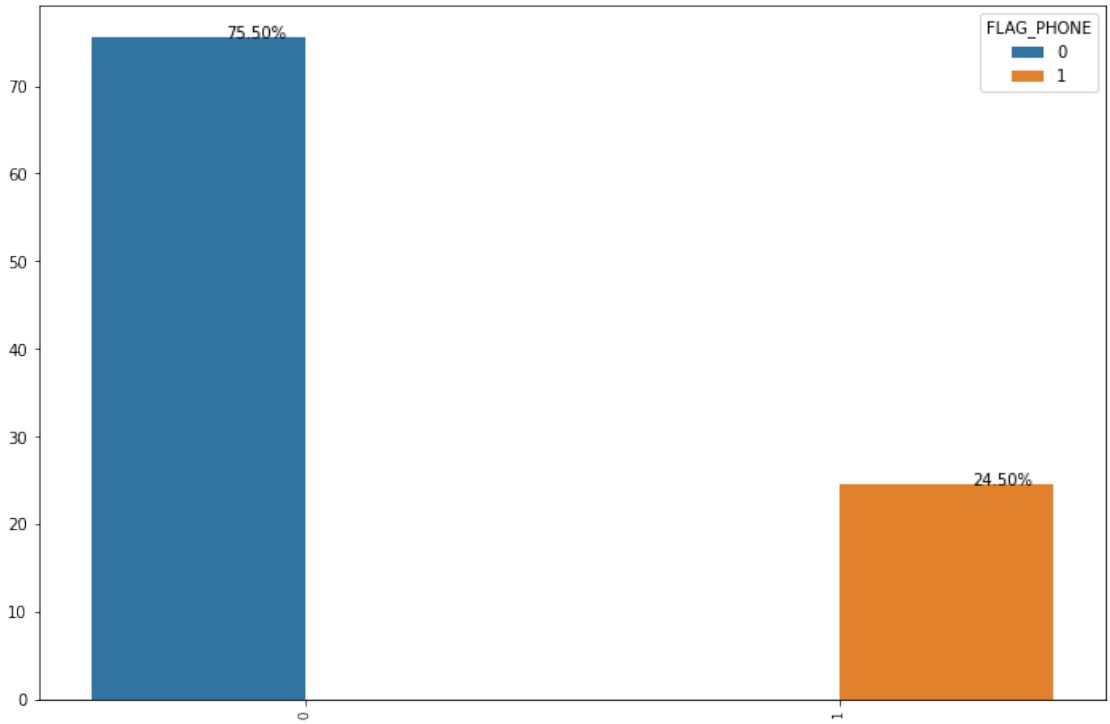



```
# less people have Home phone

#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FLAG_PHONE',hue='TARGET',data=df_train)
plt.xticks(rotation=90)
#ax.set_xticklabels(df_train['FLAG_EMP_PHONE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for FLAG_PHONE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FLAG_PHONE')
```




```
plot_for_categorical_defaulters(df_train,'FLAG_PHONE')
```



- More defaulters dont have Home Phone

```
#top 10 correlated column with FLAG_PHONE
df_train.corr()
[['FLAG_PHONE']].sort_values('FLAG_PHONE',ascending=False)[:10]
```

```
                            FLAG_PHONE
FLAG_PHONE                    1.000000
FLAG_WORK_PHONE               0.293105
REGION_POPULATION_RELATIVE    0.090939
EXT_SOURCE_1                  0.081241
ELEVATORS_AVG                 0.077343
ELEVATORS_MEDI                0.076233
FLAG_DOCUMENT_5               0.075455
FLOORSMAX_AVG                 0.071840
ELEVATORS_MODE                0.071398
FLOORSMAX_MEDI                0.070842
```

FLAG_CONT_MOBILE

Was Client's mobile reachable (1=YES, 0=NO)

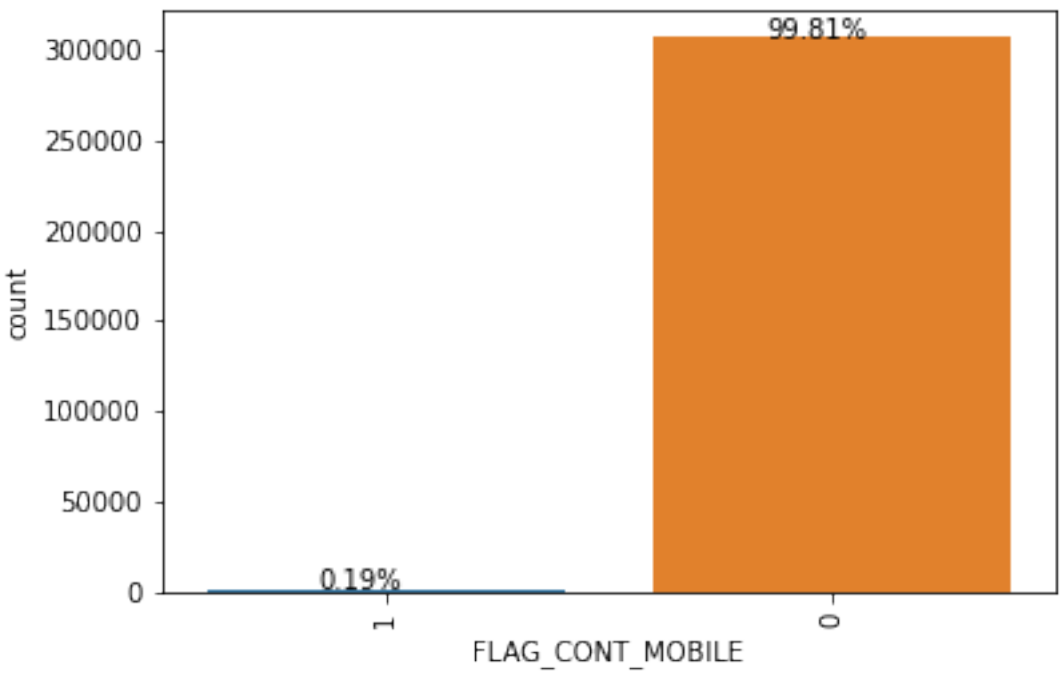
```
df_train['FLAG_CONT_MOBILE'].value_counts()
```

```
1    306937
0       574
Name: FLAG_CONT_MOBILE, dtype: int64
```

```
df_train['FLAG_CONT_MOBILE'].isna().sum()
#have NA values
```

```
0
```

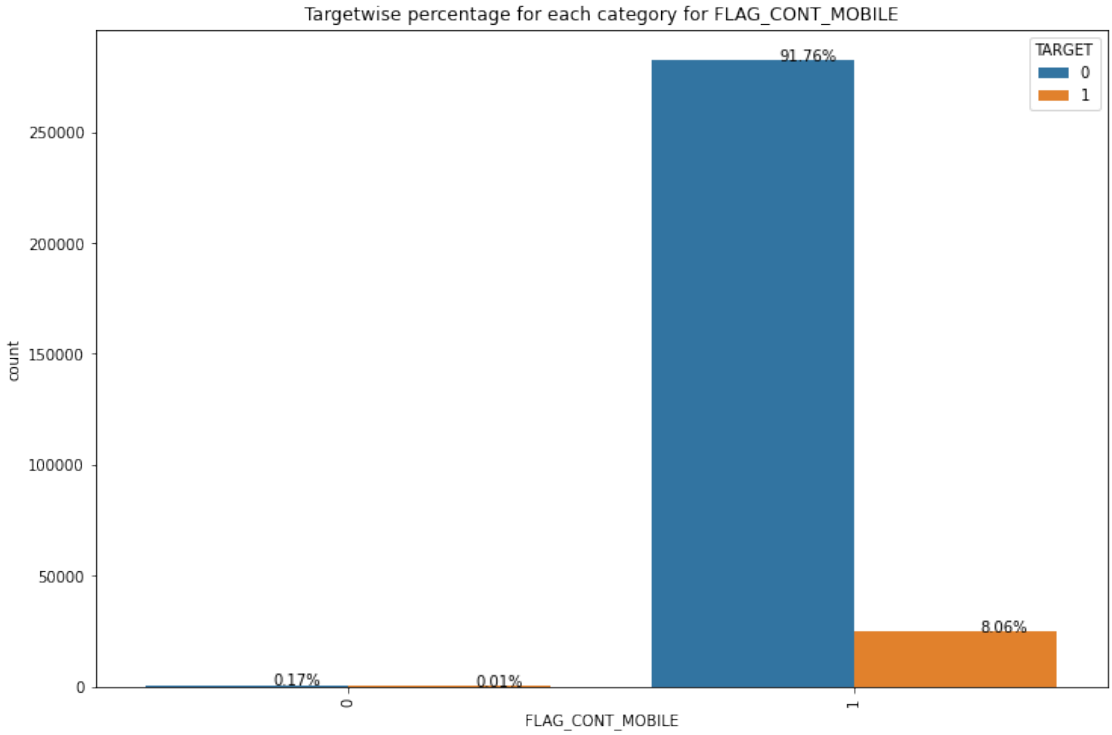
```
#lets see its distribution

ax=sns.countplot(x='FLAG_CONT_MOBILE',data=df_train)
ax.set_xticklabels(df_train['FLAG_CONT_MOBILE'].value_counts().index.t
o_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
```



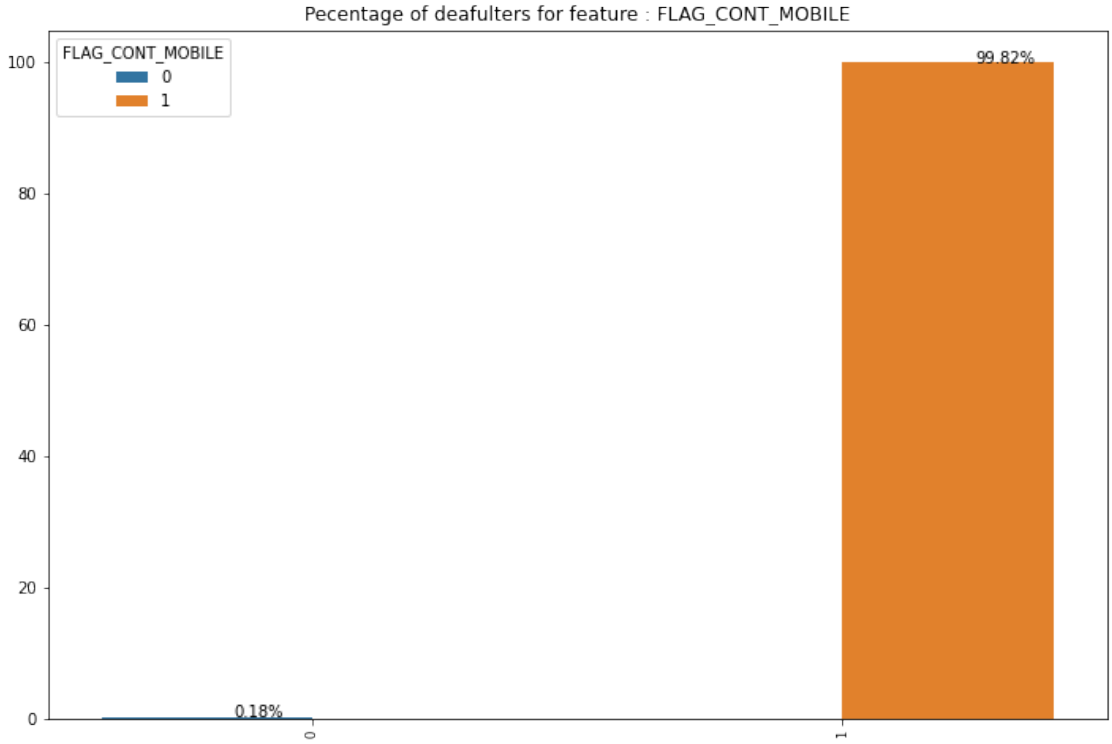
- very less people phone was not reachable ,we can drop this column as it have very low variance

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FLAG_CONT_MOBILE',hue='TARGET',data=df_train)
plt.xticks(rotation=90)
#ax.set_xticklabels(df_train['FLAG_EMP_PHONE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
FLAG_CONT_MOBILE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FLAG_CONT_MOBILE')
```




```
plot_for_categorical_defaulters(df_train,'FLAG_CONT_MOBILE')
```




```
df_train.corr()
[['FLAG_CONT_MOBILE']].sort_values('FLAG_CONT_MOBILE',ascending=False)
[:10]
```

```
                              FLAG_CONT_MOBILE
FLAG_CONT_MOBILE                      1.000000
AMT_REQ_CREDIT_BUREAU_YEAR            0.026700
AMT_CREDIT                            0.023653
AMT_ANNUITY                           0.022350
FLAG_WORK_PHONE                       0.021580
AMT_GOODS_PRICE                       0.020706
Client_Age                            0.014998
REGION_RATING_CLIENT_W_CITY           0.013651
REGION_RATING_CLIENT                  0.013041
DAYS_EMPLOYED                         0.012745
```

FLAG_EMAIL

DId client provided Email address

```
df_train['FLAG_EMAIL'].value_counts()
```

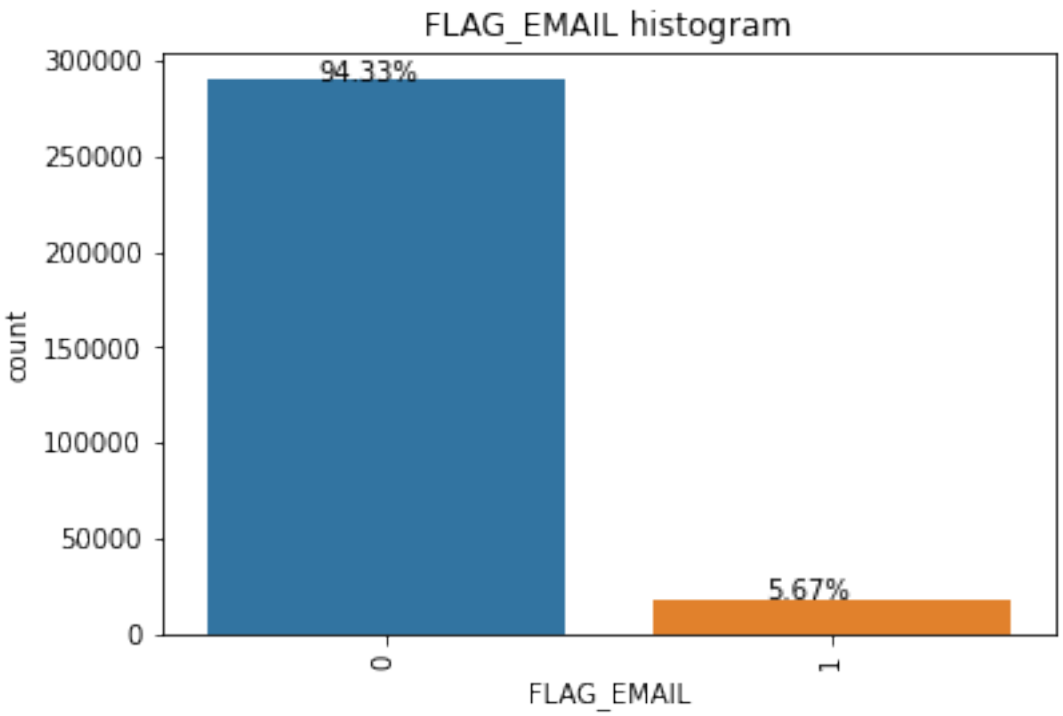
```
0    290069
1     17442
Name: FLAG_EMAIL, dtype: int64
```

```
df_train['FLAG_EMAIL'].isna().sum()
#have NA values
```

```
0
```

```
#lets see its distribution

ax=sns.countplot(x='FLAG_EMAIL',data=df_train)
ax.set_xticklabels(df_train['FLAG_EMAIL'].value_counts().index.to_list
(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("FLAG_EMAIL histogram")
```

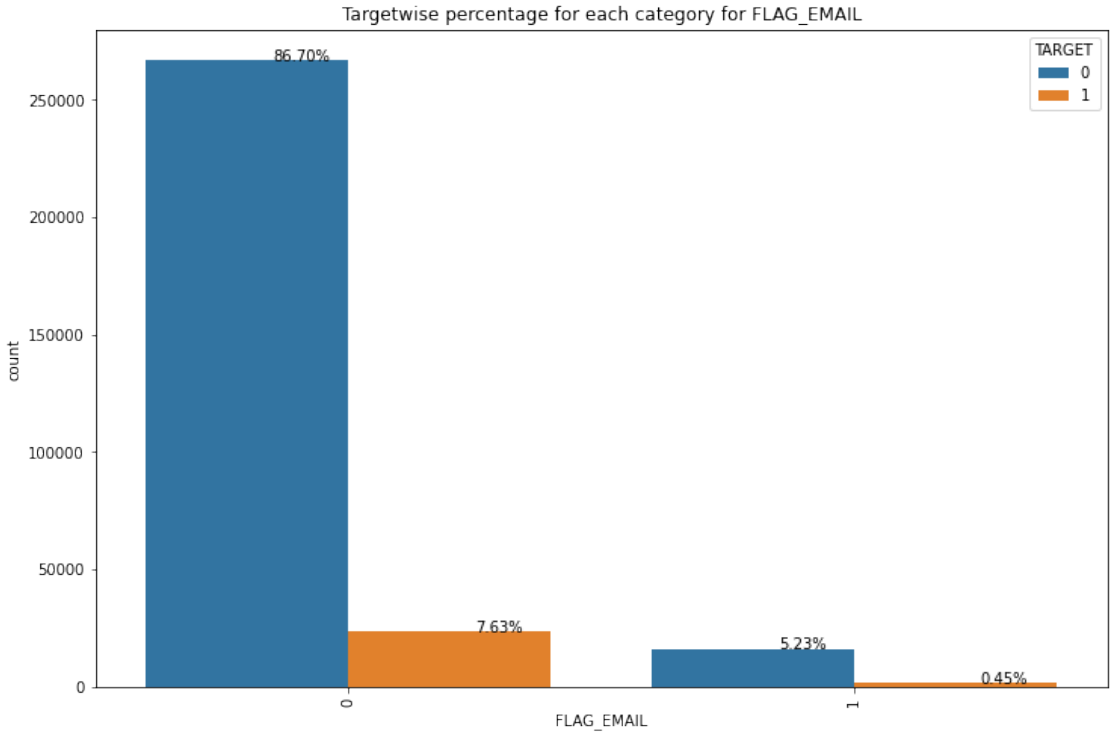
```
Text(0.5, 1.0, 'FLAG_EMAIL histogram')
```



- Very less people have Email id, Its normal behaviour

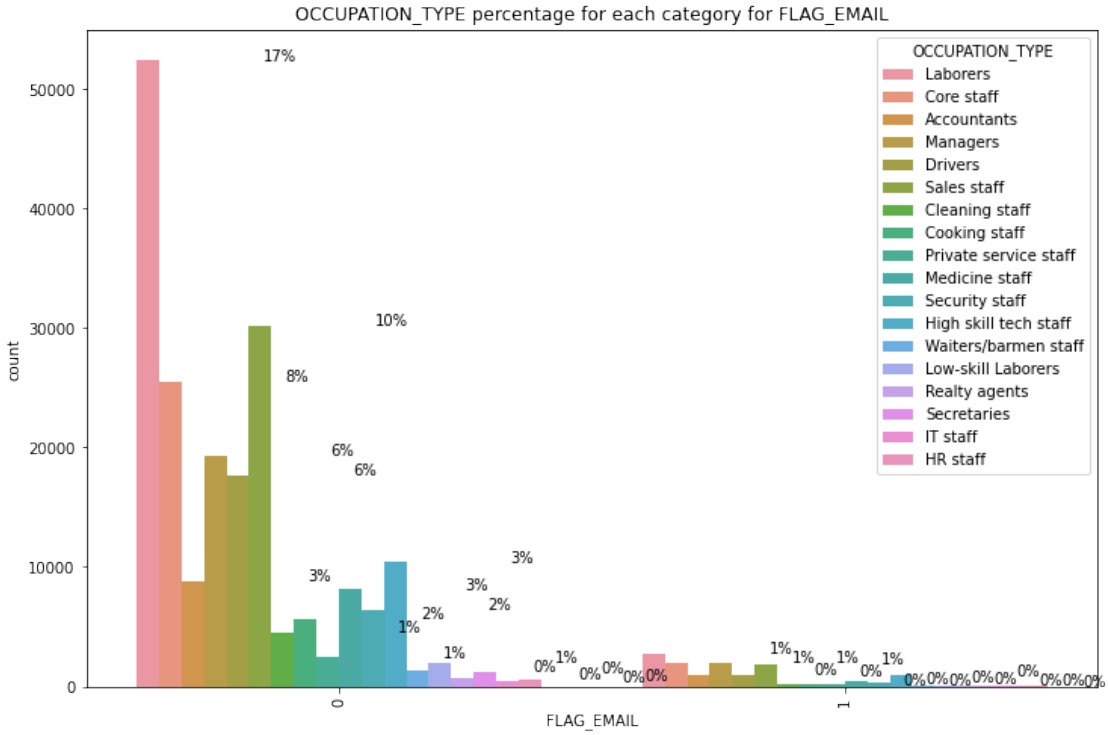
```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='FLAG_EMAIL',hue='TARGET',data=df_train)
plt.xticks(rotation=90)
#ax.set_xticklabels(df_train['FLAG_EMP_PHONE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Targetwise percentage for each category for FLAG_EMAIL")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
FLAG_EMAIL')
```

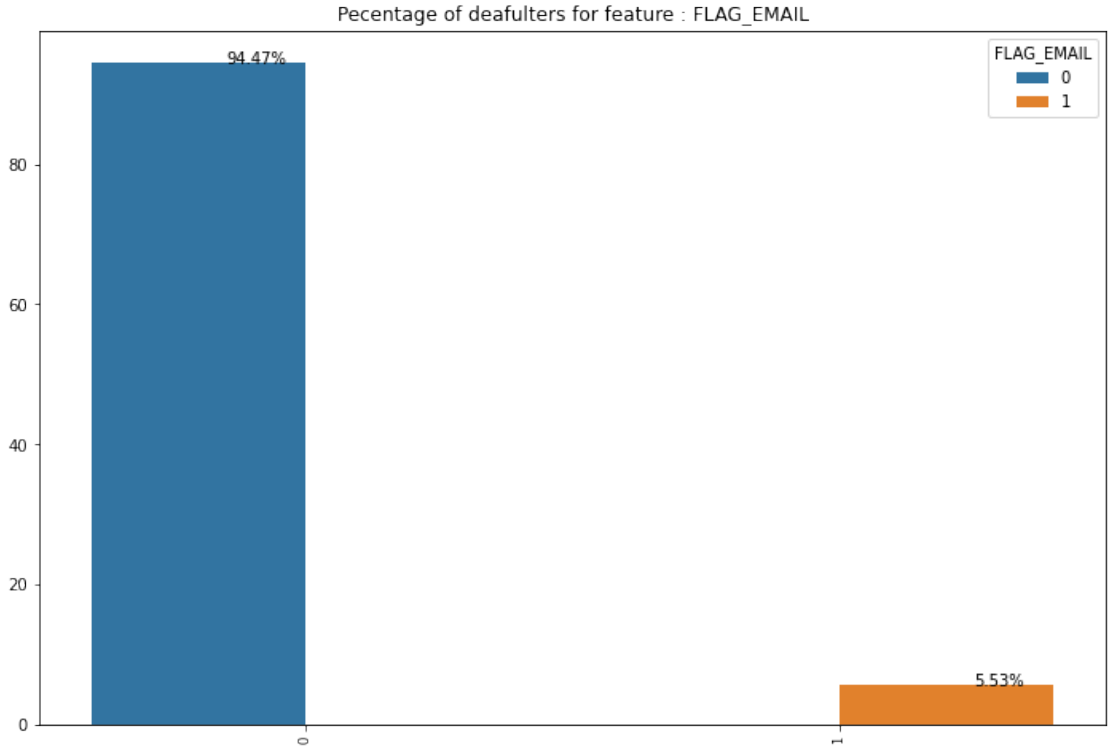



```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
sns.color_palette("Set1")
ax=sns.countplot(x='FLAG_EMAIL',hue='OCCUPATION_TYPE',data=df_train)
plt.xticks(rotation=90)
#ax.set_xticklabels(df_train['FLAG_EMP_PHONE'].value_counts().index.to
_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.0f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01),size=10)
ax.set_title("OCCUPATION_TYPE percentage for each category for
FLAG_EMAIL")
```

```
Text(0.5, 1.0, 'OCCUPATION_TYPE percentage for each category for
FLAG_EMAIL')
```




```
plot_for_categorical_defaulters(df_train,'FLAG_EMAIL')
```



- Most defaulters dont have provided Email address

```
#Lets go column by column and see How we can impute this columns.
# we will follow at down to top approach
# we will start imputing the data from least NA columns and go to next
columns
```

DAYS_LAST_PHONE_CHANGE

How many days before application did client change phone

```
pd.set_option('display.max_colwidth', None)
pd.set_option('display.max_columns', None)
```

```
app_data_col_Details[app_data_col_Details["Row"]=='DAYS_LAST_PHONE_CHA
NGE']['Description']
```

```
98    How many days before application did client change phone
Name: Description, dtype: object
```

```
df_train['DAYS_LAST_PHONE_CHANGE'].describe()
```

```
count   307510.000000
mean      -962.858788
std        826.808487
min      -4292.000000
25%      -1570.000000
50%       -757.000000
75%       -274.000000
max          0.000000
Name: DAYS_LAST_PHONE_CHANGE, dtype: float64
```

```
df_train['DAYS_LAST_PHONE_CHANGE'].isna().sum()
# have just 1 NA values,we can fill it with zero
#df_train[df_train['DAYS_LAST_PHONE_CHANGE'].isna()]
```

```
       SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR
\
15709      118330       0         Cash loans           M            Y


      FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL    AMT_CREDIT  \
15709               Y             0     126000.000000 278613.000000

       AMT_ANNUITY  AMT_GOODS_PRICE NAME_TYPE_SUITE
NAME_INCOME_TYPE  \
15709 25911.000000    252000.000000   Unaccompanied          Working


      NAME_EDUCATION_TYPE NAME_FAMILY_STATUS  NAME_HOUSING_TYPE  \
15709   Incomplete higher            Married  House / apartment

       REGION_POPULATION_RELATIVE  DAYS_BIRTH  DAYS_EMPLOYED  \
```

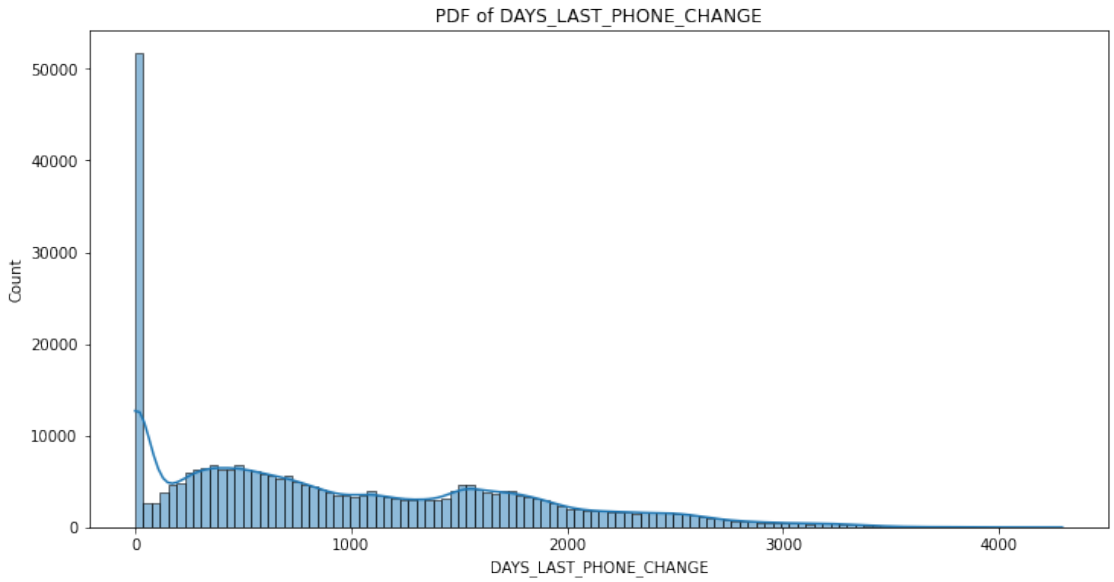
```
15709                    0.010643       -8574           -293

       DAYS_REGISTRATION  DAYS_ID_PUBLISH  OWN_CAR_AGE  FLAG_MOBIL  \
15709       -4790.000000            -1075    21.000000           0

       FLAG_EMP_PHONE  FLAG_WORK_PHONE  FLAG_CONT_MOBILE
FLAG_PHONE  \
15709               1                0                 1           0


       FLAG_EMAIL OCCUPATION_TYPE  CNT_FAM_MEMBERS
REGION_RATING_CLIENT  \
15709           0             NaN         2.000000
2

       REGION_RATING_CLIENT_W_CITY WEEKDAY_APPR_PROCESS_START  \
15709                            2                     FRIDAY

       HOUR_APPR_PROCESS_START  REG_REGION_NOT_LIVE_REGION  \
15709                       13                           0

       REG_REGION_NOT_WORK_REGION  LIVE_REGION_NOT_WORK_REGION  \
15709                           0                             0

       REG_CITY_NOT_LIVE_CITY  REG_CITY_NOT_WORK_CITY  \
15709                       0                       0

       LIVE_CITY_NOT_WORK_CITY ORGANIZATION_TYPE  EXT_SOURCE_1
EXT_SOURCE_2  \
15709                       0     Trade: type 3           NaN
NaN

       EXT_SOURCE_3  APARTMENTS_AVG  BASEMENTAREA_AVG  \
15709           NaN        0.123700          0.060100

       YEARS_BEGINEXPLUATATION_AVG  YEARS_BUILD_AVG  COMMONAREA_AVG  \
15709                     0.993000              NaN             NaN

       ELEVATORS_AVG  ENTRANCES_AVG  FLOORSMAX_AVG  FLOORSMIN_AVG  \
15709       0.120000       0.103400       0.375000            NaN

       LANDAREA_AVG  LIVINGAPARTMENTS_AVG  LIVINGAREA_AVG  \
15709      0.080900                   NaN        0.128300

       NONLIVINGAPARTMENTS_AVG  NONLIVINGAREA_AVG  APARTMENTS_MODE  \
15709                      NaN                NaN         0.126100

       BASEMENTAREA_MODE  YEARS_BEGINEXPLUATATION_MODE
YEARS_BUILD_MODE  \
```

```
15709           0.062400                      0.993000
NaN

       COMMONAREA_MODE  ELEVATORS_MODE  ENTRANCES_MODE  FLOORSMAX_MODE
\
15709              NaN        0.120800        0.103400        0.375000


       FLOORSMIN_MODE  LANDAREA_MODE  LIVINGAPARTMENTS_MODE
LIVINGAREA_MODE  \
15709            NaN       0.082800                    NaN
0.133700

       NONLIVINGAPARTMENTS_MODE  NONLIVINGAREA_MODE
APARTMENTS_MEDI  \
15709                       NaN                 NaN         0.124900


       BASEMENTAREA_MEDI  YEARS_BEGINEXPLUATATION_MEDI
YEARS_BUILD_MEDI  \
15709           0.060100                      0.993000
NaN

       COMMONAREA_MEDI  ELEVATORS_MEDI  ENTRANCES_MEDI  FLOORSMAX_MEDI
\
15709              NaN        0.120000        0.103400        0.375000


       FLOORSMIN_MEDI  LANDAREA_MEDI  LIVINGAPARTMENTS_MEDI
LIVINGAREA_MEDI  \
15709            NaN       0.082300                    NaN
0.130600

       NONLIVINGAPARTMENTS_MEDI  NONLIVINGAREA_MEDI FONDKAPREMONT_MODE
\
15709                       NaN                 NaN                NaN


       HOUSETYPE_MODE  TOTALAREA_MODE WALLSMATERIAL_MODE
EMERGENCYSTATE_MODE  \
15709  block of flats        0.106700              Panel
No

       OBS_30_CNT_SOCIAL_CIRCLE  DEF_30_CNT_SOCIAL_CIRCLE  \
15709                  0.000000                  0.000000

       OBS_60_CNT_SOCIAL_CIRCLE  DEF_60_CNT_SOCIAL_CIRCLE  \
15709                  0.000000                  0.000000
```

```
       DAYS_LAST_PHONE_CHANGE  FLAG_DOCUMENT_2  FLAG_DOCUMENT_3  \
15709                     NaN                0                0

       FLAG_DOCUMENT_4  FLAG_DOCUMENT_5  FLAG_DOCUMENT_6
FLAG_DOCUMENT_7  \
15709                0                0                0
0

       FLAG_DOCUMENT_8  FLAG_DOCUMENT_9  FLAG_DOCUMENT_10
FLAG_DOCUMENT_11  \
15709                1                0                 0
0

       FLAG_DOCUMENT_12  FLAG_DOCUMENT_13  FLAG_DOCUMENT_14
FLAG_DOCUMENT_15  \
15709                 0                 0                 0
0

       FLAG_DOCUMENT_16  FLAG_DOCUMENT_17  FLAG_DOCUMENT_18
FLAG_DOCUMENT_19  \
15709                 0                 0                 1
0

       FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  AMT_REQ_CREDIT_BUREAU_HOUR
\
15709                 0                 0                         NaN


       AMT_REQ_CREDIT_BUREAU_DAY  AMT_REQ_CREDIT_BUREAU_WEEK  \
15709                        NaN                         NaN

       AMT_REQ_CREDIT_BUREAU_MON  AMT_REQ_CREDIT_BUREAU_QRT  \
15709                        NaN                        NaN

       AMT_REQ_CREDIT_BUREAU_YEAR  Client_Age
15709                         NaN          23
```

```
#So we can clearly see that This client dont have Mobile phone,so we
can fill na with 0

# df_train['DAYS_LAST_PHONE_CHANGE'].fillna(0,inplace=True)
# df_test['DAYS_LAST_PHONE_CHANGE'].fillna(0,inplace=True)

#lets see its PDF and CDF
plt.figure(figsize=(12,6))
sns.histplot(-1*df_train['DAYS_LAST_PHONE_CHANGE'],kde=True)
#plt.xlim([40000,5000000])
plt.title("PDF of DAYS_LAST_PHONE_CHANGE")
```

```
Text(0.5, 1.0, 'PDF of DAYS_LAST_PHONE_CHANGE')
```




```
#zero have the highest frequnecy
```

```
len(df_train[df_train['DAYS_LAST_PHONE_CHANGE'].isin([0])])/
df_train.shape[0]
```

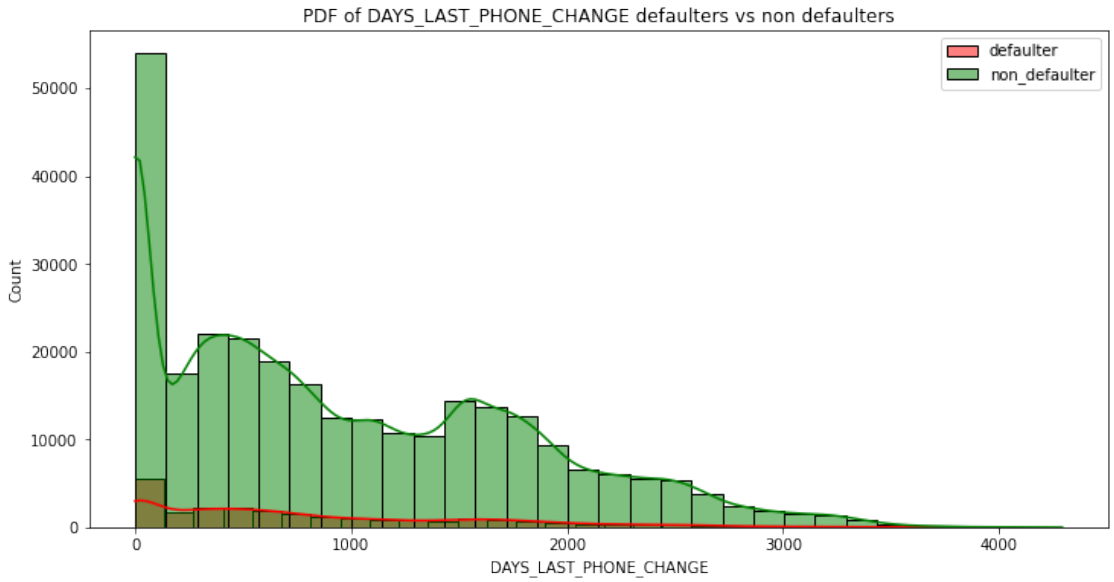
```
0.12250618677055455
```

- 12 % of the Client have this DAYS_LAST_PHONE_CHANGE value zero

```
# It looks like  with values frequnecy is decreasing (because we have
just taken -1*)
```

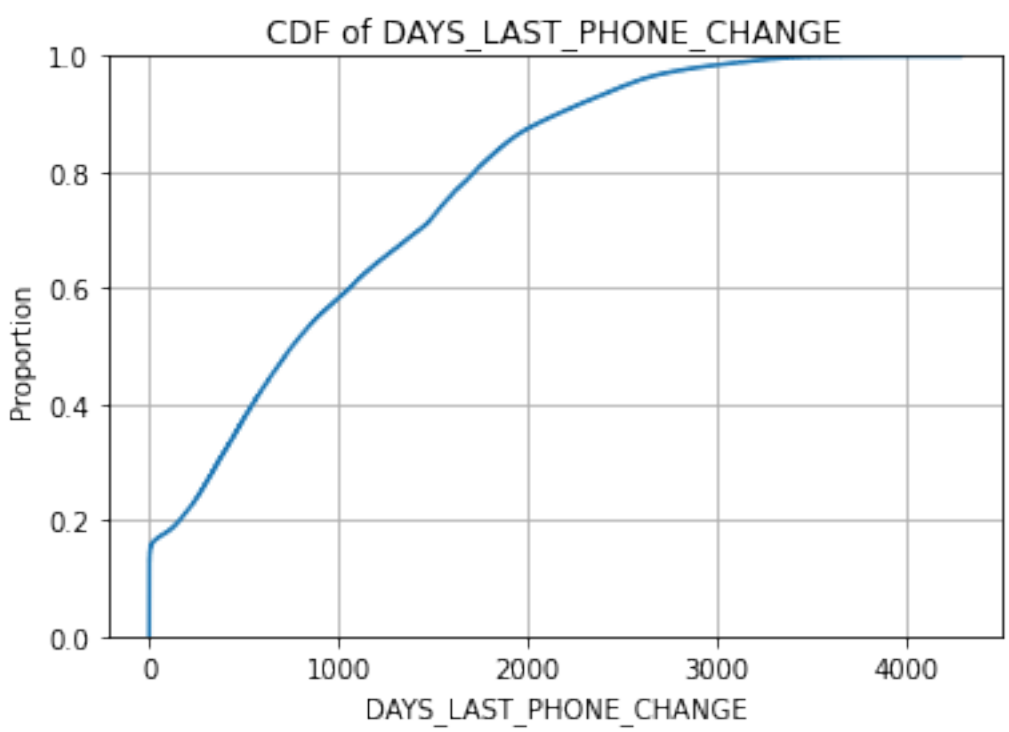
```
plt.figure(figsize=(12,6))
sns.histplot(-
1*df_defaulter['DAYS_LAST_PHONE_CHANGE'],bins=30,kde=True,label='defau
lter',color='red')
sns.histplot(-
1*df_non_defaulter['DAYS_LAST_PHONE_CHANGE'],bins=30,kde=True,label='n
on_defaulter',color="green")
#plt.xlim([40000,5000000])
plt.title("PDF of DAYS_LAST_PHONE_CHANGE defaulters vs non
defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e258ba8490>
```




```
#Most defaulters have simliar distributions for DAYS_LAST_PHONE_CHANGE

sns.ecdfplot(-1*df_train['DAYS_LAST_PHONE_CHANGE'])
#plt.xlim([40000,5000000])
plt.grid()
#plt.xticks(df_train['AMT_INCOME_TOTAL'].value_counts().index.to_list(
))
plt.title("CDF of DAYS_LAST_PHONE_CHANGE")
```

```
Text(0.5, 1.0, 'CDF of DAYS_LAST_PHONE_CHANGE')
```



```
np.quantile(-1*df_train['DAYS_LAST_PHONE_CHANGE'].dropna(),0.99)/365
```

```
8.627397260273973
```

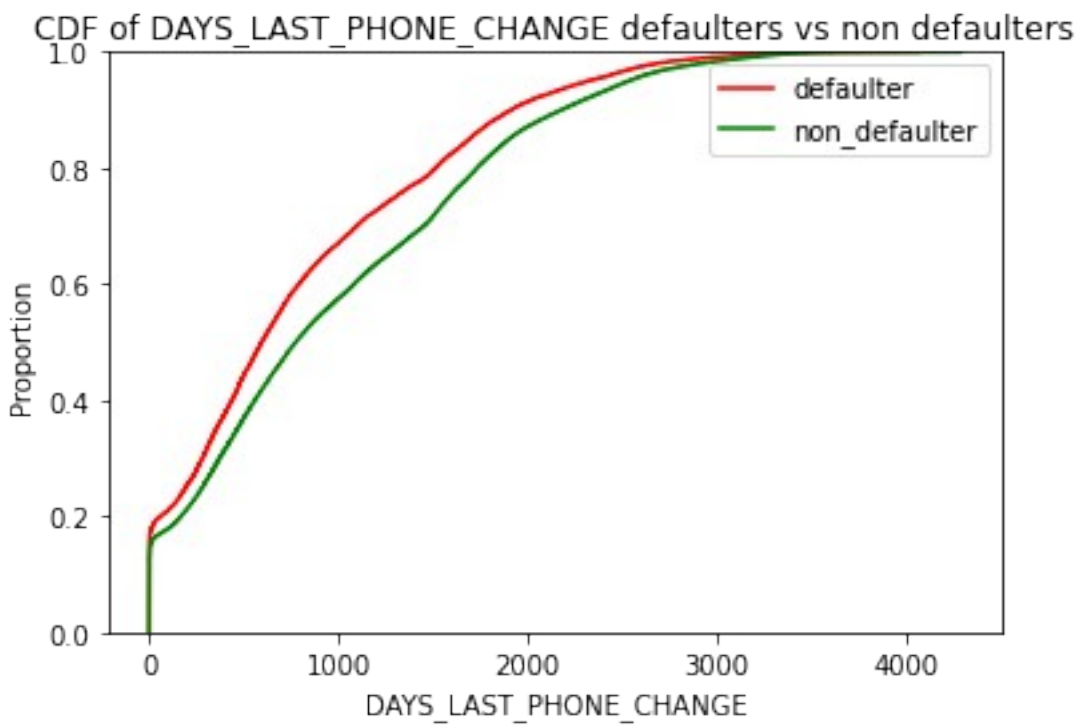
```
np.quantile(-1*df_train['DAYS_LAST_PHONE_CHANGE'].dropna(),0.80)/365
```

```
4.712328767123288
```

- 99% client seems to have DAYS_LAST_PHONE_CHANGE more than 8 years
- 80% client seems to have DAYS_LAST_PHONE_CHANGE more than 4 years

```
sns.ecdfplot(-
1*df_defaulter['DAYS_LAST_PHONE_CHANGE'],label='defaulter',color='red'
)
sns.ecdfplot(-
1*df_non_defaulter['DAYS_LAST_PHONE_CHANGE'],label='non_defaulter',col
or='green')
#plt.xlim([40000,1000000])
plt.title("CDF of DAYS_LAST_PHONE_CHANGE defaulters vs non
defaulters")
plt.legend()
#plt.axis('off')
```

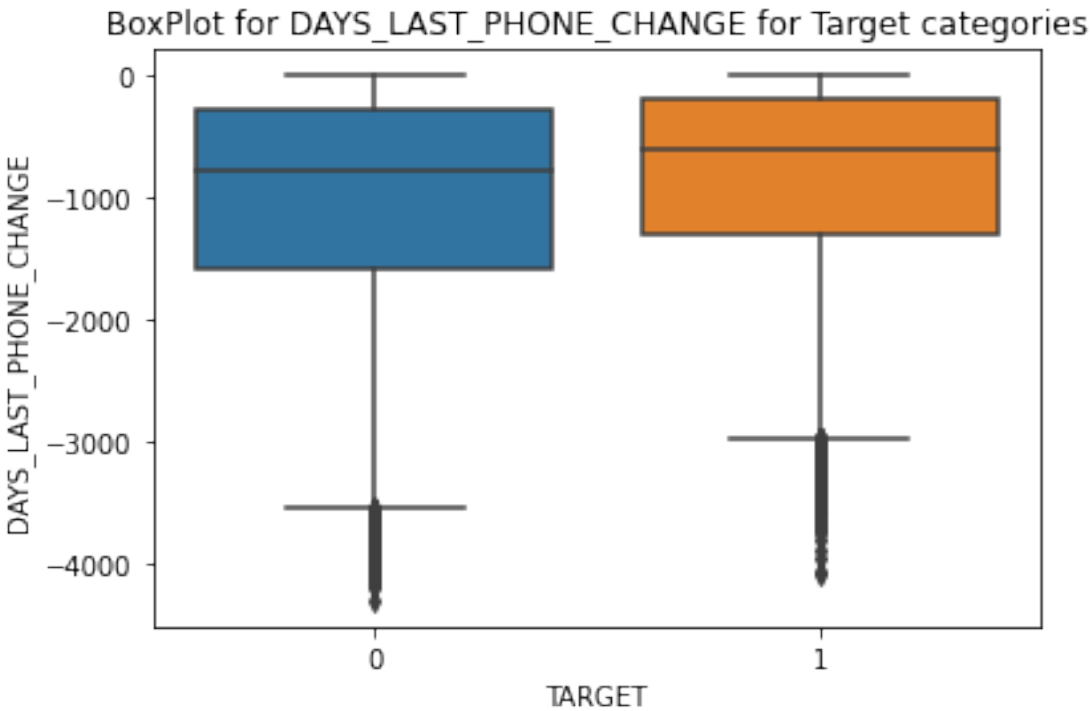
```
<matplotlib.legend.Legend at 0x1e28e737040>
```



- Avergae DAYS_LAST_PHONE_CHANGE of defaulters are lesser in compare to non defaulters

```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="DAYS_LAST_PHONE_CHANGE",
data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for DAYS_LAST_PHONE_CHANGE for Target categories")
```

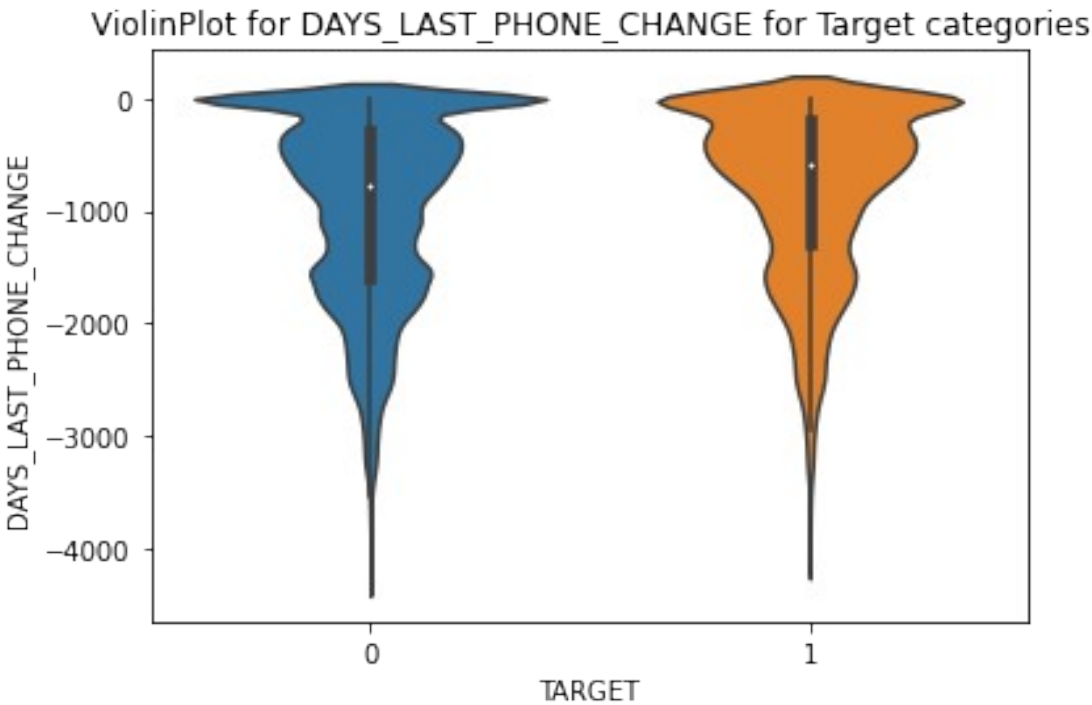
```
Text(0.5, 1.0, 'BoxPlot for DAYS_LAST_PHONE_CHANGE for Target
categories')
```



- deafaulters are supposed to be averagely lesser DAYS_LAST_PHONE_CHANGE than Non defaulters

```
#Violin Plot
ax = sns.violinplot(x="TARGET", y="DAYS_LAST_PHONE_CHANGE",
data=df_train)
#.ylim([40000,5000000])
plt.title("ViolinPlot for DAYS_LAST_PHONE_CHANGE for Target
categories")
#plt.legend()
```

```
Text(0.5, 1.0, 'ViolinPlot for DAYS_LAST_PHONE_CHANGE for Target
categories')
```




```
#top 10 correlated column with DAYS_REGISTRATION
df_train.corr()
[['DAYS_LAST_PHONE_CHANGE']].sort_values('DAYS_LAST_PHONE_CHANGE',asce
nding=False)[:10]
```

```
                        DAYS_LAST_PHONE_CHANGE
DAYS_LAST_PHONE_CHANGE                 1.000000
DAYS_ID_PUBLISH                        0.088576
DAYS_BIRTH                             0.082939
DAYS_REGISTRATION                      0.056983
TARGET                                 0.055218
REG_CITY_NOT_LIVE_CITY                 0.054183
FLAG_DOCUMENT_18                       0.047604
REG_CITY_NOT_WORK_CITY                 0.046788
FLAG_DOCUMENT_11                       0.044569
FLAG_DOCUMENT_16                       0.041698
```

CNT_FAM_MEMBERS

How many family members does client have

```
app_data_col_Details[app_data_col_Details["Row"]=='CNT_FAM_MEMBERS']
```

```
                Row                                Description
32  CNT_FAM_MEMBERS  How many family members does client have
```

```
df_train['CNT_FAM_MEMBERS'].isna().sum()
```

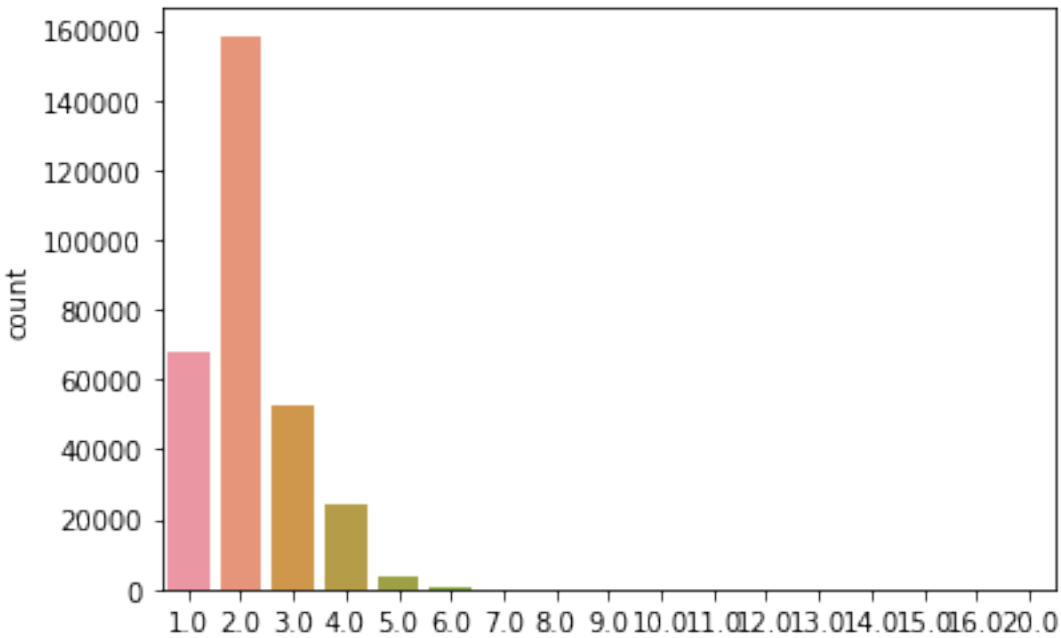
```
2
```

```
CNT_FAM_MEMBERS=df_train['CNT_FAM_MEMBERS'].values
```

```
#lets check its distribution
```

```
sns.countplot(CNT_FAM_MEMBERS)
```

```
<AxesSubplot:ylabel='count'>
```




```
# we can see that all values are below 5 but some outliers,Let see it
mode value
```

```
max(set(CNT_FAM_MEMBERS.tolist()), key=CNT_FAM_MEMBERS.tolist().count)
```

```
2.0
```

```
#This mean ,we can impute it with value Mode having NA values
from sklearn.impute import SimpleImputer

def fillna_CNT_FAM_MEMBERS(df):
    CNT_FAM_MEMBERS=df_train['CNT_FAM_MEMBERS'].values
    most_frequent_value=max(set(CNT_FAM_MEMBERS.tolist()),
key=CNT_FAM_MEMBERS.tolist().count)
    df['CNT_FAM_MEMBERS']=
df['CNT_FAM_MEMBERS'].replace(np.nan,most_frequent_value)
```

```
fillna_CNT_FAM_MEMBERS(df_train)
```

```
df_train['CNT_FAM_MEMBERS'].isna().sum()
```

```
0
```

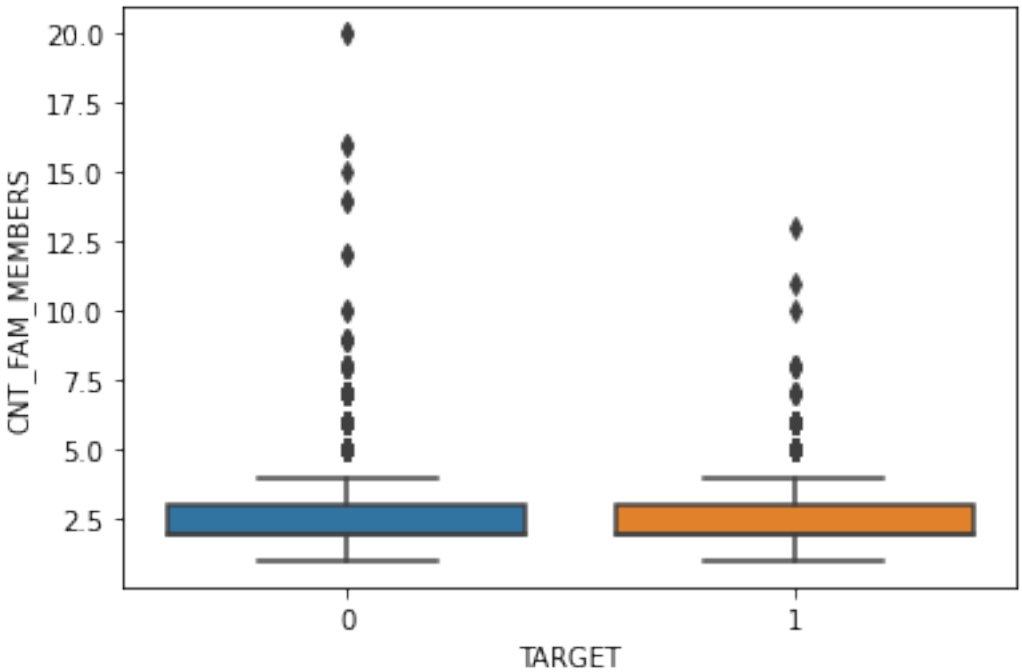
```
## Done !!
```

```
df_train[['TARGET','CNT_FAM_MEMBERS']].corr()['TARGET']
```

```
TARGET            1.000000
CNT_FAM_MEMBERS   0.009308
Name: TARGET, dtype: float64
```

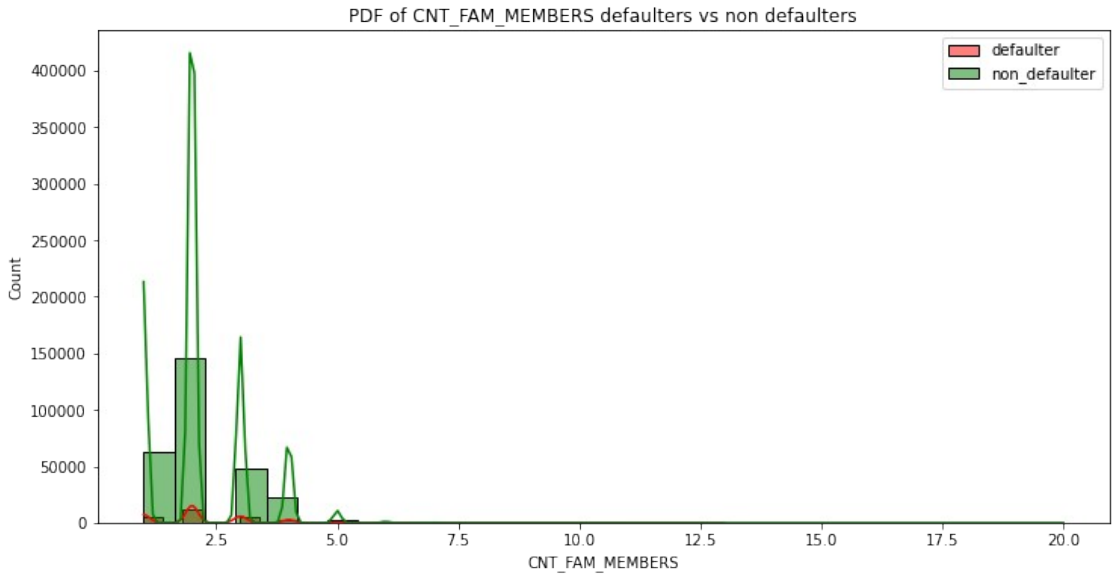
```
sns.boxplot(x='TARGET',y='CNT_FAM_MEMBERS',data=df_train)
```

```
<AxesSubplot:xlabel='TARGET', ylabel='CNT_FAM_MEMBERS'>
```



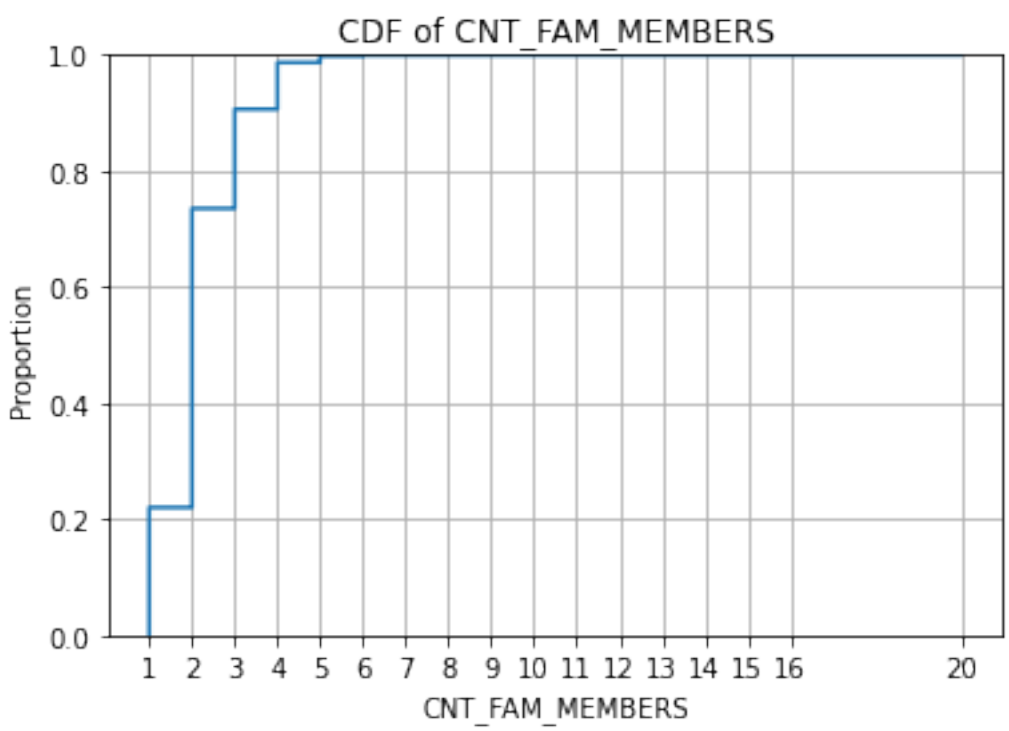

```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['CNT_FAM_MEMBERS'],bins=30,kde=True,label='d
efaulter',color='red')
sns.histplot(df_non_defaulter['CNT_FAM_MEMBERS'],bins=30,kde=True,labe
l='non_defaulter',color="green")
#plt.xlim([40000,5000000])
plt.title("PDF of CNT_FAM_MEMBERS defaulters vs non defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e2120d5a60>
```




```
sns.ecdfplot(df_train['CNT_FAM_MEMBERS'])
#plt.xlim([40000,5000000])
plt.grid()
plt.xticks(df_train['CNT_FAM_MEMBERS'].value_counts().index.to_list())
plt.title("CDF of CNT_FAM_MEMBERS")
```

```
Text(0.5, 1.0, 'CDF of CNT_FAM_MEMBERS')
```




```
np.quantile(df_train['CNT_FAM_MEMBERS'].dropna(),0.99)
```

```
5.0
```

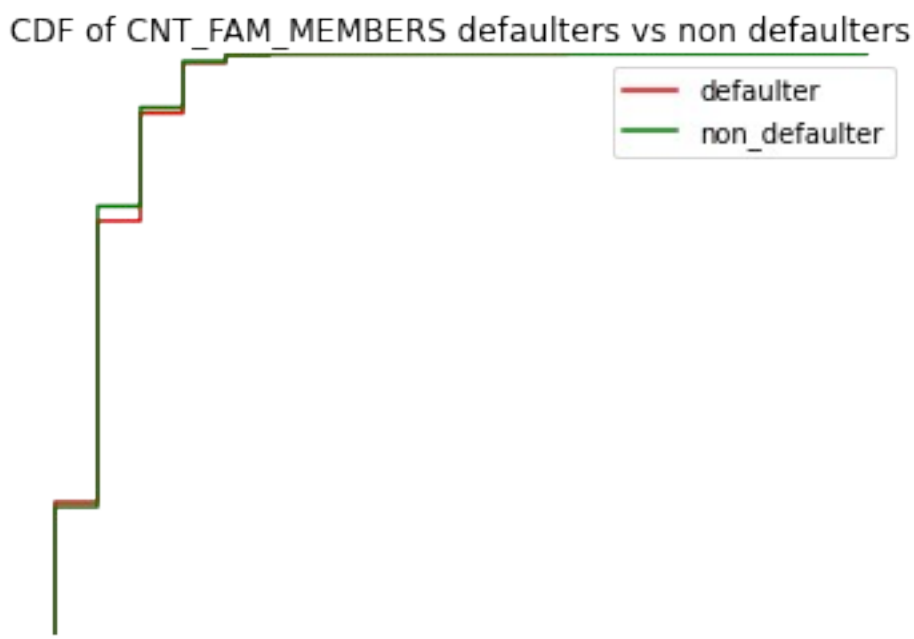
```
np.quantile(df_train['CNT_FAM_MEMBERS'].dropna(),0.80)
```

```
3.0
```

- 99% client seems to have CNT_FAM_MEMBERS below 5
- 80% client seems to have CNT_FAM_MEMBERS below 3

```
sns.ecdfplot(1*df_defaulter['CNT_FAM_MEMBERS'],label='defaulter',color
='red')
sns.ecdfplot(df_non_defaulter['CNT_FAM_MEMBERS'],label='non_defaulter'
,color='green')
#plt.xlim([40000,1000000])
plt.title("CDF of CNT_FAM_MEMBERS defaulters vs non defaulters")
plt.legend()
plt.axis('off')
```

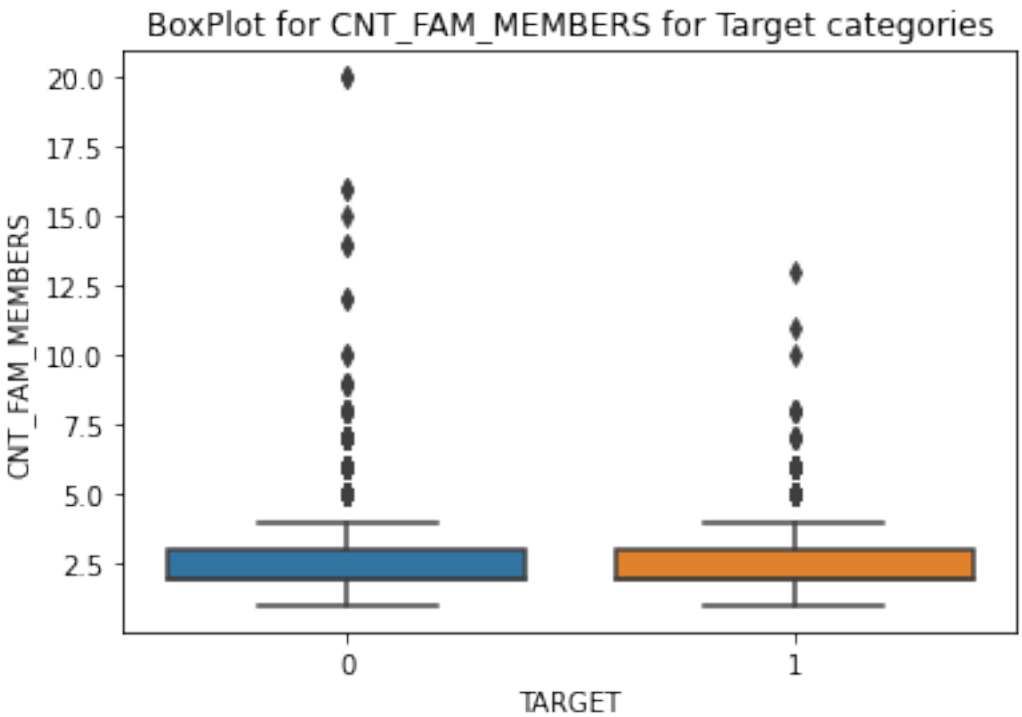
```
(0.04999999999999993, 20.95, 0.0, 1.0)
```



- Both have almost same distribution

```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="CNT_FAM_MEMBERS", data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for CNT_FAM_MEMBERS for Target categories")
```

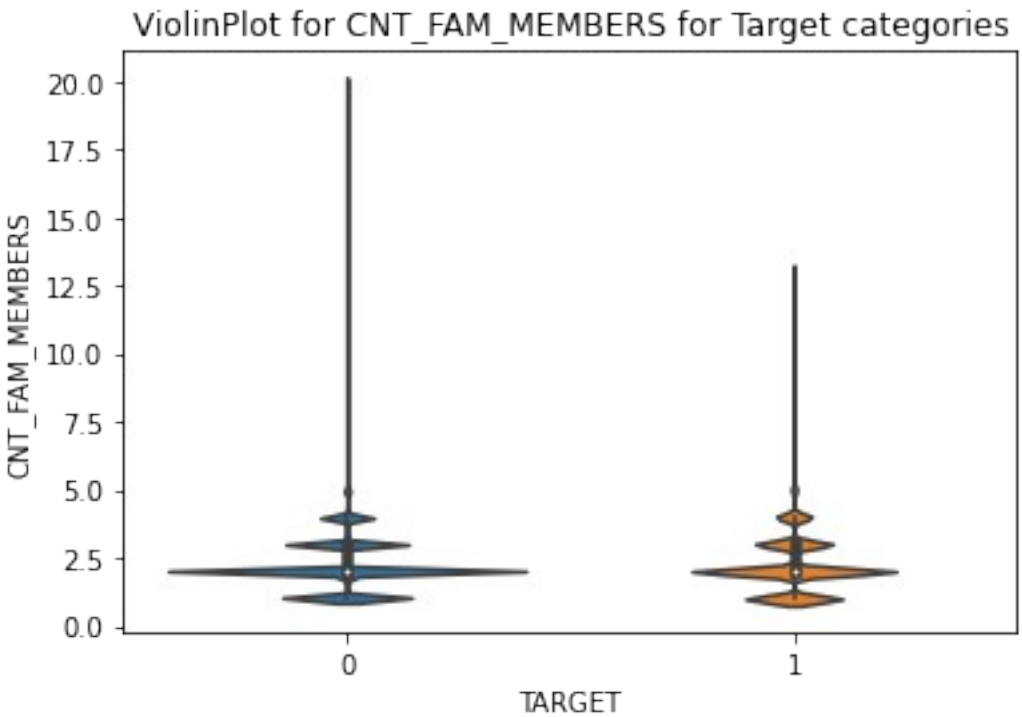
```
Text(0.5, 1.0, 'BoxPlot for CNT_FAM_MEMBERS for Target categories')
```



- No abstract information that CLient having more Family members have nore problem in repayment

```
#Violin Plot
ax = sns.violinplot(x="TARGET", y="CNT_FAM_MEMBERS", data=df_train)
#.ylim([40000,5000000])
plt.title("ViolinPlot for CNT_FAM_MEMBERS for Target categories")
#plt.legend()
```

```
Text(0.5, 1.0, 'ViolinPlot for CNT_FAM_MEMBERS for Target categories')
```




```
#top 10 correlated column with DAYS_REGISTRATION
df_train.corr()
[['CNT_FAM_MEMBERS']].sort_values('CNT_FAM_MEMBERS',ascending=False)
[:10]
```

```
                        CNT_FAM_MEMBERS
CNT_FAM_MEMBERS                1.000000
CNT_CHILDREN                   0.879160
DAYS_BIRTH                     0.278892
FLAG_EMP_PHONE                 0.233950
DAYS_REGISTRATION              0.173415
LIVE_CITY_NOT_WORK_CITY        0.077496
AMT_ANNUITY                    0.075539
REG_CITY_NOT_WORK_CITY         0.070719
FLAG_WORK_PHONE                0.067189
FLAG_DOCUMENT_3                0.066131
```

-> Higly correlated with Number of children and slightly correlated CLient Age bith column

AMT_ANNUITY

```
app_data_col_Details[app_data_col_Details["Row"]=='AMT_ANNUITY']
```

```
            Row   Description
12  AMT_ANNUITY  Loan annuity
```

Annuities are basically loans that are paid back over a set period of time at a set interest rate with consistent payments each period. A mortgage or car loan are simple examples of an annuity.

```
df_train['AMT_ANNUITY'].isna().sum()
```

```
12
```

```
#Lets see

#df_train[df_train['AMT_ANNUITY'].isna()]

df_test[df_test['AMT_ANNUITY'].isna()]
```

```
       SK_ID_CURR NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR
FLAG_OWN_REALTY  \
896        106206         Cash loans           F            N
N
1145       108106         Cash loans           F            N
Y
2277       116047         Cash loans           M            Y
Y
4141       129535         Cash loans           F            N
N
6791       149918         Cash loans           F            N
N
6985       151100         Cash loans           F            N
Y
7307       153420         Cash loans           M            N
Y
8207       160119         Cash loans           M            Y
Y
9246       167328         Cash loans           M            N
N
9830       171359         Cash loans           F            Y
N
10779      178317         Cash loans           F            N
N
10861      178888         Cash loans           F            N
N
11591      184238         Cash loans           F            N
Y
13387      197167         Cash loans           M            N
Y
13564      198247         Cash loans           F            N
N
13738      199378         Cash loans           M            N
Y
14008      201254         Cash loans           M            N
N
22146      260949         Cash loans           F            N
N
22663      264782         Cash loans           F            N
N
26027      289463         Cash loans           M            N
```

```
Y
26046      289634         Cash loans           M            Y
N
33294      342701         Cash loans           M            N
N
43434      416265         Cash loans           F            N
Y
44210      423135         Cash loans           F            N
N

       CNT_CHILDREN  AMT_INCOME_TOTAL     AMT_CREDIT  AMT_ANNUITY  \
896               2     270000.000000  790830.000000          NaN
1145              1      81000.000000  675000.000000          NaN
2277              0     270000.000000  441000.000000          NaN
4141              0     135000.000000  364896.000000          NaN
6791              0     180000.000000 1350000.000000          NaN
6985              0     315000.000000  886320.000000          NaN
7307              0     225000.000000  319500.000000          NaN
8207              0     315000.000000  440784.000000          NaN
9246              1     157500.000000  792162.000000          NaN
9830              0     450000.000000 1800000.000000          NaN
10779             0     157500.000000  714001.500000          NaN
10861             1     135000.000000  256500.000000          NaN
11591             0     180000.000000  171000.000000          NaN
13387             0     112500.000000  234576.000000          NaN
13564             1     157500.000000  104256.000000          NaN
13738             0     315000.000000  126000.000000          NaN
14008             0      76500.000000  220288.500000          NaN
22146             0     112500.000000  225000.000000          NaN
22663             0      67500.000000  136260.000000          NaN
26027             0     157500.000000  781920.000000          NaN
26046             0     225000.000000 1024740.000000          NaN
33294             0     292500.000000  504306.000000          NaN
43434             0     121500.000000  121747.500000          NaN
44210             2     112500.000000  301464.000000          NaN

       AMT_GOODS_PRICE NAME_TYPE_SUITE      NAME_INCOME_TYPE  \
896      675000.000000   Unaccompanied  Commercial associate
1145     675000.000000   Unaccompanied               Working
2277     441000.000000   Unaccompanied               Working
4141     315000.000000   Unaccompanied               Working
6791    1350000.000000   Unaccompanied             Pensioner
6985     720000.000000   Unaccompanied         State servant
7307     319500.000000   Unaccompanied  Commercial associate
8207     360000.000000   Unaccompanied               Working
9246     630000.000000   Unaccompanied  Commercial associate
9830    1800000.000000   Unaccompanied  Commercial associate
10779    589500.000000   Unaccompanied               Working
10861    256500.000000          Family               Working
11591    171000.000000          Family             Pensioner
```

13387	202500.000000	Unaccompanied	Pensioner
13564	90000.000000	Unaccompanied	Commercial associate
13738	126000.000000	Family	Commercial associate
14008	167292.000000	Unaccompanied	Working
22146	225000.000000	Unaccompanied	Commercial associate
22663	112500.000000	Unaccompanied	State servant
26027	675000.000000	Unaccompanied	Working
26046	900000.000000	Unaccompanied	Working
33294	445500.000000	Unaccompanied	Working
43434	92457.000000	Unaccompanied	Commercial associate
44210	238500.000000	Unaccompanied	Working

	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS \
896	Higher education	Married
1145	Higher education	Married
2277	Incomplete higher	Married
4141	Secondary / secondary special	Civil marriage
6791	Higher education	Widow
6985	Secondary / secondary special	Married
7307	Secondary / secondary special	Separated
8207	Higher education	Married
9246	Secondary / secondary special	Separated
9830	Higher education	Married
10779	Secondary / secondary special	Married
10861	Incomplete higher	Married
11591	Higher education	Married
13387	Secondary / secondary special	Civil marriage
13564	Secondary / secondary special	Civil marriage
13738	Secondary / secondary special	Civil marriage
14008	Secondary / secondary special	Single / not married
22146	Secondary / secondary special	Single / not married
22663	Secondary / secondary special	Single / not married
26027	Secondary / secondary special	Married
26046	Higher education	Married
33294	Secondary / secondary special	Single / not married
43434	Higher education	Separated
44210	Secondary / secondary special	Married

	NAME_HOUSING_TYPE	REGION_POPULATION_RELATIVE	DAYS_BIRTH \
896	Rented apartment	0.010643	-11344
1145	House / apartment	0.035792	-9824
2277	With parents	0.046220	-8856
4141	House / apartment	0.035792	-17386
6791	House / apartment	0.046220	-20168
6985	House / apartment	0.046220	-18028
7307	House / apartment	0.030755	-16134
8207	Rented apartment	0.026392	-13101
9246	House / apartment	0.008068	-13212
9830	House / apartment	0.072508	-12805
10779	House / apartment	0.046220	-18566

10861	With parents	0.018850	-9893
11591	House / apartment	0.026392	-22235
13387	House / apartment	0.031329	-23792
13564	House / apartment	0.046220	-14308
13738	House / apartment	0.026392	-17284
14008	House / apartment	0.035792	-18080
22146	Municipal apartment	0.046220	-13109
22663	Co-op apartment	0.035792	-8482
26027	House / apartment	0.018850	-11554
26046	House / apartment	0.072508	-19594
33294	House / apartment	0.008230	-9950
43434	House / apartment	0.035792	-13875
44210	House / apartment	0.031329	-13675

	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH	OWN_CAR_AGE
\				
896	-1582	-678.000000	-434	NaN
1145	-1727	-4121.000000	-1988	NaN
2277	-1397	-8837.000000	-1403	7.000000
4141	-5645	-9184.000000	-822	NaN
6791	365243	-2068.000000	-2068	NaN
6985	-978	-7173.000000	-1560	NaN
7307	-108	-4229.000000	-4201	NaN
8207	-343	-7146.000000	-4178	28.000000
9246	-2117	-3602.000000	-4386	NaN
9830	-1105	-1918.000000	-4706	3.000000
10779	-1148	-6067.000000	-2092	NaN
10861	-572	-9869.000000	-59	NaN
11591	365243	-1995.000000	-4811	NaN
13387	365243	-3154.000000	-4854	NaN
13564	-1240	-6111.000000	-3724	NaN
13738	-260	-655.000000	-669	NaN

14008	-1699	-2276.000000	-1624	NaN
22146	-314	-7247.000000	-3627	NaN
22663	-786	-4642.000000	-1156	NaN
26027	-4159	-5324.000000	-3706	NaN
26046	-1003	-584.000000	-1963	1.000000
33294	-1057	-676.000000	-2639	NaN
43434	-1129	-1696.000000	-4724	NaN
44210	-1838	-2157.000000	-141	NaN

	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE	FLAG_CONT_MOBILE \
896	1	1	1	1
1145	1	1	1	1
2277	1	1	0	1
4141	1	1	1	1
6791	1	0	0	1
6985	1	1	0	1
7307	1	1	1	1
8207	1	1	1	1
9246	1	1	1	1
9830	1	1	0	1
10779	1	1	0	1
10861	1	1	1	1
11591	1	0	0	1
13387	1	0	0	1
13564	1	1	1	1

13738	1	1	1	1
14008	1	1	1	1
22146	1	1	1	1
22663	1	1	1	1
26027	1	1	0	1
26046	1	1	0	1
33294	1	1	0	1
43434	1	1	1	1
44210	1	1	0	1

	FLAG_PHONE	FLAG_EMAIL	OCCUPATION_TYPE	CNT_FAM_MEMBERS
\				
896	0	0	NaN	4.000000
1145	1	0	Accountants	3.000000
2277	0	0	Laborers	2.000000
4141	0	0	Cooking staff	2.000000
6791	1	0	NaN	1.000000
6985	1	1	Laborers	2.000000
7307	1	0	Drivers	1.000000
8207	0	0	NaN	2.000000
9246	1	0	Drivers	2.000000
9830	1	0	Accountants	2.000000
10779	1	0	Cleaning staff	2.000000
10861	0	0	Laborers	3.000000
11591	1	0	NaN	2.000000

13387	0	0	NaN	2.000000
13564	1	0	Sales staff	3.000000
13738	0	0	NaN	2.000000
14008	0	0	Laborers	1.000000
22146	1	0	High skill tech staff	1.000000
22663	1	0	Core staff	1.000000
26027	1	0	Security staff	2.000000
26046	1	0	Drivers	2.000000
33294	0	0	Low-skill Laborers	1.000000
43434	1	0	NaN	1.000000
44210	0	0	Security staff	4.000000

	REGION_RATING_CLIENT	REGION_RATING_CLIENT_W_CITY	\
896	2	2	
1145	2	2	
2277	1	1	
4141	2	2	
6791	1	1	
6985	1	1	
7307	2	2	
8207	2	2	
9246	3	3	
9830	1	1	
10779	1	1	
10861	2	2	
11591	2	2	
13387	2	2	
13564	1	1	
13738	2	2	
14008	2	2	
22146	1	1	
22663	2	2	
26027	2	2	
26046	1	1	
33294	2	2	
43434	2	2	
44210	2	2	

	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCESS_START	\
896	MONDAY	17	
1145	MONDAY	11	
2277	MONDAY	9	
4141	FRIDAY	12	
6791	WEDNESDAY	12	
6985	THURSDAY	17	
7307	WEDNESDAY	10	
8207	TUESDAY	12	
9246	FRIDAY	9	
9830	MONDAY	9	
10779	SATURDAY	19	
10861	FRIDAY	12	
11591	FRIDAY	14	
13387	FRIDAY	12	
13564	THURSDAY	16	
13738	TUESDAY	11	
14008	WEDNESDAY	9	
22146	THURSDAY	19	
22663	WEDNESDAY	10	
26027	FRIDAY	18	
26046	THURSDAY	14	
33294	WEDNESDAY	14	
43434	THURSDAY	10	
44210	WEDNESDAY	8	

	REG_REGION_NOT_LIVE_REGION	REG_REGION_NOT_WORK_REGION	\
896	1	1	
1145	0	0	
2277	0	0	
4141	1	1	
6791	0	0	
6985	0	1	
7307	0	0	
8207	1	1	
9246	0	0	
9830	0	0	
10779	0	0	
10861	0	0	
11591	0	0	
13387	0	0	
13564	0	0	
13738	0	1	
14008	0	0	
22146	0	0	
22663	0	0	
26027	0	0	
26046	0	0	
33294	0	1	
43434	0	0	

44210	0	0

	LIVE_REGION_NOT_WORK_REGION	REG_CITY_NOT_LIVE_CITY	\
896	1	1	
1145	0	0	
2277	0	0	
4141	0	1	
6791	0	0	
6985	1	0	
7307	0	0	
8207	0	1	
9246	0	0	
9830	0	0	
10779	0	0	
10861	0	0	
11591	0	0	
13387	0	0	
13564	0	1	
13738	1	0	
14008	0	0	
22146	0	0	
22663	0	0	
26027	0	0	
26046	0	0	
33294	1	0	
43434	0	0	
44210	0	0	

	REG_CITY_NOT_WORK_CITY	LIVE_CITY_NOT_WORK_CITY	\
896	1	1	
1145	0	0	
2277	1	1	
4141	1	0	
6791	0	0	
6985	0	0	
7307	0	0	
8207	1	1	
9246	0	0	
9830	0	0	
10779	0	0	
10861	0	0	
11591	0	0	
13387	0	0	
13564	1	0	
13738	1	1	
14008	1	1	
22146	0	0	
22663	1	1	
26027	0	0	
26046	0	0	

```
33294                       1                         1
43434                       0                         0
44210                       0                         0

            ORGANIZATION_TYPE  EXT_SOURCE_1  EXT_SOURCE_2
EXT_SOURCE_3  \
896    Business Entity Type 1      0.346479      0.692297
0.234015
1145                   School      0.640190      0.687088
0.524496
2277         Industry: type 7      0.223027      0.596802
0.588488
4141            Self-employed           NaN      0.526072
0.141992
6791                      XNA           NaN      0.715429
0.391055
6985               University      0.760584      0.689261
0.337673
7307        Transport: type 4           NaN      0.755102
0.283712
8207                 Services           NaN      0.534607
NaN
9246   Business Entity Type 3      0.109529      0.222793
0.758393
9830                 Services      0.590508      0.600339
0.424130
10779              Government           NaN      0.622158
0.627991
10861           Trade: type 7      0.284242      0.646769
0.024744
11591                     XNA           NaN      0.691042
0.747663
13387                     XNA           NaN      0.502801
0.798137
13564       Transport: type 4      0.263366      0.634567
0.397946
13738  Business Entity Type 3      0.504568      0.630251
0.783832
14008  Business Entity Type 1           NaN      0.674962
0.492060
22146  Business Entity Type 3      0.436711      0.622626
NaN
22663     Security Ministries      0.153173      0.312224
0.155689
26027  Business Entity Type 3      0.333596      0.496826
0.484851
26046           Self-employed      0.613013      0.698748
0.365617
33294            Construction           NaN      0.315913
NaN
```

```
43434           Trade: type 3      0.327669      0.607875
0.639708
44210                  School      0.652113      0.638351
0.474051

       APARTMENTS_AVG  BASEMENTAREA_AVG
YEARS_BEGINEXPLUATATION_AVG  \
896          0.016500               NaN                      0.977100

1145              NaN               NaN                           NaN

2277         0.148500          0.115700                      0.984600

4141              NaN               NaN                           NaN

6791              NaN               NaN                           NaN

6985              NaN               NaN                           NaN

7307         0.020600          0.002700                      0.937400

8207         0.148500               NaN                      0.982600

9246         0.096900          0.067100                      0.985100

9830              NaN               NaN                           NaN

10779        0.027800          0.046900                      0.985600

10861        0.334000          0.211700                      0.984600

11591             NaN               NaN                           NaN

13387             NaN               NaN                           NaN

13564        0.224700          0.011100                      0.983600

13738        0.138500          0.139300                      0.989100

14008             NaN               NaN                           NaN

22146        0.016500          0.000000                      0.978600

22663             NaN               NaN                           NaN

26027             NaN               NaN                           NaN

26046        0.147400          0.095200                      0.979600
```

33294	NaN	NaN	NaN
43434	NaN	NaN	NaN
44210	0.024100	0.003200	0.966200

	YEARS_BUILD_AVG	COMMONAREA_AVG	ELEVATORS_AVG	ENTRANCES_AVG \
896	NaN	NaN	0.000000	0.069000
1145	NaN	NaN	NaN	NaN
2277	NaN	NaN	0.160000	0.137900
4141	NaN	NaN	NaN	NaN
6791	NaN	NaN	NaN	NaN
6985	NaN	NaN	NaN	NaN
7307	0.143200	0.000000	0.000000	0.034500
8207	NaN	NaN	0.160000	0.137900
9246	NaN	NaN	0.000000	0.069000
9830	NaN	NaN	NaN	NaN
10779	0.802800	NaN	0.000000	0.103400
10861	0.789200	0.070700	0.360000	0.310300
11591	NaN	NaN	NaN	NaN
13387	NaN	NaN	NaN	NaN
13564	0.775600	0.036500	0.240000	0.206900
13738	NaN	NaN	0.000000	0.229700
14008	NaN	NaN	NaN	NaN
22146	0.707600	NaN	0.000000	0.069000
22663	NaN	NaN	NaN	NaN

26027	NaN	NaN	NaN	NaN
26046	0.721200	0.060100	0.160000	0.137900
33294	NaN	NaN	NaN	NaN
43434	NaN	NaN	NaN	NaN
44210	0.537600	0.002900	0.000000	0.069000

	FLOORSMAX_AVG	FLOORSMIN_AVG	LANDAREA_AVG	LIVINGAPARTMENTS_AVG \
896	0.041700	NaN	0.024200	NaN
1145	NaN	NaN	NaN	NaN
2277	0.333300	NaN	0.146400	NaN
4141	NaN	NaN	NaN	NaN
6791	NaN	NaN	NaN	NaN
6985	NaN	NaN	NaN	NaN
7307	0.041700	0.083300	0.007100	0.016800
8207	0.333300	NaN	NaN	NaN
9246	0.166700	NaN	0.000000	NaN
9830	NaN	NaN	NaN	NaN
10779	0.083300	0.125000	0.027500	0.022700
10861	0.333300	0.375000	0.092300	0.272300
11591	NaN	NaN	NaN	NaN
13387	NaN	NaN	NaN	NaN
13564	0.333300	0.375000	0.114800	0.181600
13738	0.166700	NaN	NaN	NaN
14008	NaN	NaN	NaN	NaN
22146	0.041700	0.083300	0.018900	0.013400
22663	NaN	NaN	NaN	

```
NaN
26027            NaN            NaN           NaN
NaN
26046       0.333300       0.000000      0.000000
0.120200
33294            NaN            NaN           NaN
NaN
43434            NaN            NaN           NaN
NaN
44210       0.041700       0.062500      0.104300
0.013400

       LIVINGAREA_AVG  NONLIVINGAPARTMENTS_AVG  NONLIVINGAREA_AVG  \
896          0.015700                      NaN           0.003600
1145              NaN                      NaN                NaN
2277         0.160800                      NaN           0.000000
4141              NaN                      NaN                NaN
6791              NaN                      NaN                NaN
6985              NaN                      NaN                NaN
7307         0.012100                 0.000000           0.000000
8207         0.153000                      NaN           0.000000
9246         0.070600                      NaN           0.000000
9830              NaN                      NaN                NaN
10779        0.026600                 0.000000           0.000000
10861        0.354500                 0.000000           0.000000
11591             NaN                      NaN                NaN
13387             NaN                      NaN                NaN
13564        0.208200                 0.007700           0.006400
13738        0.133300                      NaN           0.000000
14008             NaN                      NaN                NaN
22146        0.014900                      NaN           0.000000
22663             NaN                      NaN                NaN
26027             NaN                      NaN                NaN
26046        0.099600                 0.000000           0.003900
33294             NaN                      NaN                NaN
43434             NaN                      NaN                NaN
44210        0.017300                 0.000000           0.000000

       APARTMENTS_MODE  BASEMENTAREA_MODE
YEARS_BEGINEXPLUATATION_MODE  \
896           0.016800                NaN
0.977200
1145               NaN                NaN
NaN
2277          0.151300           0.120100
0.984600
4141               NaN                NaN
NaN
6791               NaN                NaN
NaN
```

```
6985               NaN                NaN
NaN
7307          0.021000           0.002800
0.937400
8207          0.151300                NaN
0.982600
9246          0.098700           0.069600
0.985100
9830               NaN                NaN
NaN
10779         0.028400           0.048700
0.985600
10861         0.340300           0.219700
0.984600
11591              NaN                NaN
NaN
13387              NaN                NaN
NaN
13564         0.229000           0.011500
0.983600
13738         0.062000           0.144600
0.987100
14008              NaN                NaN
NaN
22146         0.016800           0.000000
0.978600
22663              NaN                NaN
NaN
26027              NaN                NaN
NaN
26046         0.150200           0.098800
0.979600
33294              NaN                NaN
NaN
43434              NaN                NaN
NaN
44210         0.016800           0.000000
0.966200

       YEARS_BUILD_MODE  COMMONAREA_MODE  ELEVATORS_MODE
ENTRANCES_MODE  \
896                 NaN              NaN        0.000000
0.069000
1145                NaN              NaN             NaN
NaN
2277                NaN              NaN        0.161100
0.137900
4141                NaN              NaN             NaN
NaN
6791                NaN              NaN             NaN
```

```
NaN
6985                NaN              NaN             NaN
NaN
7307           0.176800         0.000000        0.000000
0.034500
8207                NaN              NaN        0.161100
0.137900
9246                NaN              NaN        0.000000
0.069000
9830                NaN              NaN             NaN
NaN
10779          0.810500              NaN        0.000000
0.103400
10861          0.797500         0.071300        0.362500
0.310300
11591               NaN              NaN             NaN
NaN
13387               NaN              NaN             NaN
NaN
13564          0.784400         0.036800        0.241700
0.206900
13738               NaN              NaN        0.000000
0.137900
14008               NaN              NaN             NaN
NaN
22146          0.719000              NaN        0.000000
0.069000
22663               NaN              NaN             NaN
NaN
26027               NaN              NaN             NaN
NaN
26046          0.732100         0.060700        0.161100
0.137900
33294               NaN              NaN             NaN
NaN
43434               NaN              NaN             NaN
NaN
44210          0.555700         0.002900        0.000000
0.069000

       FLOORSMAX_MODE  FLOORSMIN_MODE  LANDAREA_MODE
LIVINGAPARTMENTS_MODE  \
896          0.041700             NaN       0.024800
NaN
1145              NaN             NaN            NaN
NaN
2277         0.333300             NaN       0.149800
NaN
4141              NaN             NaN            NaN
NaN
```

```
6791              NaN             NaN            NaN
NaN
6985              NaN             NaN            NaN
NaN
7307         0.041700        0.083300       0.007300
0.018400
8207         0.333300             NaN            NaN
NaN
9246         0.166700             NaN       0.000000
NaN
9830              NaN             NaN            NaN
NaN
10779        0.083300        0.125000       0.028100
0.024800
10861        0.333300        0.375000       0.094400
0.297500
11591             NaN             NaN            NaN
NaN
13387             NaN             NaN            NaN
NaN
13564        0.333300        0.375000       0.117500
0.198300
13738        0.166700             NaN            NaN
NaN
14008             NaN             NaN            NaN
NaN
22146        0.041700        0.083300       0.019300
0.014700
22663             NaN             NaN            NaN
NaN
26027             NaN             NaN            NaN
NaN
26046        0.333300        0.000000       0.000000
0.131300
33294             NaN             NaN            NaN
NaN
43434             NaN             NaN            NaN
NaN
44210        0.041700        0.041700       0.028100
0.014700

       LIVINGAREA_MODE  NONLIVINGAPARTMENTS_MODE
NONLIVINGAREA_MODE  \
896           0.016300                       NaN            0.003800

1145               NaN                       NaN                 NaN

2277          0.167500                       NaN            0.000000

4141               NaN                       NaN                 NaN
```

6791	NaN	NaN	NaN
6985	NaN	NaN	NaN
7307	0.012600	0.000000	0.000000
8207	0.159400	NaN	0.000000
9246	0.073600	NaN	0.000000
9830	NaN	NaN	NaN
10779	0.027700	0.000000	0.000000
10861	0.369300	0.000000	0.000000
11591	NaN	NaN	NaN
13387	NaN	NaN	NaN
13564	0.216900	0.007800	0.006700
13738	0.062700	NaN	0.000000
14008	NaN	NaN	NaN
22146	0.015600	NaN	0.000000
22663	NaN	NaN	NaN
26027	NaN	NaN	NaN
26046	0.103800	0.000000	0.004100
33294	NaN	NaN	NaN
43434	NaN	NaN	NaN
44210	0.014300	0.000000	0.000000

```
       APARTMENTS_MEDI  BASEMENTAREA_MEDI
YEARS_BEGINEXPLUATATION_MEDI  \
896           0.016700                NaN
0.977100
1145               NaN                NaN
NaN
```

```
2277          0.149900           0.115700
0.984600
4141               NaN                NaN
NaN
6791               NaN                NaN
NaN
6985               NaN                NaN
NaN
7307          0.020800           0.002700
0.937400
8207          0.149900                NaN
0.982600
9246          0.097800           0.067100
0.985100
9830               NaN                NaN
NaN
10779         0.028100           0.046900
0.985600
10861         0.337300           0.211700
0.984600
11591              NaN                NaN
NaN
13387              NaN                NaN
NaN
13564         0.226900           0.011100
0.983600
13738         0.101000           0.139300
0.989100
14008              NaN                NaN
NaN
22146         0.016700           0.000000
0.978600
22663              NaN                NaN
NaN
26027              NaN                NaN
NaN
26046         0.148900           0.095200
0.979600
33294              NaN                NaN
NaN
43434              NaN                NaN
NaN
44210         0.016700           0.000000
0.966200

       YEARS_BUILD_MEDI  COMMONAREA_MEDI  ELEVATORS_MEDI
ENTRANCES_MEDI  \
896                 NaN              NaN        0.000000
0.069000
1145                NaN              NaN             NaN
```

```
NaN
2277                NaN              NaN        0.160000
0.137900
4141                NaN              NaN             NaN
NaN
6791                NaN              NaN             NaN
NaN
6985                NaN              NaN             NaN
NaN
7307           0.154700         0.000000        0.000000
0.034500
8207                NaN              NaN        0.160000
0.137900
9246                NaN              NaN        0.000000
0.069000
9830                NaN              NaN             NaN
NaN
10779          0.805400              NaN        0.000000
0.103400
10861          0.792000         0.071100        0.360000
0.310300
11591               NaN              NaN             NaN
NaN
13387               NaN              NaN             NaN
NaN
13564          0.778600         0.036700        0.240000
0.206900
13738               NaN              NaN        0.000000
0.137900
14008               NaN              NaN             NaN
NaN
22146          0.711500              NaN        0.000000
0.069000
22663               NaN              NaN             NaN
NaN
26027               NaN              NaN             NaN
NaN
26046          0.724900         0.060500        0.160000
0.137900
33294               NaN              NaN             NaN
NaN
43434               NaN              NaN             NaN
NaN
44210          0.543800         0.002900        0.000000
0.069000

       FLOORSMAX_MEDI  FLOORSMIN_MEDI  LANDAREA_MEDI
LIVINGAPARTMENTS_MEDI  \
896          0.041700             NaN       0.024600
NaN
```

```
1145              NaN             NaN            NaN
NaN
2277         0.333300             NaN       0.149000
NaN
4141              NaN             NaN            NaN
NaN
6791              NaN             NaN            NaN
NaN
6985              NaN             NaN            NaN
NaN
7307         0.041700        0.083300       0.007200
0.017100
8207         0.333300             NaN            NaN
NaN
9246         0.166700             NaN       0.000000
NaN
9830              NaN             NaN            NaN
NaN
10779        0.083300        0.125000       0.028000
0.023100
10861        0.333300        0.375000       0.093900
0.277000
11591             NaN             NaN            NaN
NaN
13387             NaN             NaN            NaN
NaN
13564        0.333300        0.375000       0.116800
0.184700
13738        0.166700             NaN            NaN
NaN
14008             NaN             NaN            NaN
NaN
22146        0.041700        0.083300       0.019200
0.013700
22663             NaN             NaN            NaN
NaN
26027             NaN             NaN            NaN
NaN
26046        0.333300        0.000000       0.000000
0.122300
33294             NaN             NaN            NaN
NaN
43434             NaN             NaN            NaN
NaN
44210        0.041700        0.062500       0.140600
0.013700

       LIVINGAREA_MEDI  NONLIVINGAPARTMENTS_MEDI
NONLIVINGAREA_MEDI  \
896           0.015900                       NaN            0.003700
```

1145	NaN	NaN	NaN
2277	0.163700	NaN	0.000000
4141	NaN	NaN	NaN
6791	NaN	NaN	NaN
6985	NaN	NaN	NaN
7307	0.012300	0.000000	0.000000
8207	0.155800	NaN	0.000000
9246	0.071900	NaN	0.000000
9830	NaN	NaN	NaN
10779	0.027100	0.000000	0.000000
10861	0.360800	0.000000	0.000000
11591	NaN	NaN	NaN
13387	NaN	NaN	NaN
13564	0.212000	0.007800	0.006500
13738	0.095500	NaN	0.000000
14008	NaN	NaN	NaN
22146	0.015200	NaN	0.000000
22663	NaN	NaN	NaN
26027	NaN	NaN	NaN
26046	0.101400	0.000000	0.004000
33294	NaN	NaN	NaN
43434	NaN	NaN	NaN
44210	0.019400	0.000000	0.000000

```
          FONDKAPREMONT_MODE  HOUSETYPE_MODE  TOTALAREA_MODE  \
896                      NaN  block of flats        0.013100
1145                     NaN             NaN             NaN
2277                     NaN  block of flats        0.155200
4141                     NaN             NaN             NaN
6791                     NaN             NaN             NaN
6985                     NaN             NaN             NaN
7307        reg oper account  block of flats        0.009500
8207                     NaN  block of flats        0.120400
9246                     NaN  block of flats        0.055600
9830                     NaN             NaN             NaN
10779       reg oper account  block of flats        0.020900
10861  reg oper spec account  block of flats        0.317400
11591                    NaN             NaN             NaN
13387                    NaN             NaN             NaN
13564       reg oper account  block of flats        0.238900
13738                    NaN  block of flats        0.193300
14008                    NaN             NaN             NaN
22146       reg oper account  block of flats        0.011700
22663                    NaN             NaN             NaN
26027                    NaN             NaN             NaN
26046       reg oper account  block of flats        0.112100
33294                    NaN             NaN             NaN
43434                    NaN             NaN             NaN
44210       reg oper account  block of flats        0.015000

      WALLSMATERIAL_MODE EMERGENCYSTATE_MODE  OBS_30_CNT_SOCIAL_CIRCLE
\
896         Stone, brick                  No                  0.000000

1145                NaN                 NaN                  4.000000

2277        Stone, brick                  No                  0.000000

4141                NaN                 NaN                  1.000000

6791                NaN                 NaN                  0.000000

6985                NaN                 NaN                  0.000000

7307        Stone, brick                  No                  0.000000

8207               Block                  No                  0.000000

9246               Panel                  No                  2.000000

9830                NaN                 NaN                  0.000000

10779              Panel                  No                  0.000000
```

10861	Panel	No	2.000000
11591	NaN	NaN	10.000000
13387	NaN	NaN	1.000000
13564	Panel	No	0.000000
13738	Stone, brick	No	0.000000
14008	NaN	NaN	0.000000
22146	Stone, brick	No	0.000000
22663	NaN	NaN	0.000000
26027	NaN	NaN	2.000000
26046	Panel	No	0.000000
33294	NaN	NaN	0.000000
43434	NaN	NaN	1.000000
44210	Block	No	3.000000

	DEF_30_CNT_SOCIAL_CIRCLE	OBS_60_CNT_SOCIAL_CIRCLE	\
896	0.000000	0.000000	
1145	2.000000	4.000000	
2277	0.000000	0.000000	
4141	0.000000	1.000000	
6791	0.000000	0.000000	
6985	0.000000	0.000000	
7307	0.000000	0.000000	
8207	0.000000	0.000000	
9246	0.000000	2.000000	
9830	0.000000	0.000000	
10779	0.000000	0.000000	
10861	0.000000	2.000000	
11591	0.000000	10.000000	
13387	0.000000	1.000000	
13564	0.000000	0.000000	
13738	0.000000	0.000000	
14008	0.000000	0.000000	
22146	0.000000	0.000000	
22663	0.000000	0.000000	
26027	2.000000	2.000000	

26046	0.000000	0.000000
33294	0.000000	0.000000
43434	0.000000	1.000000
44210	0.000000	3.000000

	DEF_60_CNT_SOCIAL_CIRCLE	DAYS_LAST_PHONE_CHANGE	FLAG_DOCUMENT_2 \
896	0.000000	-615.000000	0
1145	1.000000	-879.000000	0
2277	0.000000	-1038.000000	0
4141	0.000000	-441.000000	0
6791	0.000000	-1153.000000	0
6985	0.000000	-875.000000	0
7307	0.000000	-673.000000	0
8207	0.000000	-40.000000	0
9246	0.000000	-519.000000	0
9830	0.000000	-2699.000000	0
10779	0.000000	-1938.000000	0
10861	0.000000	-628.000000	0
11591	0.000000	-1552.000000	0
13387	0.000000	-170.000000	0
13564	0.000000	-2167.000000	0
13738	0.000000	-582.000000	0
14008	0.000000	-286.000000	0
22146	0.000000	-3861.000000	0
22663	0.000000	-631.000000	0
26027	2.000000	-349.000000	0
26046	0.000000	-354.000000	0
33294	0.000000	-217.000000	

0
43434 0.000000 -905.000000
0
44210 0.000000 -411.000000
0

	FLAG_DOCUMENT_3	FLAG_DOCUMENT_4	FLAG_DOCUMENT_5	FLAG_DOCUMENT_6 \
896	0	0	1	0
1145	1	0	0	0
2277	0	0	0	0
4141	1	0	0	0
6791	1	0	0	0
6985	1	0	0	0
7307	0	0	0	0
8207	0	0	0	0
9246	1	0	0	0
9830	1	0	0	0
10779	1	0	0	0
10861	1	0	0	0
11591	1	0	0	0
13387	1	0	0	0
13564	1	0	0	0
13738	1	0	0	0
14008	1	0	0	0
22146	1	0	0	0
22663	1	0	0	0
26027	1	0	0	0
26046	1	0	0	0

33294 1 0 0
0
43434 1 0 0
0
44210 1 0 0
0

FLAG_DOCUMENT_7 FLAG_DOCUMENT_8 FLAG_DOCUMENT_9
FLAG_DOCUMENT_10 \
896 0 0 0
0
1145 0 0 0
0
2277 0 1 0
0
4141 0 0 0
0
6791 0 0 0
0
6985 0 0 0
0
7307 0 1 0
0
8207 0 1 0
0
9246 0 0 0
0
9830 0 0 0
0
10779 0 0 0
0
10861 0 0 0
0
11591 0 0 0
0
13387 0 0 0
0
13564 0 0 0
0
13738 0 0 0
0
14008 0 0 0
0
22146 0 0 0
0
22663 0 0 0
0
26027 0 0 0
0
26046 0 0 0

0
33294 0 0 0
0
43434 0 0 0
0
44210 0 0 0
0

	FLAG_DOCUMENT_11	FLAG_DOCUMENT_12	FLAG_DOCUMENT_13	FLAG_DOCUMENT_14 \
896	0	0	0	0
1145	0	0	0	0
2277	0	0	0	0
4141	0	0	0	0
6791	0	0	0	0
6985	0	0	0	0
7307	0	0	0	0
8207	0	0	0	0
9246	0	0	0	0
9830	0	0	0	0
10779	0	0	0	0
10861	0	0	0	0
11591	0	0	0	0
13387	0	0	0	0
13564	0	0	0	0
13738	0	0	0	0
14008	0	0	0	0
22146	0	0	0	0
22663	0	0	0	0
26027	0	0	0	0

26046 0 0 0
0
33294 0 0 0
0
43434 0 0 0
0
44210 0 0 0
0

FLAG_DOCUMENT_15 FLAG_DOCUMENT_16 FLAG_DOCUMENT_17
FLAG_DOCUMENT_18 \
896 0 0 0
0
1145 0 0 0
0
2277 0 0 0
0
4141 0 0 0
0
6791 0 0 0
0
6985 0 0 0
0
7307 0 0 0
0
8207 0 0 0
0
9246 0 0 0
0
9830 0 0 0
0
10779 0 0 0
0
10861 0 0 0
0
11591 0 0 0
0
13387 0 0 0
0
13564 0 0 0
0
13738 0 0 0
0
14008 0 0 0
0
22146 0 0 0
0
22663 0 0 0
0
26027 0 0 0

0
26046 0 0 0
0
33294 0 0 0
0
43434 0 0 0
0
44210 0 0 0
0

	FLAG_DOCUMENT_19	FLAG_DOCUMENT_20	FLAG_DOCUMENT_21	\
896	0	0	0	
1145	0	0	0	
2277	0	0	0	
4141	0	0	0	
6791	0	0	0	
6985	0	0	0	
7307	0	0	0	
8207	0	0	0	
9246	0	0	0	
9830	0	0	0	
10779	0	0	0	
10861	0	0	0	
11591	0	0	0	
13387	0	0	0	
13564	0	0	0	
13738	0	0	0	
14008	0	0	0	
22146	0	0	0	
22663	0	0	0	
26027	0	0	0	
26046	0	0	0	
33294	0	0	0	
43434	0	0	0	
44210	0	0	0	

	AMT_REQ_CREDIT_BUREAU_HOUR	AMT_REQ_CREDIT_BUREAU_DAY	\
896	0.000000	0.000000	
1145	0.000000	0.000000	
2277	0.000000	0.000000	
4141	0.000000	0.000000	
6791	0.000000	0.000000	
6985	0.000000	0.000000	
7307	0.000000	0.000000	
8207	0.000000	0.000000	
9246	0.000000	0.000000	
9830	0.000000	0.000000	
10779	0.000000	0.000000	
10861	0.000000	0.000000	
11591	0.000000	0.000000	

```
13387                    0.000000                    0.000000
13564                    0.000000                    0.000000
13738                    0.000000                    0.000000
14008                    0.000000                    0.000000
22146                         NaN                         NaN
22663                    0.000000                    0.000000
26027                    0.000000                    0.000000
26046                    0.000000                    0.000000
33294                    0.000000                    0.000000
43434                    0.000000                    0.000000
44210                    0.000000                    0.000000

       AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
896                      0.000000                    0.000000
1145                     0.000000                    0.000000
2277                     0.000000                    0.000000
4141                     0.000000                    0.000000
6791                     0.000000                    0.000000
6985                     0.000000                    0.000000
7307                     0.000000                    0.000000
8207                     0.000000                    0.000000
9246                     0.000000                    0.000000
9830                     0.000000                    0.000000
10779                    0.000000                    0.000000
10861                    0.000000                    0.000000
11591                    0.000000                    0.000000
13387                    0.000000                    0.000000
13564                    0.000000                    0.000000
13738                    0.000000                    0.000000
14008                    0.000000                    0.000000
22146                         NaN                         NaN
22663                    0.000000                    0.000000
26027                    0.000000                    0.000000
26046                    0.000000                    0.000000
33294                    0.000000                    0.000000
43434                    0.000000                    0.000000
44210                    0.000000                    0.000000

       AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
Client_Age
896                     0.000000                    2.000000
31
1145                    1.000000                    0.000000
27
2277                    2.000000                    1.000000
24
4141                    1.000000                    2.000000
48
6791                    1.000000                    4.000000
55
```

```
6985                    1.000000                    0.000000
49
7307                    1.000000                    3.000000
44
8207                    0.000000                    1.000000
36
9246                    0.000000                    3.000000
36
9830                    1.000000                    2.000000
35
10779                   1.000000                    3.000000
51
10861                   0.000000                    2.000000
27
11591                   0.000000                    6.000000
61
13387                   0.000000                    5.000000
65
13564                   0.000000                    1.000000
39
13738                   1.000000                    6.000000
47
14008                   0.000000                    2.000000
50
22146                        NaN                         NaN
36
22663                   3.000000                    3.000000
23
26027                   0.000000                    1.000000
32
26046                   1.000000                    1.000000
54
33294                   0.000000                    5.000000
27
43434                   1.000000                    2.000000
38
44210                   0.000000                    2.000000
37
```

```
df_train['AMT_ANNUITY'].describe()
```

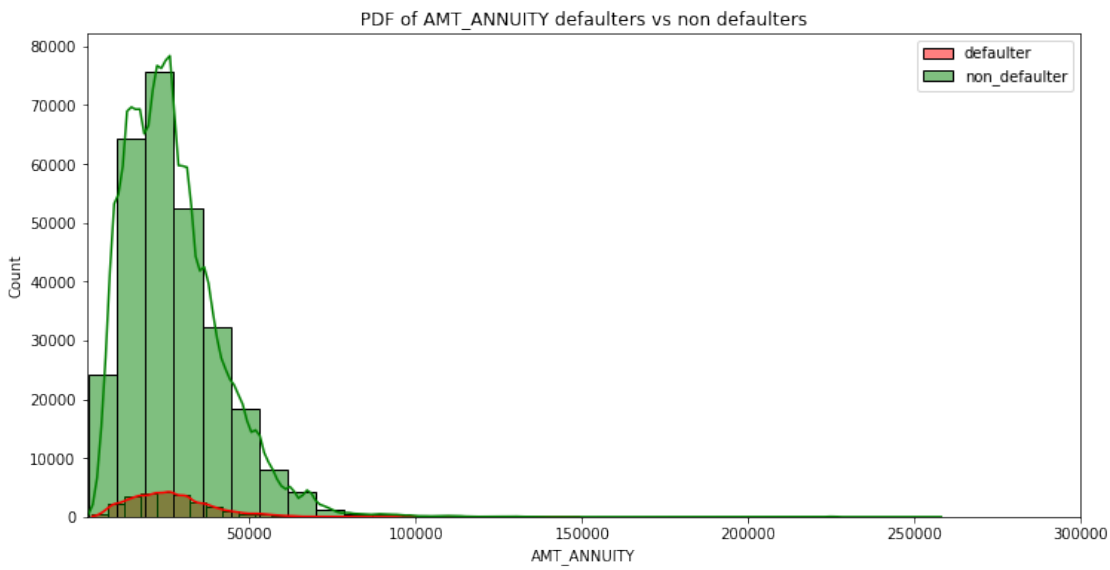
```
count   307499.000000
mean     27108.573909
std      14493.737315
min       1615.500000
25%      16524.000000
50%      24903.000000
75%      34596.000000
max     258025.500000
Name: AMT_ANNUITY, dtype: float64
```

```
df_train[['TARGET','AMT_ANNUITY']].corr()['TARGET']
```

```
TARGET         1.000000
AMT_ANNUITY   -0.012817
Name: TARGET, dtype: float64
```

```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['AMT_ANNUITY'],bins=30,kde=True,label='defau
lter',color='red')
sns.histplot(df_non_defaulter['AMT_ANNUITY'],bins=30,kde=True,label='n
on_defaulter',color="green")
plt.xlim([1000,300000])
#plt.xticks(np.arange(1000,50000,10))
plt.title("PDF of AMT_ANNUITY defaulters vs non defaulters")
plt.legend()
```

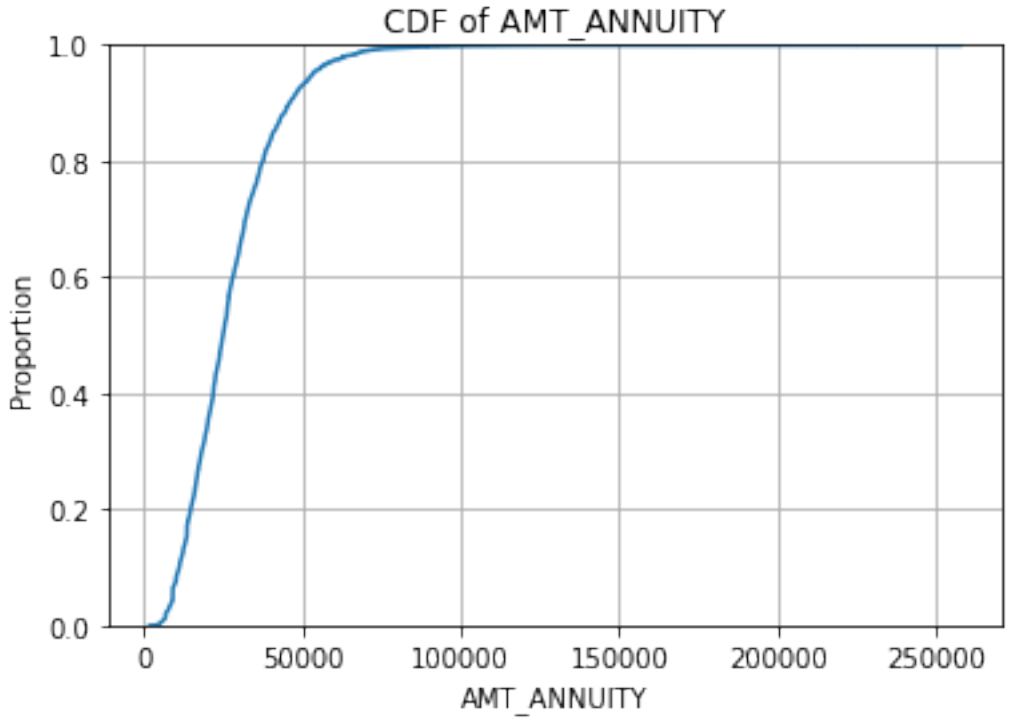
```
<matplotlib.legend.Legend at 0x1e2a1961a60>
```



- It seems that most defaulters lies below AMT_ANNUITY 50000

```
sns.ecdfplot(df_train['AMT_ANNUITY'])
#plt.xlim([40000,5000000])
plt.grid()
#plt.xticks(df_train['AMT_ANNUITY'].value_counts().index.to_list())
plt.title("CDF of AMT_ANNUITY")
```

```
Text(0.5, 1.0, 'CDF of AMT_ANNUITY')
```




```
np.quantile(df_train['AMT_ANNUITY'].dropna(),0.99)
```

```
70006.5
```

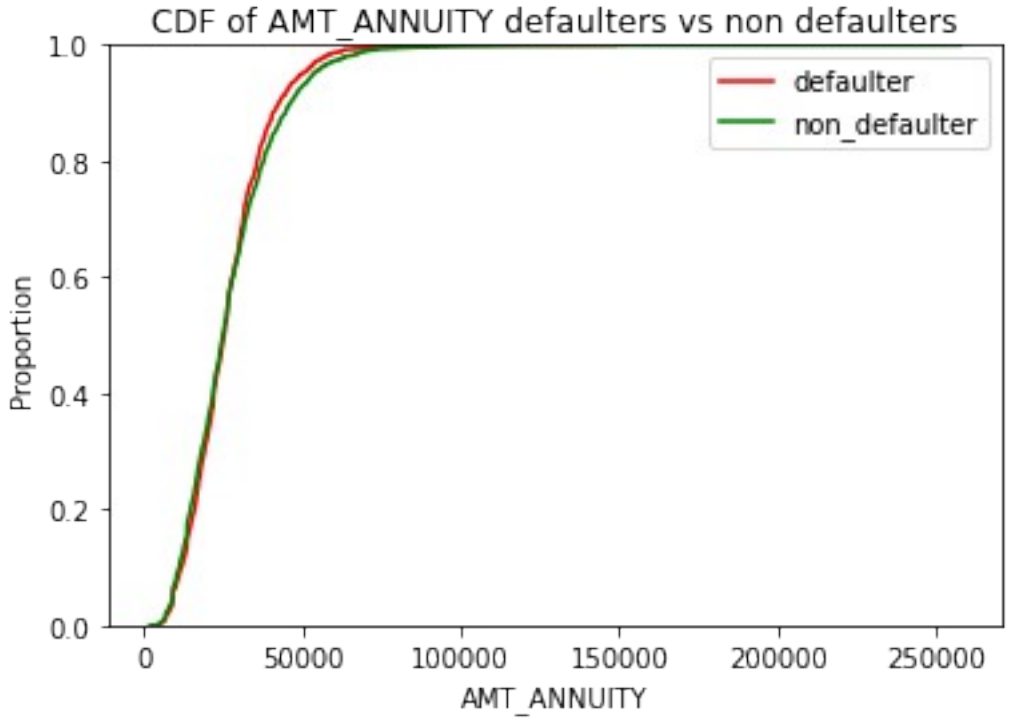
```
np.quantile(df_train['AMT_ANNUITY'].dropna(),0.80)
```

```
37516.5
```

- 99% client seems to have AMT_ANNUITY below 70000
- 80% client seems to have AMT_ANNUITY below 387500

```
sns.ecdfplot(df_defaulter['AMT_ANNUITY'],label='defaulter',color='red'
)
sns.ecdfplot(df_non_defaulter['AMT_ANNUITY'],label='non_defaulter',col
or='green')
#plt.xlim([40000,1000000])
plt.title("CDF of AMT_ANNUITY defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

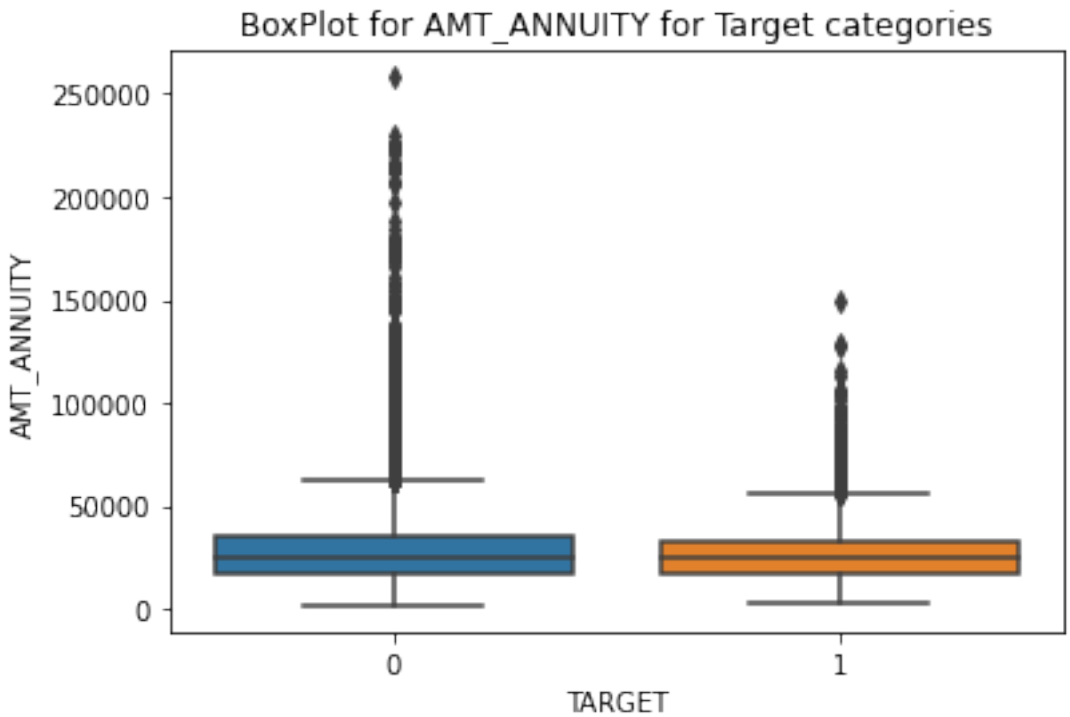
```
<matplotlib.legend.Legend at 0x1e2a1adbf70>
```



- Both have almost same distribution but deafaulters AMT_ANNUITY is lower

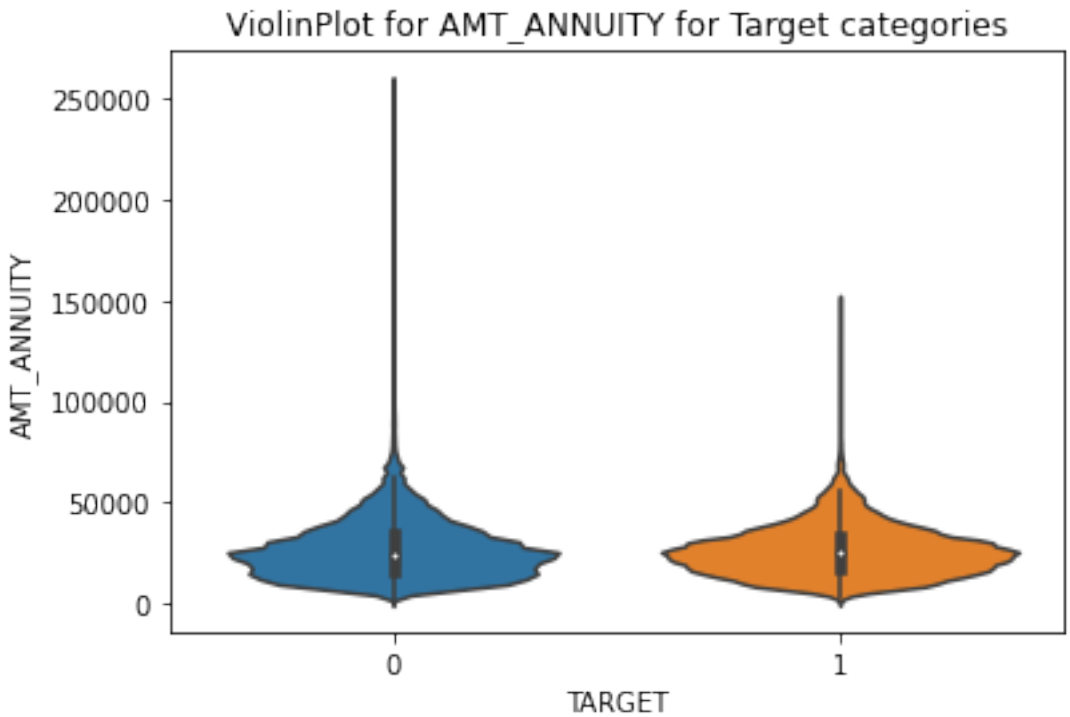
```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="AMT_ANNUITY", data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for AMT_ANNUITY for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for AMT_ANNUITY for Target categories')
```




```
#Violin Plot
ax = sns.violinplot(x="TARGET", y="AMT_ANNUITY", data=df_train)
#.ylim([40000,5000000])
plt.title("ViolinPlot for AMT_ANNUITY for Target categories")
#plt.legend()
```

```
Text(0.5, 1.0, 'ViolinPlot for AMT_ANNUITY for Target categories')
```




```
#top 10 correlated column with DAYS_REGISTRATION
df_train.corr()
[['AMT_ANNUITY']].sort_values('AMT_ANNUITY',ascending=False)[:10]
```

```
                   AMT_ANNUITY
AMT_ANNUITY           1.000000
AMT_GOODS_PRICE       0.775109
AMT_CREDIT            0.770138
AMT_INCOME_TOTAL      0.191657
FLAG_DOCUMENT_8       0.130584
FLOORSMAX_AVG         0.130174
FLOORSMAX_MEDI        0.129179
FLOORSMAX_MODE        0.126302
EXT_SOURCE_2          0.125804
EXT_SOURCE_1          0.119398
```

```
#Lets come back to NA values

# we see which are most 5 correlated columns for this column

df_train.corr()['AMT_ANNUITY'].sort_values(ascending=False)[:5]
```

```
AMT_ANNUITY        1.000000
AMT_GOODS_PRICE    0.775109
AMT_CREDIT         0.770138
AMT_INCOME_TOTAL   0.191657
FLAG_DOCUMENT_8    0.130584
Name: AMT_ANNUITY, dtype: float64
```

- 'AMT_ANNUITY' has very high correlation with 'AMT_GOODS_PRICE','AMT_CREDIT' -> This is obivous as Loan asked for substance price(AMT_GOODS_PRICE) increase Loan Amount (AMT_CREDIT) Hence annual Annnuity also Increase

```
app_data_col_Details[app_data_col_Details["Row"].isin(['AMT_ANNUITY','
AMT_GOODS_PRICE','AMT_CREDIT'])]
```

```
                Row  \
11       AMT_CREDIT
12      AMT_ANNUITY
13  AMT_GOODS_PRICE


Description
11                                                  Credit amount of
the loan
12
Loan annuity
13  For consumer loans it is the price of the goods for which the loan
is given
```

REGION_RATING_CLIENT

Our rating of the region where client lives (1,2,3)

```
df_train['REGION_RATING_CLIENT'].isna().sum()
```

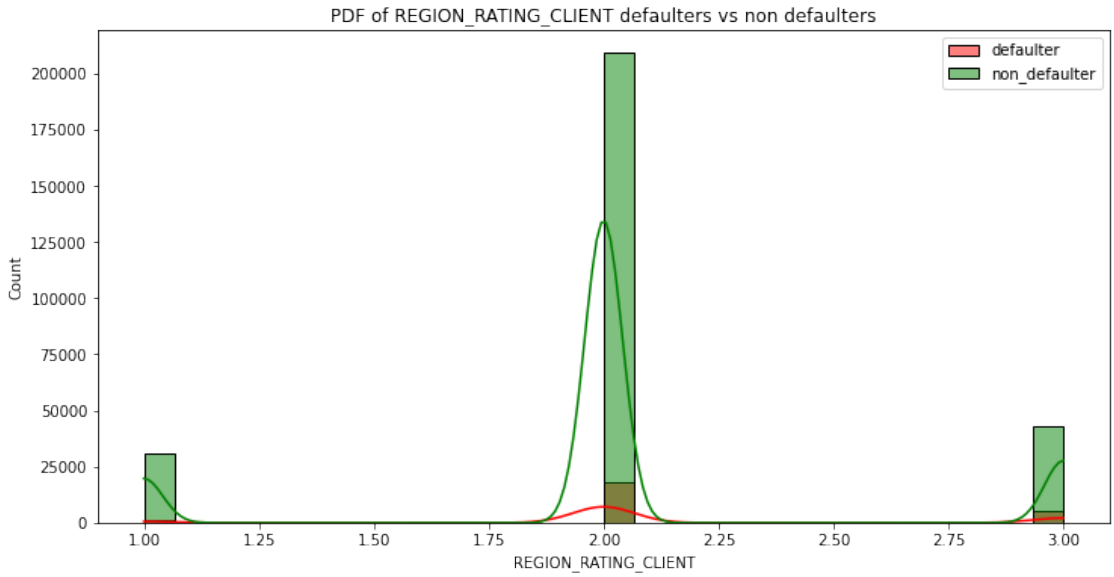
```
0
```

```
df_train['REGION_RATING_CLIENT'].value_counts()
```

```
2    226984
3     48330
1     32197
Name: REGION_RATING_CLIENT, dtype: int64
```

```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['REGION_RATING_CLIENT'],bins=30,kde=True,lab
el='defaulter',color='red')
sns.histplot(df_non_defaulter['REGION_RATING_CLIENT'],bins=30,kde=True
,label='non_defaulter',color="green")
#plt.xlim([40000,5000000])
plt.title("PDF of REGION_RATING_CLIENT defaulters vs non defaulters")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e29cd17b50>
```

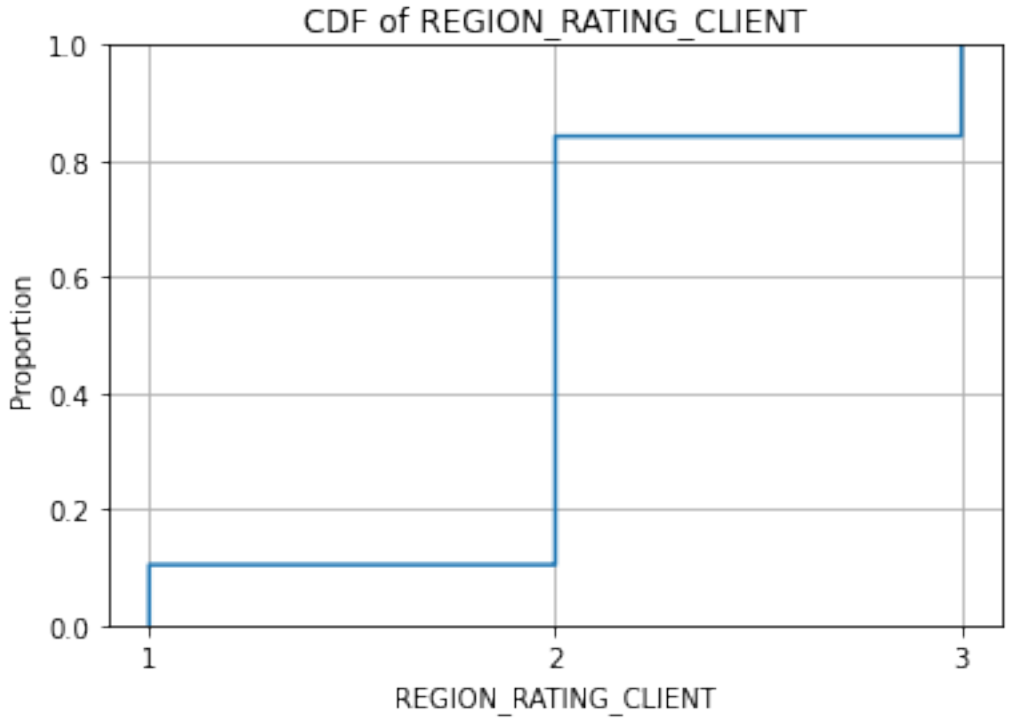


-> Most client lives in region rate 2

- For Region rating 3 ,more defaulters compare to 1

```
sns.ecdfplot(df_train['REGION_RATING_CLIENT'])
#plt.xlim([40000,5000000])
plt.grid()
plt.xticks(df_train['REGION_RATING_CLIENT'].value_counts().index.to_li
st())
plt.title("CDF of REGION_RATING_CLIENT")
```

```
Text(0.5, 1.0, 'CDF of REGION_RATING_CLIENT')
```



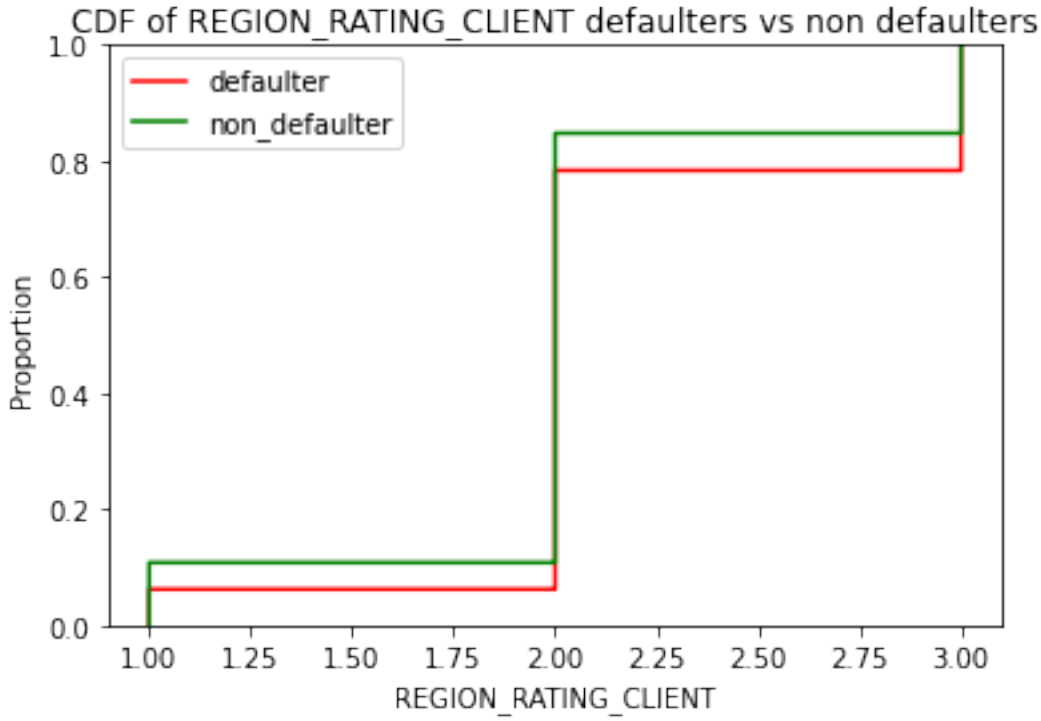

```
np.quantile(df_train['REGION_RATING_CLIENT'].dropna(),0.84)
```

```
2.0
```

- 16% client seems to be living inregion of rating above 2
- 84% client seems to be living inregion of rating below or equal 2

```
sns.ecdfplot(1*df_defaulter['REGION_RATING_CLIENT'],label='defaulter',
color='red')
sns.ecdfplot(df_non_defaulter['REGION_RATING_CLIENT'],label='non_defau
lter',color='green')
#plt.xlim([40000,1000000])
plt.title("CDF of REGION_RATING_CLIENT defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1e27635ae80>
```



- Most defaulter clients living in AREA 1 and 3 are lesser than 2

```
#Violin Plot
ax = sns.violinplot(x="TARGET", y="REGION_RATING_CLIENT",
data=df_train)
#.ylim([40000,5000000])
plt.title("ViolinPlot for REGION_RATING_CLIENT for Target categories")
#plt.legend()
```

```
Text(0.5, 1.0, 'ViolinPlot for REGION_RATING_CLIENT for Target
categories')
```

```
#top 10 correlated columns
df_train.corr()
[['REGION_RATING_CLIENT']].sort_values('REGION_RATING_CLIENT',ascendin
g=False)[:10]
```

```
                              REGION_RATING_CLIENT
REGION_RATING_CLIENT                      1.000000
REGION_RATING_CLIENT_W_CITY               0.950842
OWN_CAR_AGE                               0.087494
DAYS_REGISTRATION                         0.080210
FLAG_DOCUMENT_3                           0.062528
TARGET                                    0.058899
LANDAREA_MODE                             0.057344
YEARS_BUILD_MODE                          0.045814
LANDAREA_MEDI                             0.045602
LANDAREA_AVG                              0.043811
```

- Higly correlated with REGION_RATING_CLIENT_W_CITY, we can drop one of Two's

REGION_RATING_CLIENT_W_CITY

Our rating of the region where client lives with taking city into account (1,2,3)

```
df_train['REGION_RATING_CLIENT_W_CITY'].isna().sum()
```

```
0
```

- This is highly correlated with REGION_RATING_CLIENT

REG_REGION_NOT_LIVE_REGION

Flag if client's permanent address does not match contact address (1=different, 0=same, at region level)

```
df_train['REG_REGION_NOT_LIVE_REGION'].value_counts()
```

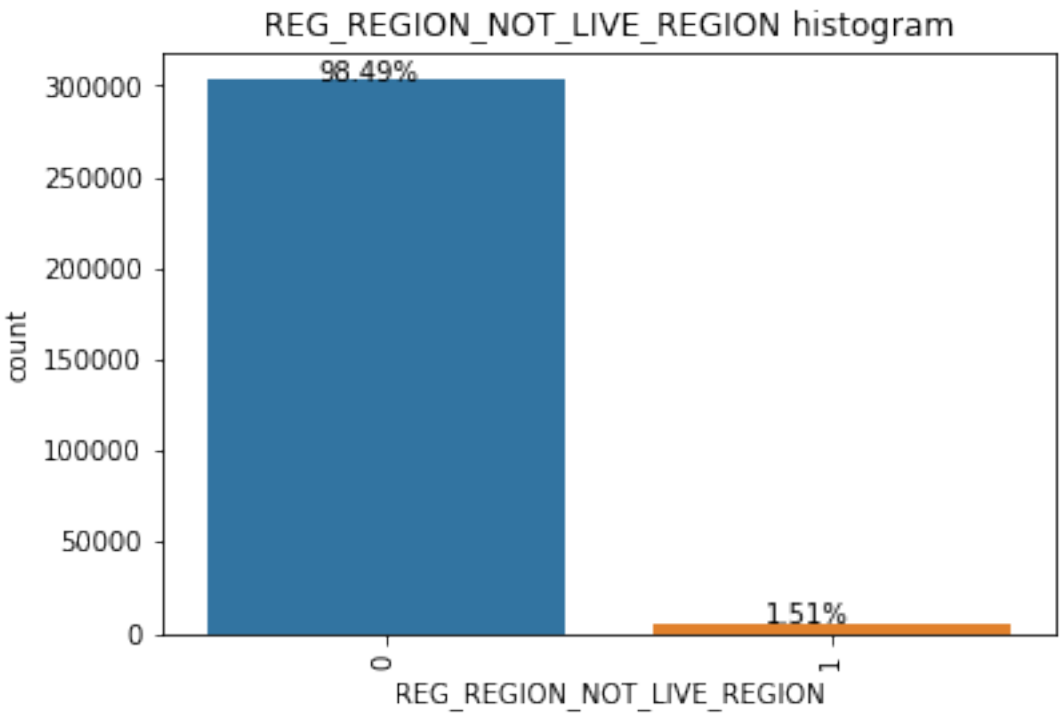
```
0    302854
1      4657
Name: REG_REGION_NOT_LIVE_REGION, dtype: int64
```

```
df_train['REG_REGION_NOT_LIVE_REGION'].isna().sum()
#No NA values
```

```
0
```

```
#lets see its distribution

ax=sns.countplot(x='REG_REGION_NOT_LIVE_REGION',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("REG_REGION_NOT_LIVE_REGION histogram")
```

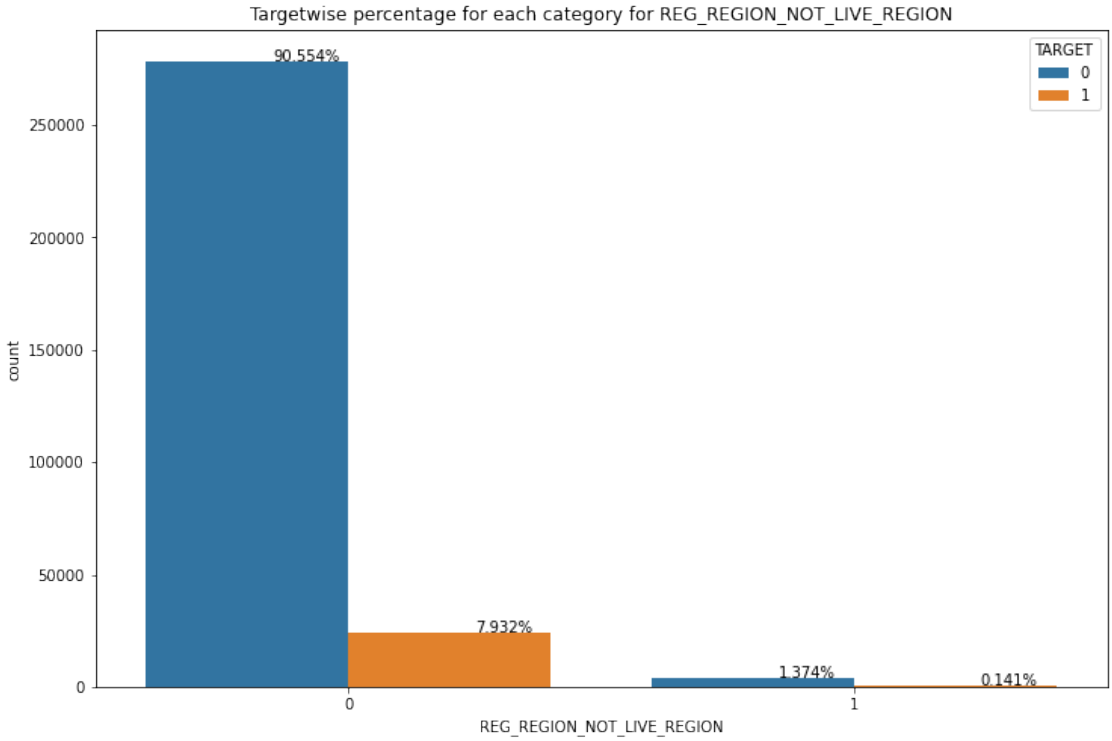
```
Text(0.5, 1.0, 'REG_REGION_NOT_LIVE_REGION histogram')
```



Most loan seekers are actually living at their permanet address

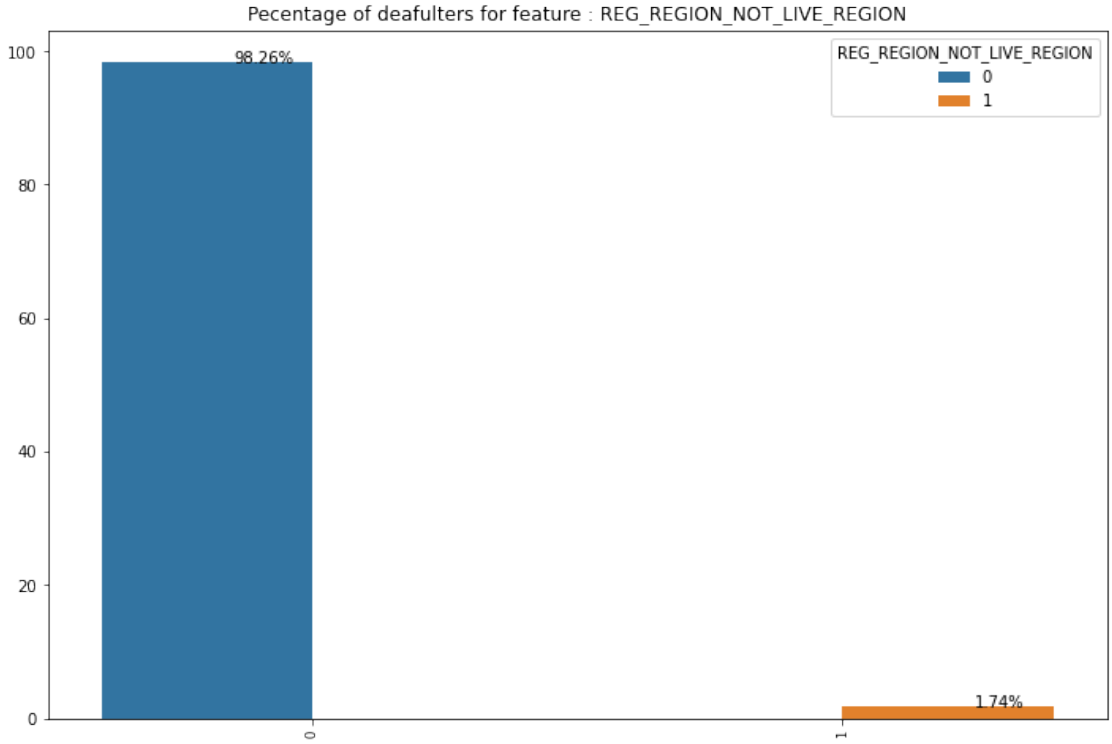
```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='REG_REGION_NOT_LIVE_REGION',hue='TARGET',data=df_t
rain)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
REG_REGION_NOT_LIVE_REGION")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
REG_REGION_NOT_LIVE_REGION')
```



- Most client are living are living on their permannet adresses

```
plot_for_categorical_defaulters(df_train,'REG_REGION_NOT_LIVE_REGION')
#plt.xlabel(('Not defaulters','deafulters'))
```



it seems people most defaulters are living in their permanet adress

- we can drop this column because of very low variance

```
#top 10 correlated columns
df_train.corr()
[['REG_REGION_NOT_LIVE_REGION']].sort_values('REG_REGION_NOT_LIVE_REGI
ON',ascending=False)[:10]
```

```
                            REG_REGION_NOT_LIVE_REGION
REG_REGION_NOT_LIVE_REGION                    1.000000
REG_REGION_NOT_WORK_REGION                    0.450804
REG_CITY_NOT_LIVE_CITY                        0.339232
REG_CITY_NOT_WORK_CITY                        0.143075
FLAG_DOCUMENT_11                              0.103987
LIVE_REGION_NOT_WORK_REGION                   0.088263
DAYS_BIRTH                                    0.065486
FLAG_WORK_PHONE                               0.064484
HOUR_APPR_PROCESS_START                       0.054894
AMT_ANNUITY                                   0.041299
```

REG_REGION_NOT_WORK_REGION

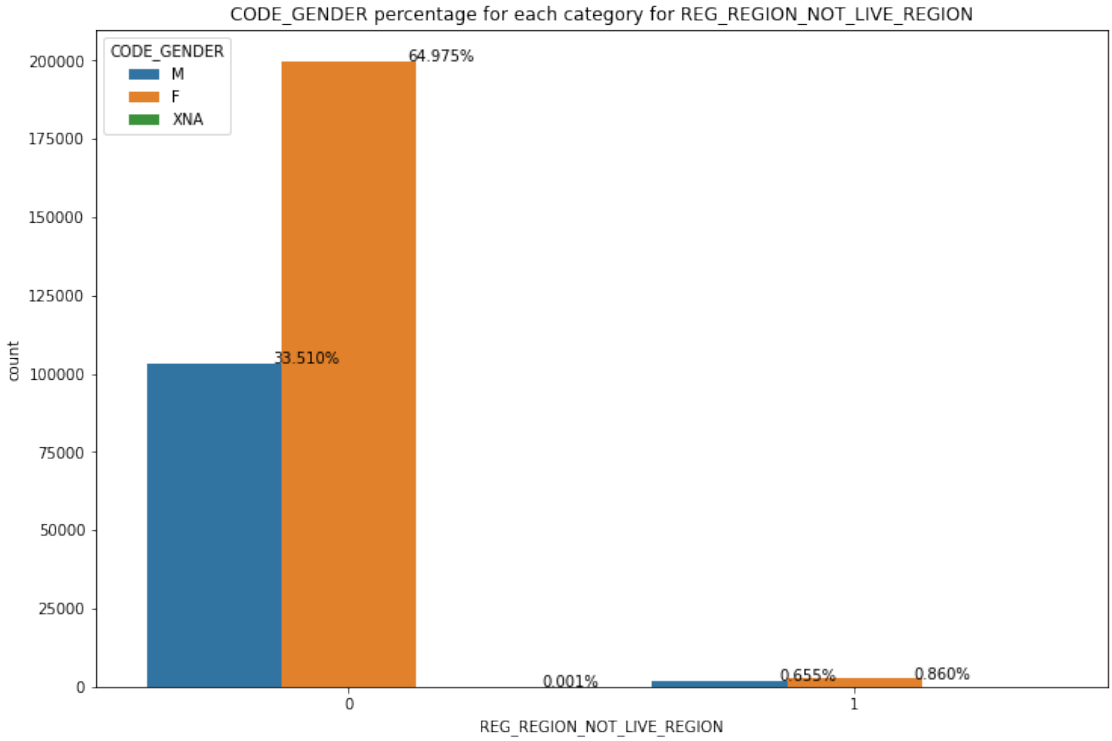
FLAG if client's permanent address does not match work address (1=different, 0=same, at region level)

```
df_train['REG_REGION_NOT_WORK_REGION'].value_counts()
```

```
0    291899
1     15612
Name: REG_REGION_NOT_WORK_REGION, dtype: int64
```

```
#lets see its distribution with resepect to gender variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='REG_REGION_NOT_LIVE_REGION',hue='CODE_GENDER',data
=df_train)
#ax.set_xticklabels(df_train['REG_REGION_NOT_LIVE_REGION'].value_count
s().index.to_list(),rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("CODE_GENDER percentage for each category for
REG_REGION_NOT_LIVE_REGION")
```

```
Text(0.5, 1.0, 'CODE_GENDER percentage for each category for
REG_REGION_NOT_LIVE_REGION')
```



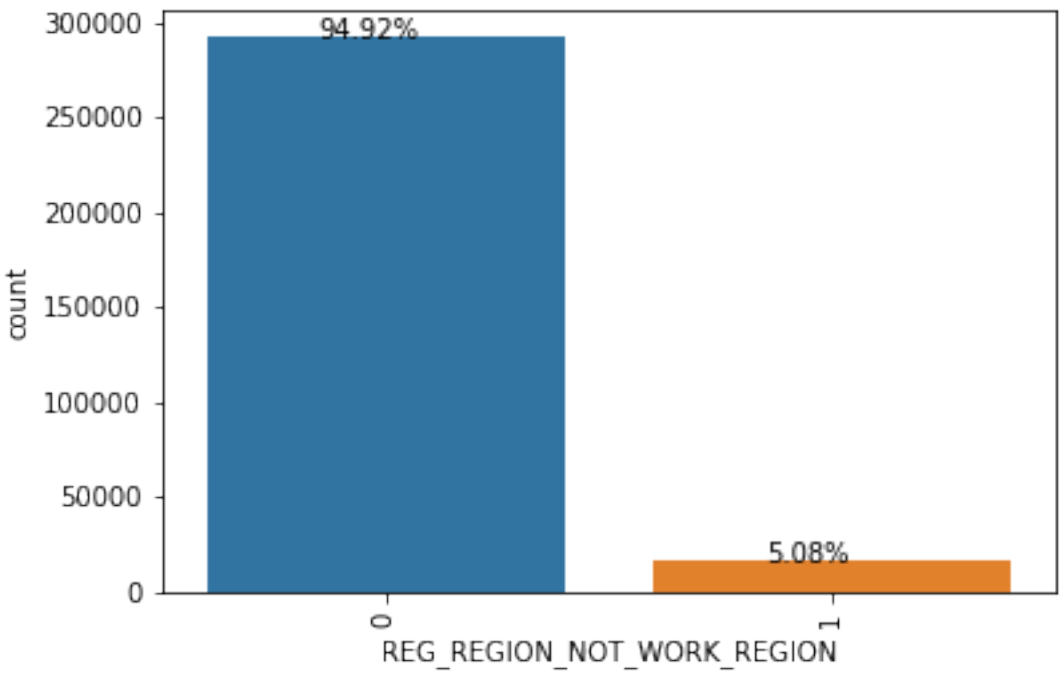

```
df_train['REG_REGION_NOT_WORK_REGION'].describe()
```

```
count   307511.000000
mean         0.050769
std          0.219526
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          1.000000
Name: REG_REGION_NOT_WORK_REGION, dtype: float64
```

```
df_train['REG_REGION_NOT_WORK_REGION'].isna().sum()
#No NA values
```

```
0
```

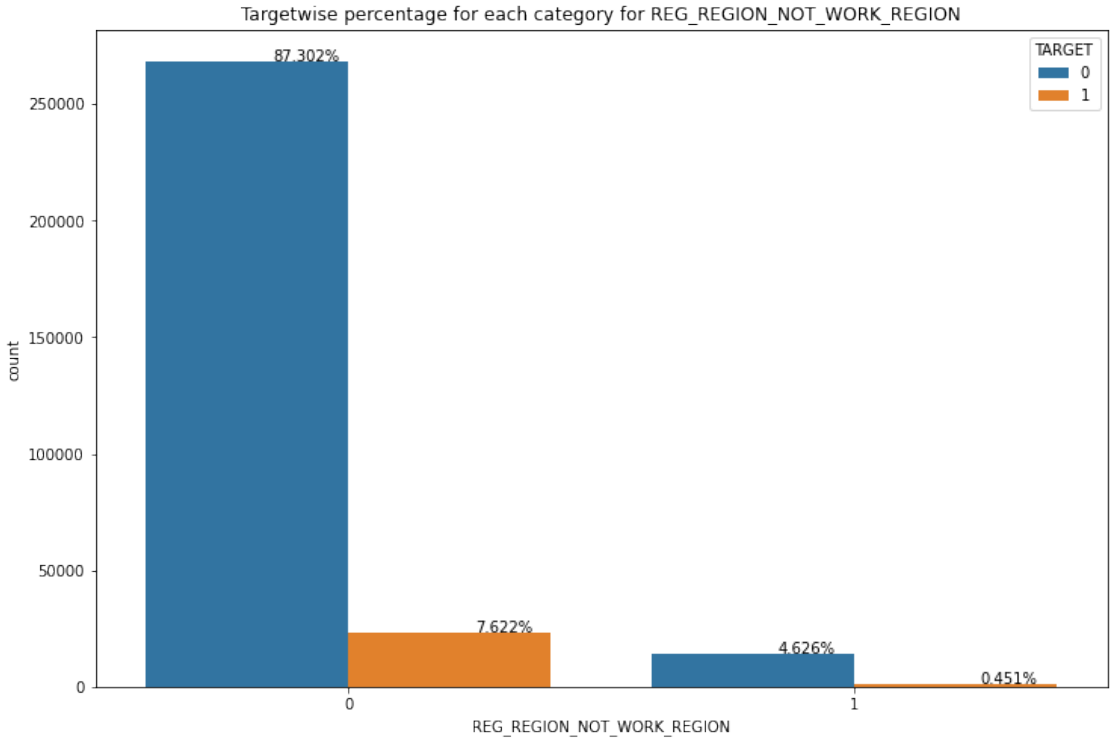
```
#lets see its distribution

ax=sns.countplot(x='REG_REGION_NOT_WORK_REGION',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("REG_REGION_NOT_WORK_REGION histogram")
```



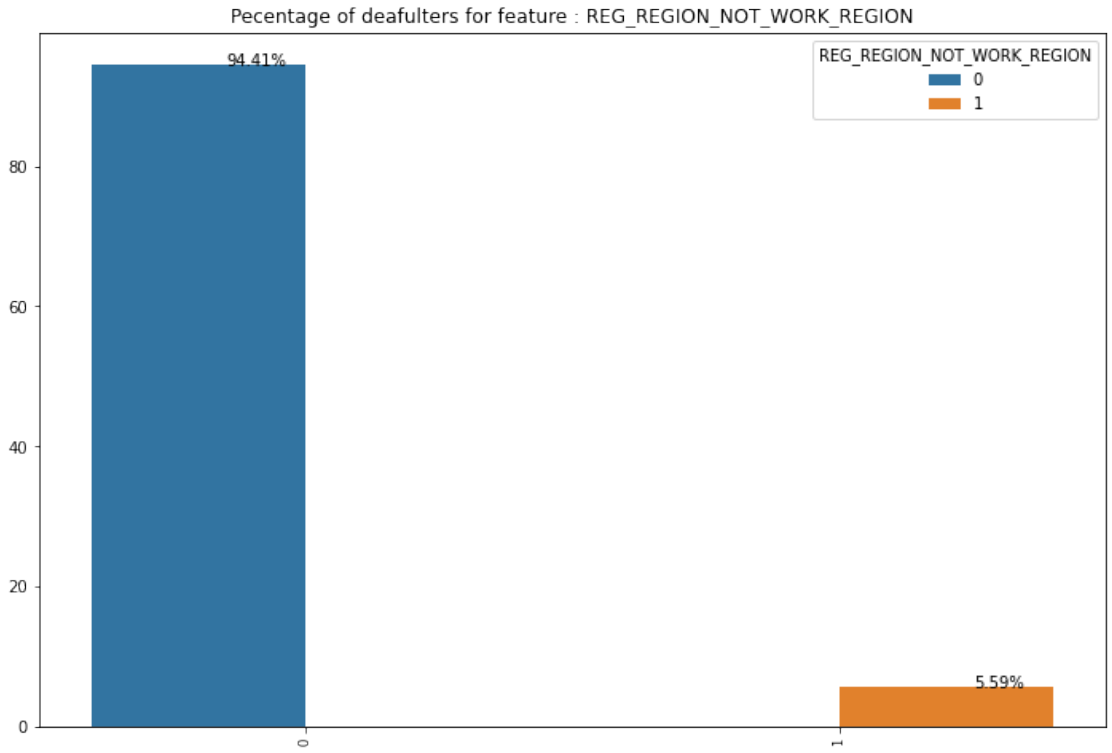
Most loan seekers are actually living at work area

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='REG_REGION_NOT_WORK_REGION',hue='TARGET',data=df_t
rain)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
REG_REGION_NOT_WORK_REGION")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
REG_REGION_NOT_WORK_REGION')
```




```
plot_for_categorical_defaulters(df_train,'REG_REGION_NOT_WORK_REGION')
```



-> Most client are living are living on their permannet adresses

- Only 5 % of defaulter clients are not living in their work city

- we can keep this column as this have better variations

```
#top 10 correlated columns
df_train.corr()
[['REG_REGION_NOT_WORK_REGION']].sort_values('REG_REGION_NOT_WORK_REGI
ON',ascending=False)[:10]
```

```
                              REG_REGION_NOT_WORK_REGION
REG_REGION_NOT_WORK_REGION                      1.000000
LIVE_REGION_NOT_WORK_REGION                     0.860627
REG_REGION_NOT_LIVE_REGION                      0.450804
REG_CITY_NOT_WORK_CITY                          0.239765
LIVE_CITY_NOT_WORK_CITY                         0.197097
REG_CITY_NOT_LIVE_CITY                          0.151397
FLAG_EMP_PHONE                                  0.108355
DAYS_BIRTH                                      0.095819
AMT_ANNUITY                                     0.079420
HOUR_APPR_PROCESS_START                         0.075037
```

- Highly correlated to LIVE_REGION_NOT_WORK_REGION, slightly correlated REG_REGION_NOT_LIVE_REGION

LIVE_REGION_NOT_WORK_REGION

Flag if client's contact address does not match work address (1=different, 0=same, at city level)

```
df_train['LIVE_REGION_NOT_WORK_REGION'].value_counts()
```

```
0    295008
1     12503
Name: LIVE_REGION_NOT_WORK_REGION, dtype: int64
```

```
df_train['REG_REGION_NOT_WORK_REGION'].isna().sum()
#No NA values
```

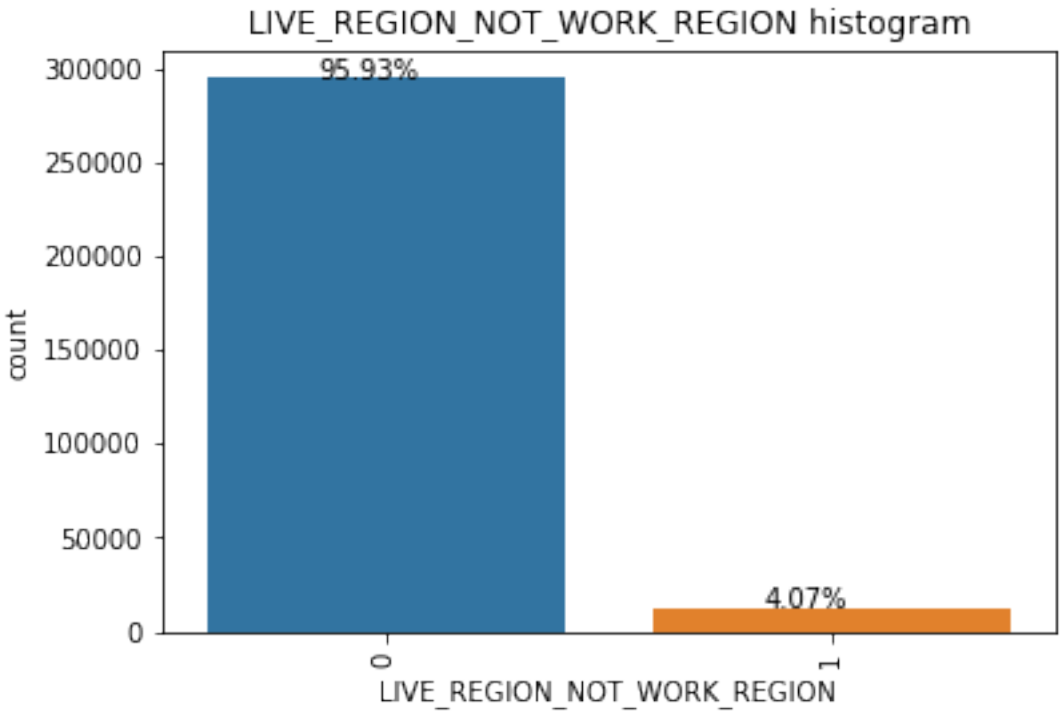
```
0
```

```
df_train['LIVE_REGION_NOT_WORK_REGION'].describe()
```

```
count   307511.000000
mean         0.040659
std          0.197499
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          1.000000
Name: LIVE_REGION_NOT_WORK_REGION, dtype: float64
```

```
#lets see its distribution

ax=sns.countplot(x='LIVE_REGION_NOT_WORK_REGION',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("LIVE_REGION_NOT_WORK_REGION histogram")
```

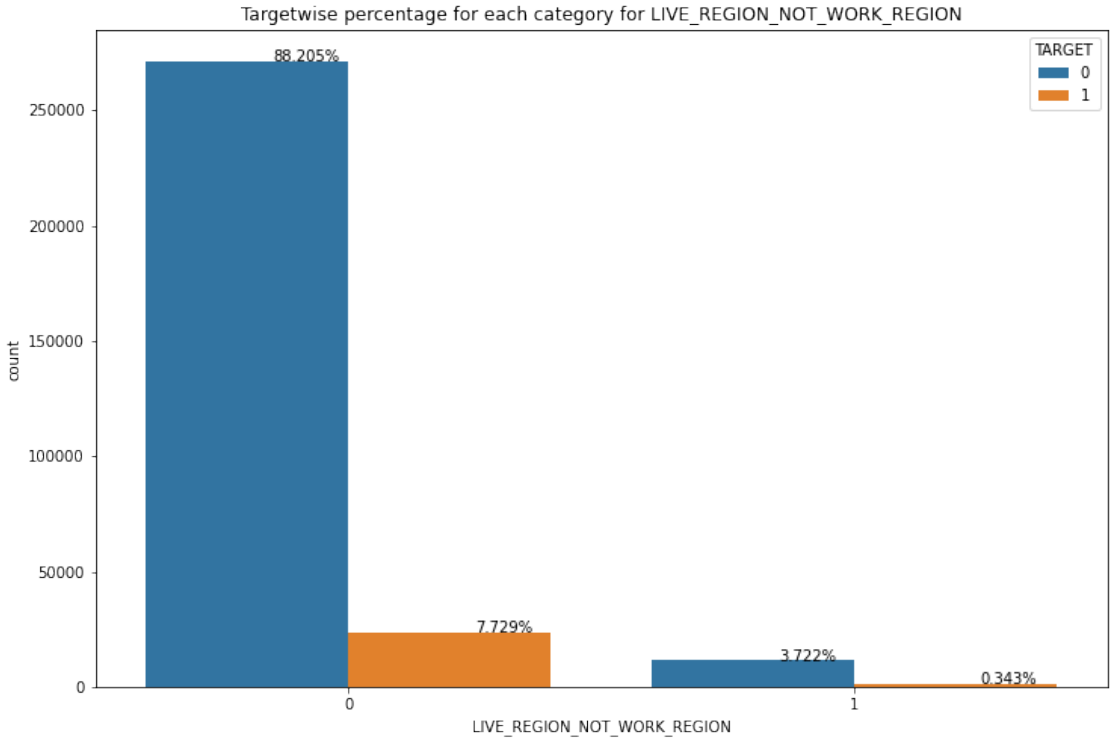
```
Text(0.5, 1.0, 'LIVE_REGION_NOT_WORK_REGION histogram')
```



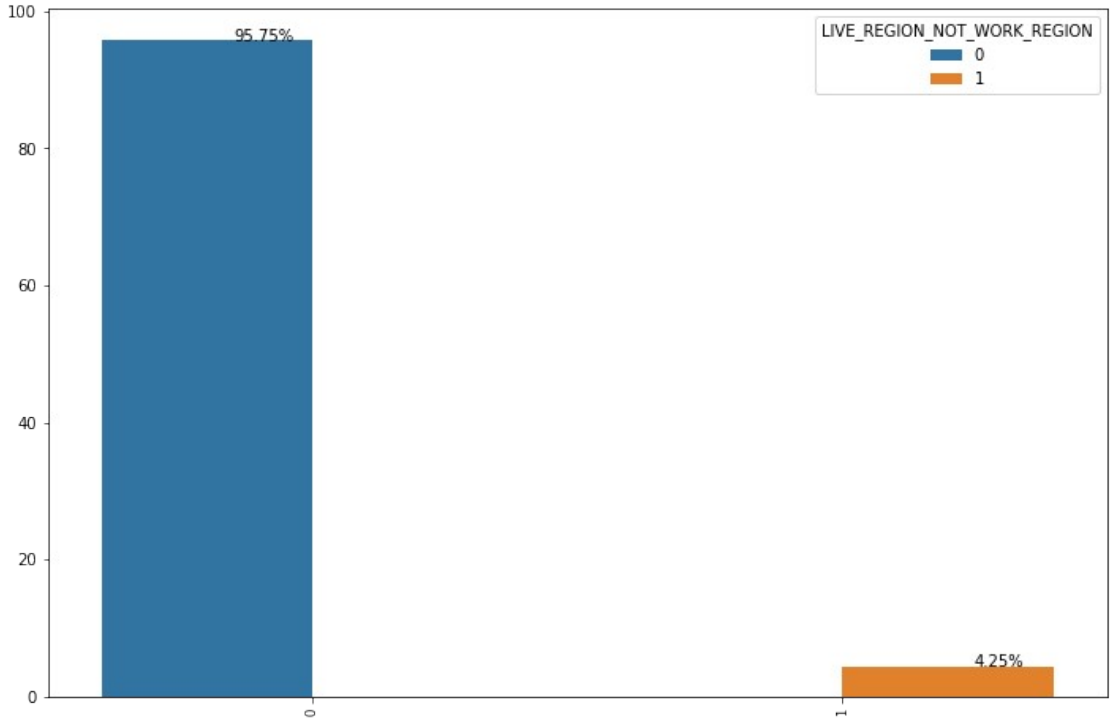
Most client are actually have contact adress as work address

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='LIVE_REGION_NOT_WORK_REGION',hue='TARGET',data=df_
train)
for p in ax.patches:
    ax.annotate('{:.3f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Targetwise percentage for each category for
LIVE_REGION_NOT_WORK_REGION")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
LIVE_REGION_NOT_WORK_REGION')
```




```
plot_for_categorical_defaulters(df_train,'LIVE_REGION_NOT_WORK_REGION'
)
```



- Only 4 % of defaulter client's contact address does not match work address
- we can drop this column as low variance

```
#top 10 correlated columns
df_train.corr()
[['LIVE_REGION_NOT_WORK_REGION']].sort_values('LIVE_REGION_NOT_WORK_RE
GION',ascending=False)[:10]
```

```
                            LIVE_REGION_NOT_WORK_REGION
LIVE_REGION_NOT_WORK_REGION                     1.000000
REG_REGION_NOT_WORK_REGION                      0.860627
LIVE_CITY_NOT_WORK_CITY                         0.236635
REG_CITY_NOT_WORK_CITY                          0.185863
FLAG_EMP_PHONE                                  0.096447
REG_REGION_NOT_LIVE_REGION                      0.088263
REGION_POPULATION_RELATIVE                      0.081016
AMT_ANNUITY                                     0.074550
DAYS_BIRTH                                      0.069567
HOUR_APPR_PROCESS_START                         0.060920
```

REG_CITY_NOT_LIVE_CITY

Flag if client's permanent address does not match contact address (1=different, 0=same, at city level

```
df_train['REG_CITY_NOT_LIVE_CITY'].value_counts()
```

```
0    283472
1     24039
Name: REG_CITY_NOT_LIVE_CITY, dtype: int64
```

```
df_train['REG_CITY_NOT_LIVE_CITY'].describe()
```

```
count   307511.000000
mean         0.078173
std          0.268444
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          1.000000
Name: REG_CITY_NOT_LIVE_CITY, dtype: float64
```

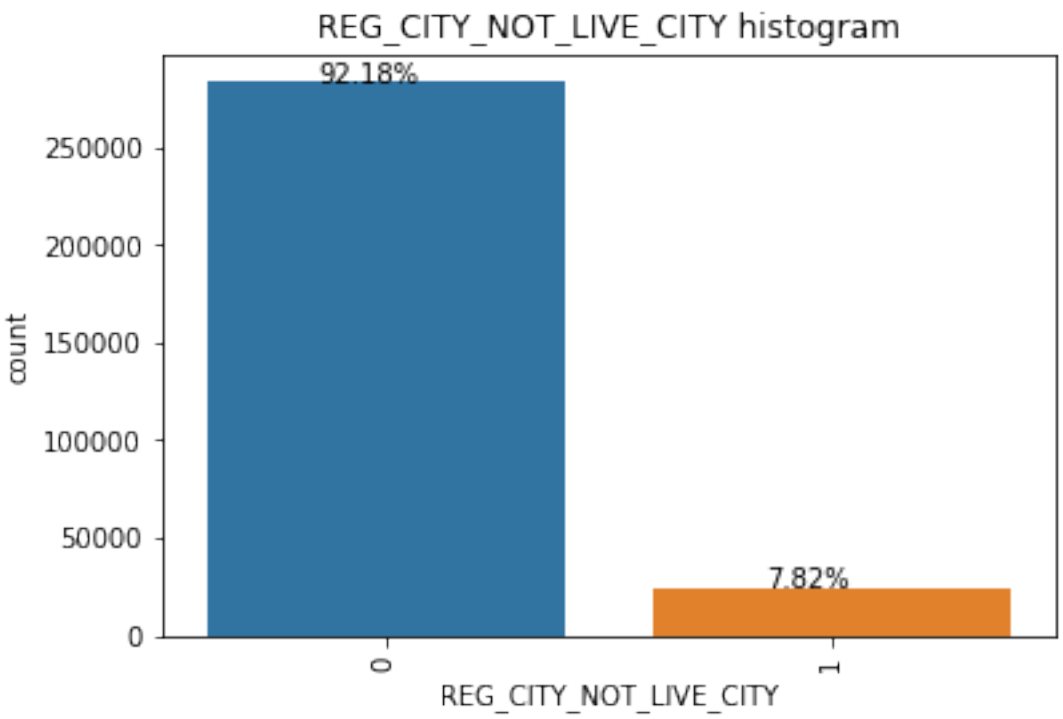
```
df_train['REG_REGION_NOT_WORK_REGION'].isna().sum()
#No NA values
```

```
0
```

```
#lets see its distribution

ax=sns.countplot(x='REG_CITY_NOT_LIVE_CITY',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
```

```
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("REG_CITY_NOT_LIVE_CITY histogram")
```

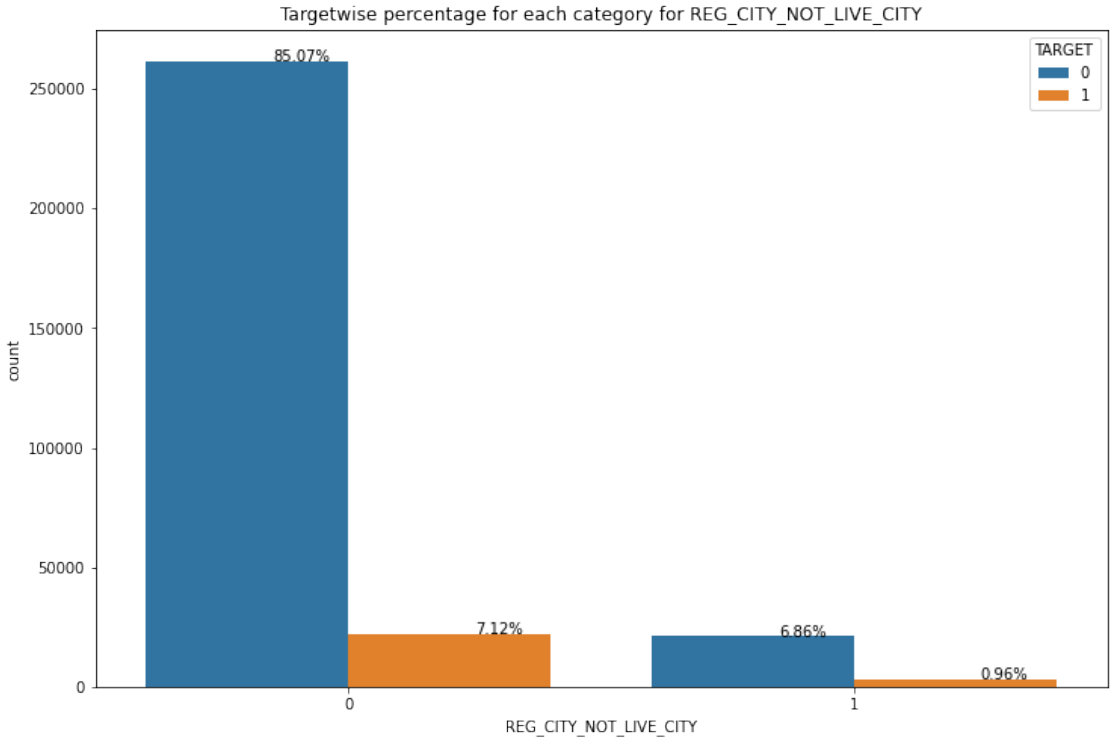
```
Text(0.5, 1.0, 'REG_CITY_NOT_LIVE_CITY histogram')
```



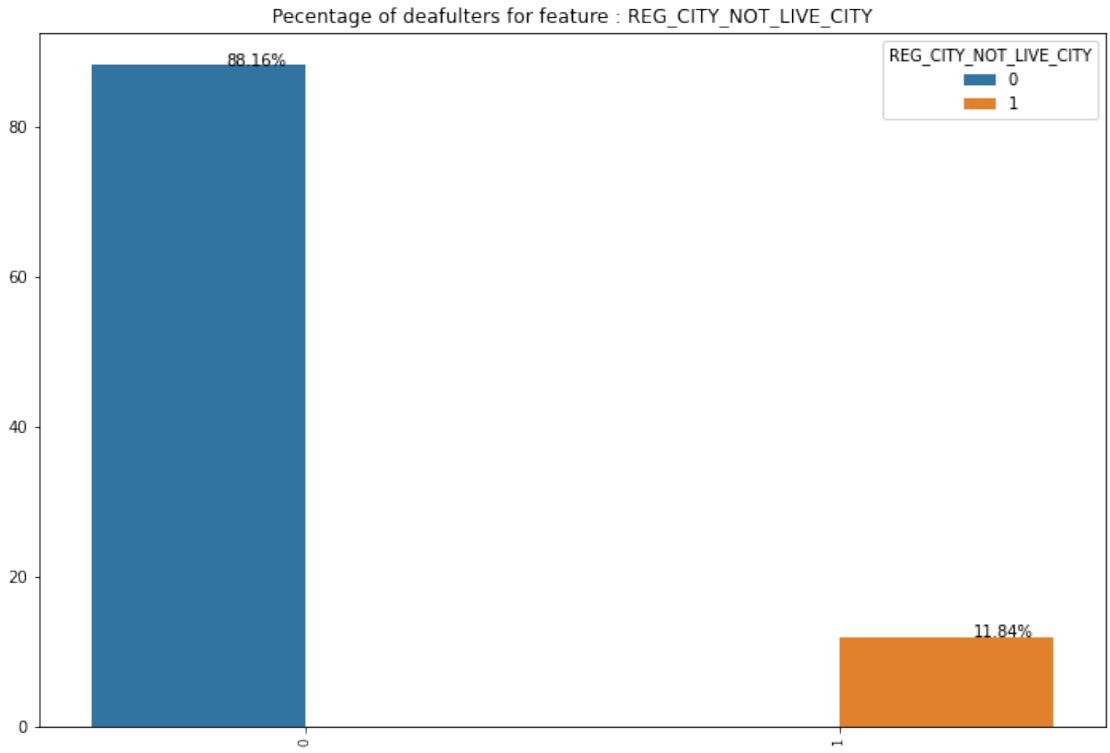
Most client are actually have contact adress as permanent address

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='REG_CITY_NOT_LIVE_CITY',hue='TARGET',data=df_train
)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
REG_CITY_NOT_LIVE_CITY")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
REG_CITY_NOT_LIVE_CITY')
```




```
plot_for_categorical_defaulters(df_train,'REG_CITY_NOT_LIVE_CITY')
```



- 12% of defaulter's contact address doesnt match with permanent adresss

```
#top 10 correlated columns
df_train.corr()
[['REG_CITY_NOT_LIVE_CITY']].sort_values('REG_CITY_NOT_LIVE_CITY',asce
nding=False)[:5]
```

```
                           REG_CITY_NOT_LIVE_CITY
REG_CITY_NOT_LIVE_CITY                   1.000000
REG_CITY_NOT_WORK_CITY                   0.440409
REG_REGION_NOT_LIVE_REGION               0.339232
DAYS_BIRTH                               0.180382
REG_REGION_NOT_WORK_REGION               0.151397
```

REG_CITY_NOT_WORK_CITY

Flag if client's permanent address does not match work address (1=different, 0=same, at city level)

```
df_train['REG_CITY_NOT_WORK_CITY'].value_counts()
```

```
0    236644
1     70867
Name: REG_CITY_NOT_WORK_CITY, dtype: int64
```

```
df_train['REG_REGION_NOT_WORK_REGION'].isna().sum()
#No NA values
```

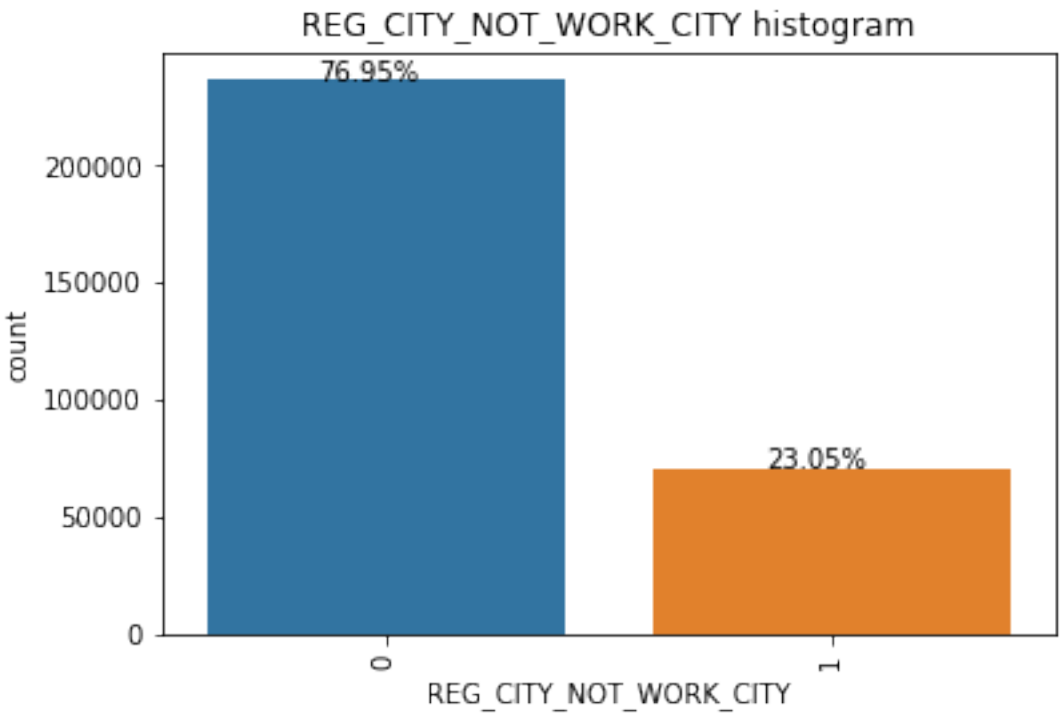
```
0
```

```
df_train['REG_CITY_NOT_WORK_CITY'].describe()
```

```
count   307511.000000
mean         0.230454
std          0.421124
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          1.000000
Name: REG_CITY_NOT_WORK_CITY, dtype: float64
```

```
#lets see its distribution

ax=sns.countplot(x='REG_CITY_NOT_WORK_CITY',data=df_train)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("REG_CITY_NOT_WORK_CITY histogram")
```

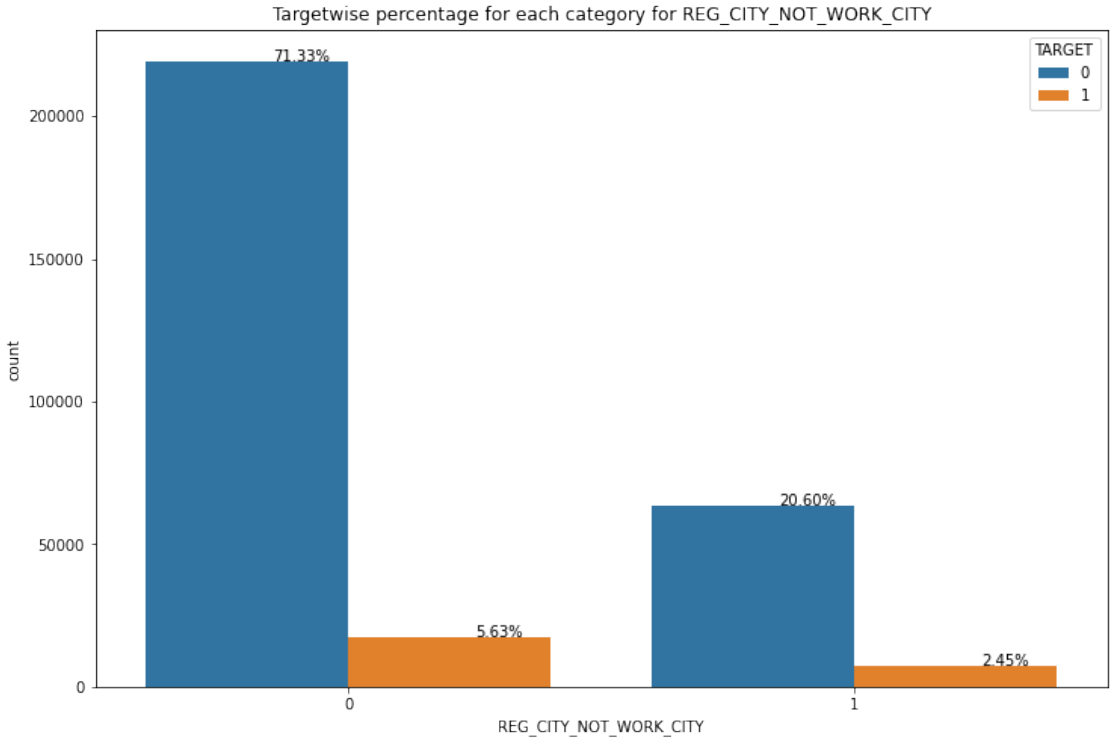
```
Text(0.5, 1.0, 'REG_CITY_NOT_WORK_CITY histogram')
```



Most client are actually have permanent adress as permanent address

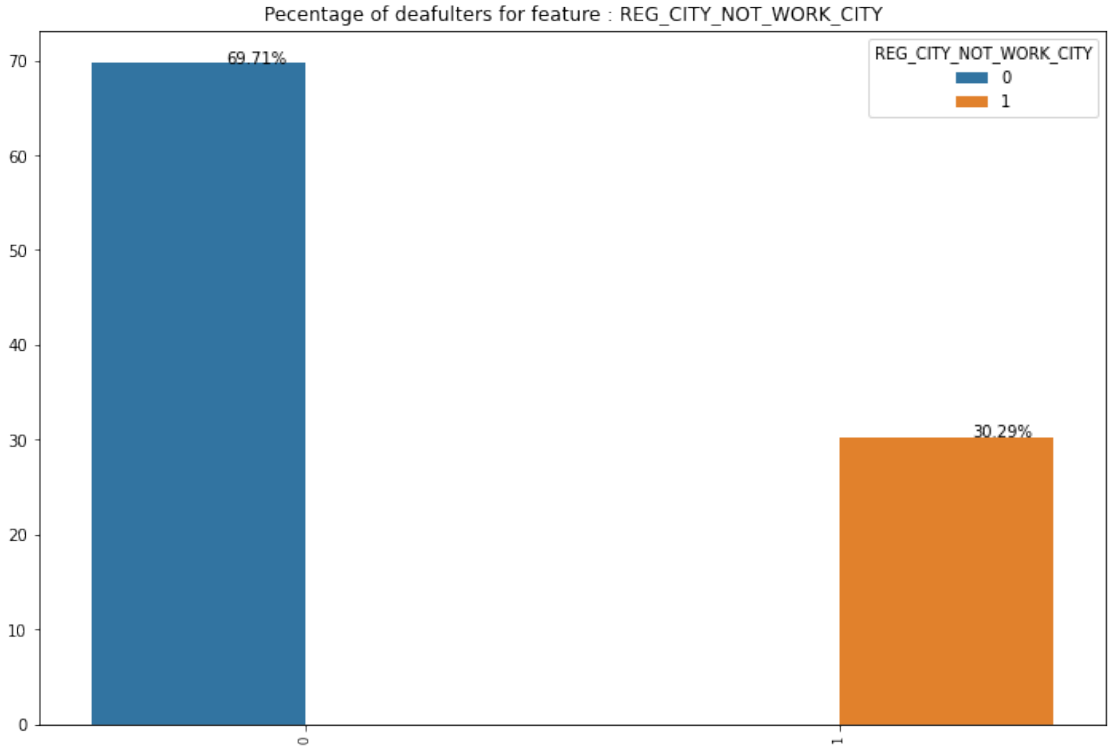
```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='REG_CITY_NOT_WORK_CITY',hue='TARGET',data=df_train
)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))

ax.set_title("Targetwise percentage for each category for
REG_CITY_NOT_WORK_CITY")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
REG_CITY_NOT_WORK_CITY')
```



- In Both category number of defaulters are lesser

```
plot_for_categorical_defaulters(df_train,'REG_CITY_NOT_WORK_CITY')
```



- 30 % of defaulter's work address doesnt match with permanent adresss

```
#top 10 correlated columns
df_train.corr()
[['REG_CITY_NOT_WORK_CITY']].sort_values('REG_CITY_NOT_WORK_CITY',asce
nding=False)[:5]
```

```
                        REG_CITY_NOT_WORK_CITY
REG_CITY_NOT_WORK_CITY                 1.000000
LIVE_CITY_NOT_WORK_CITY                0.825575
REG_CITY_NOT_LIVE_CITY                 0.440409
FLAG_EMP_PHONE                         0.256427
DAYS_BIRTH                             0.242401
```

- Highly correlated with LIVE_CITY_NOT_WORK_CITY
- slightly correlated with REG_CITY_NOT_LIVE_CITY

LIVE_CITY_NOT_WORK_CITY

Flag if client's contact address does not match work address (1=different, 0=same, at city level)

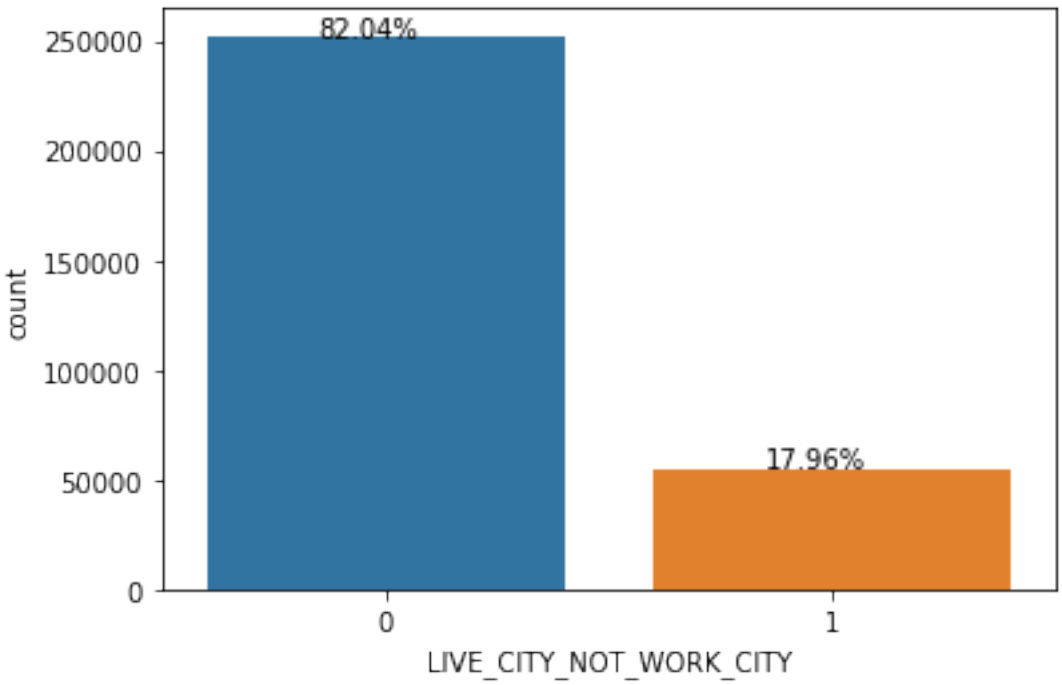
```
df_train['LIVE_CITY_NOT_WORK_CITY'].value_counts()
```

```
0    252296
1     55215
Name: LIVE_CITY_NOT_WORK_CITY, dtype: int64
```

```
df_train['LIVE_CITY_NOT_WORK_CITY'].isna().sum()
#No NA values
```

```
0
```

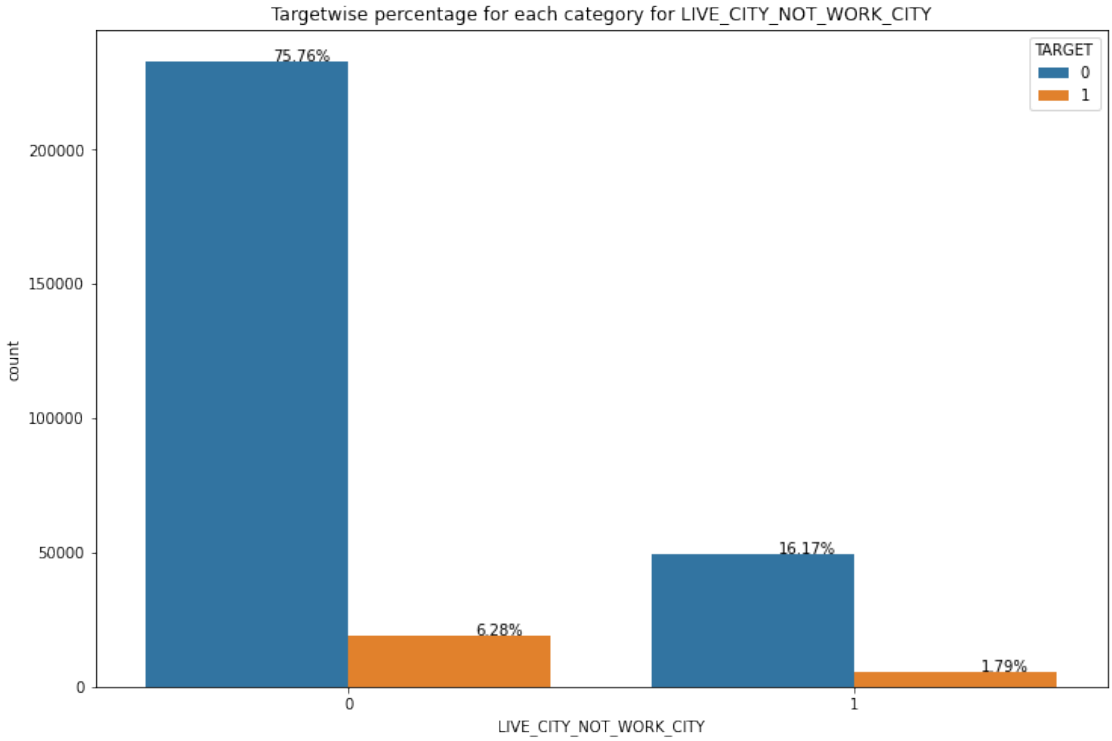
```
#lets see its distribution

ax=sns.countplot(x='LIVE_CITY_NOT_WORK_CITY',data=df_train)
#plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("LIVE_CITY_NOT_WORK_CITY histogram")
```



Most client's contact address does match work address

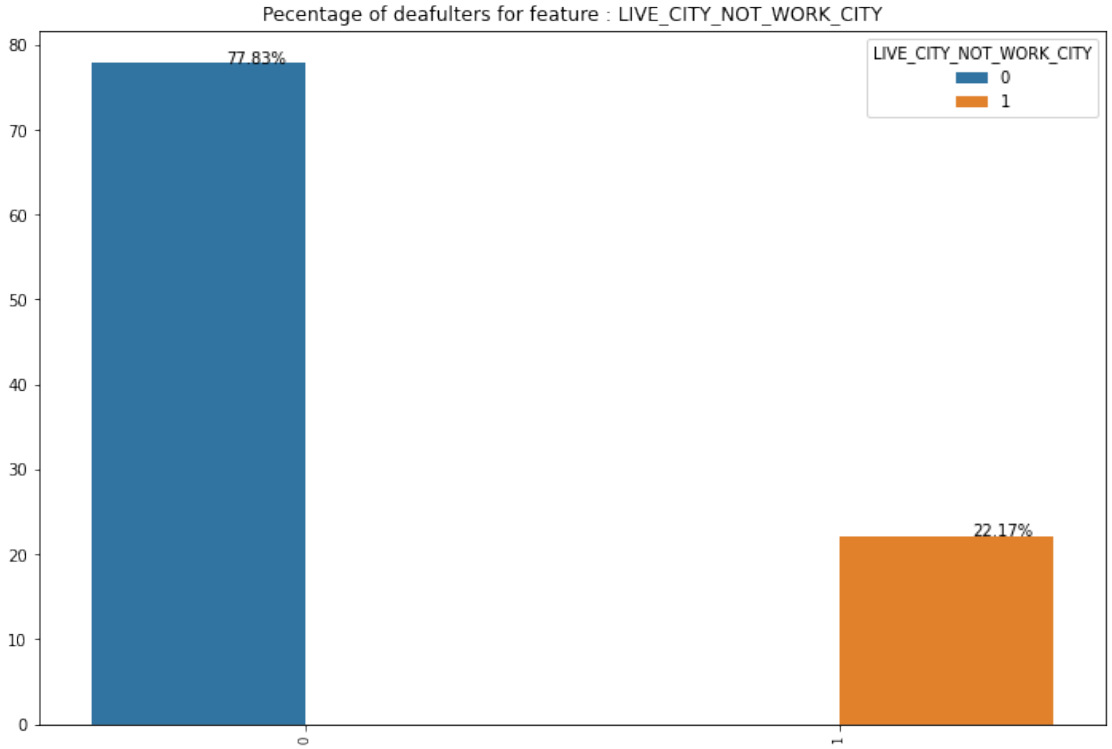
```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='LIVE_CITY_NOT_WORK_CITY',hue='TARGET',data=df_trai
n)
for p in ax.patches:
    ax.annotate('{:.2f}%'.format(p.get_height()*100/len(df_train)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("Targetwise percentage for each category for
LIVE_CITY_NOT_WORK_CITY")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
LIVE_CITY_NOT_WORK_CITY')
```



- In Both category number of defaulters are lesser

```
plot_for_categorical_defaulters(df_train,'LIVE_CITY_NOT_WORK_CITY')
```



- 22 % of defaulter client's contact address does not match work address

```
#top 10 correlated columns
df_train.corr()
[['LIVE_CITY_NOT_WORK_CITY']].sort_values('LIVE_CITY_NOT_WORK_CITY',as
cending=False)[:5]
```

```
                            LIVE_CITY_NOT_WORK_CITY
LIVE_CITY_NOT_WORK_CITY                    1.000000
REG_CITY_NOT_WORK_CITY                     0.825575
LIVE_REGION_NOT_WORK_REGION                0.236635
FLAG_EMP_PHONE                             0.219219
REG_REGION_NOT_WORK_REGION                 0.197097
```

- Highly correlated with REG_CITY_NOT_WORK_CITY

I think we can aggregate that columns and create a single feature

- 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

EXT_SOURCE_1,EXT_SOURCE_2,EXT_SOURCE_3

Normalized score from external data source 1,2,3

```
ext_source=['EXT_SOURCE_1','EXT_SOURCE_2','EXT_SOURCE_3']
```

```
for i in ext_source:
    print(i + "\n")
    print(df_train[i].describe())
    print('\n')
```

```
EXT_SOURCE_1

count    134133.000000
mean          0.502130
std           0.211062
min           0.014568
25%           0.334007
50%           0.505998
75%           0.675053
max           0.962693
Name: EXT_SOURCE_1, dtype: float64


EXT_SOURCE_2
```

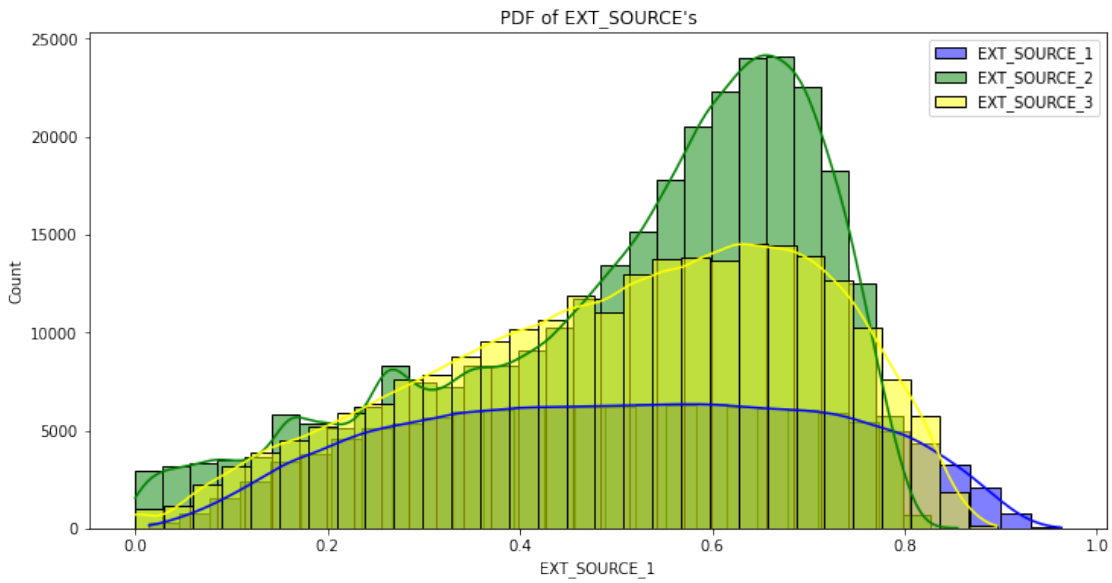
```
count    3.068510e+05
mean     5.143927e-01
std      1.910602e-01
min      8.173617e-08
25%      3.924574e-01
50%      5.659614e-01
75%      6.636171e-01
max      8.549997e-01
Name: EXT_SOURCE_2, dtype: float64


EXT_SOURCE_3

count    246546.000000
mean          0.510853
std           0.194844
min           0.000527
25%           0.370650
50%           0.535276
75%           0.669057
max           0.896010
Name: EXT_SOURCE_3, dtype: float64
```

```python
#Lets see count of NA value

for i in ext_source:
    print(i + "\n")
    print(df_train[i].isna().sum())
    print('\n')
```

```
EXT_SOURCE_1

173378


EXT_SOURCE_2

660


EXT_SOURCE_3

60965
```

```
plt.figure(figsize=(12,6))
sns.histplot(df_train['EXT_SOURCE_1'],bins=30,kde=True,label='EXT_SOUR
CE_1',color='blue')
sns.histplot(df_train['EXT_SOURCE_2'],bins=30,kde=True,label='EXT_SOUR
CE_2',color="green")
sns.histplot(df_train['EXT_SOURCE_3'],bins=30,kde=True,label='EXT_SOUR
CE_3',color="yellow")
#plt.xlim([40000,5000000])
plt.title("PDF of EXT_SOURCE's")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1e235c9eee0>
```



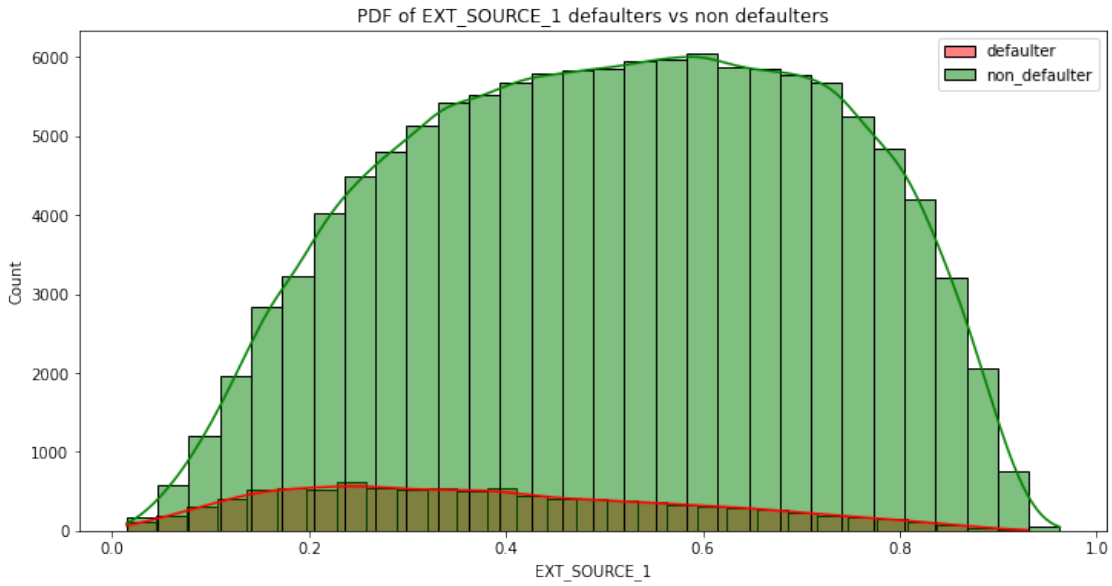
EXT_SOURCE_1 seems to be more uniform mostly high density lies between 0.2 to 0.85

EXT_SOURCE_2 seems to be little normal mostly high density lies between 0.2 to 0.8

EXT_SOURCE_3 seems to be more normal mostly high density lies between 0.5 to 0.7

```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['EXT_SOURCE_1'],bins=30,kde=True,label='defa
ulter',color='red')
sns.histplot(df_non_defaulter['EXT_SOURCE_1'],bins=30,kde=True,label='
non_defaulter',color="green")
#plt.xlim([1000,300000])
#plt.xticks(np.arange(1000,50000,10))
plt.title("PDF of EXT_SOURCE_1 defaulters vs non defaulters")
plt.legend()
```

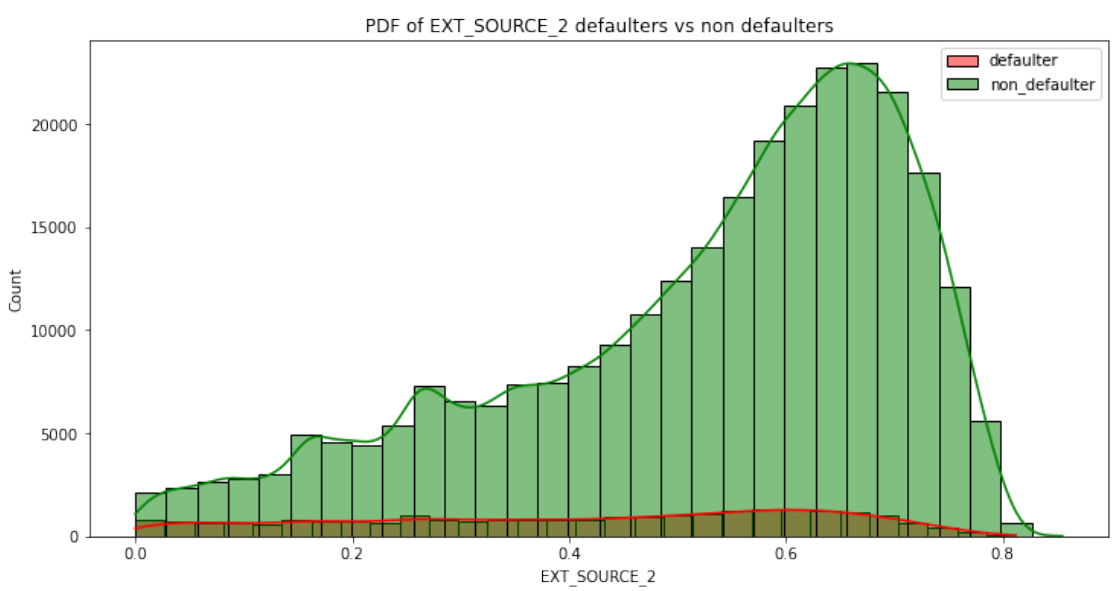
```
<matplotlib.legend.Legend at 0x1e27d113520>
```



- It seems that most defaulters lies below EXT_SOurce 0.1 to 0.6,very few with high score EX1

```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['EXT_SOURCE_2'],bins=30,kde=True,label='defa
ulter',color='red')
sns.histplot(df_non_defaulter['EXT_SOURCE_2'],bins=30,kde=True,label='
non_defaulter',color="green")
#plt.xlim([1000,300000])
#plt.xticks(np.arange(1000,50000,10))
plt.title("PDF of EXT_SOURCE_2 defaulters vs non defaulters")
plt.legend()
```

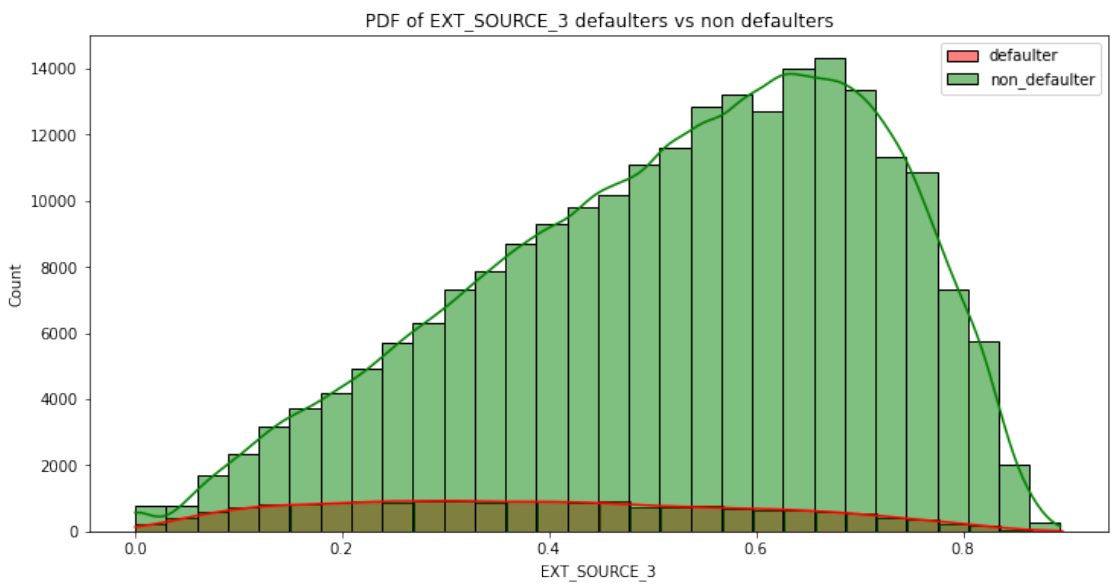
```
<matplotlib.legend.Legend at 0x1e2801f2c70>
```



- It seems that most defaulters lies all over the values of EXT_Source2

```
plt.figure(figsize=(12,6))
sns.histplot(df_defaulter['EXT_SOURCE_3'],bins=30,kde=True,label='defa
ulter',color='red')
sns.histplot(df_non_defaulter['EXT_SOURCE_3'],bins=30,kde=True,label='
non_defaulter',color="green")
#plt.xlim([1000,300000])
#plt.xticks(np.arange(1000,50000,10))
plt.title("PDF of EXT_SOURCE_3 defaulters vs non defaulters")
plt.legend()
```

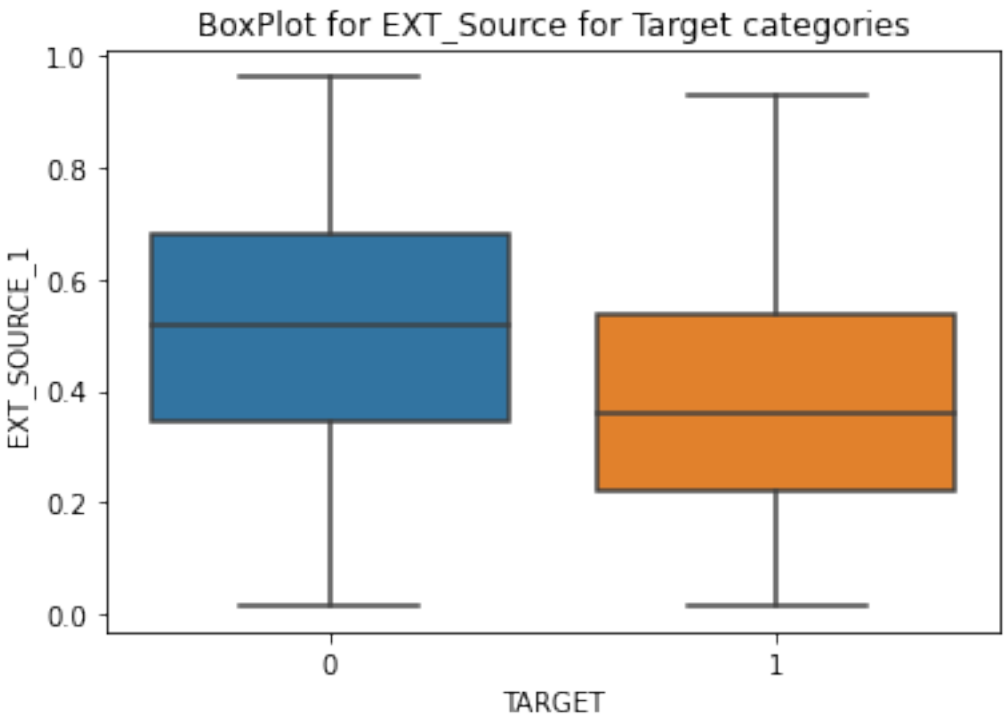
```
<matplotlib.legend.Legend at 0x1e24dddb190>
```



- It seems that most defaulters lies all over the values of EXT_Source3 as well

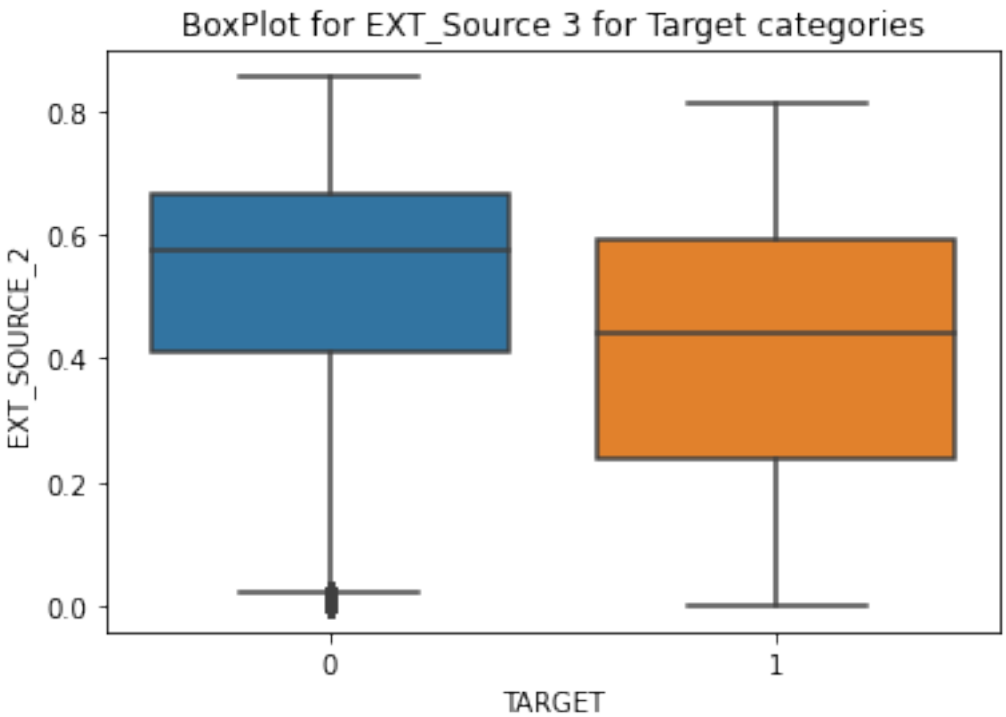
```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="EXT_SOURCE_1", data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for EXT_Source 1 for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for EXT_Source for Target categories')
```



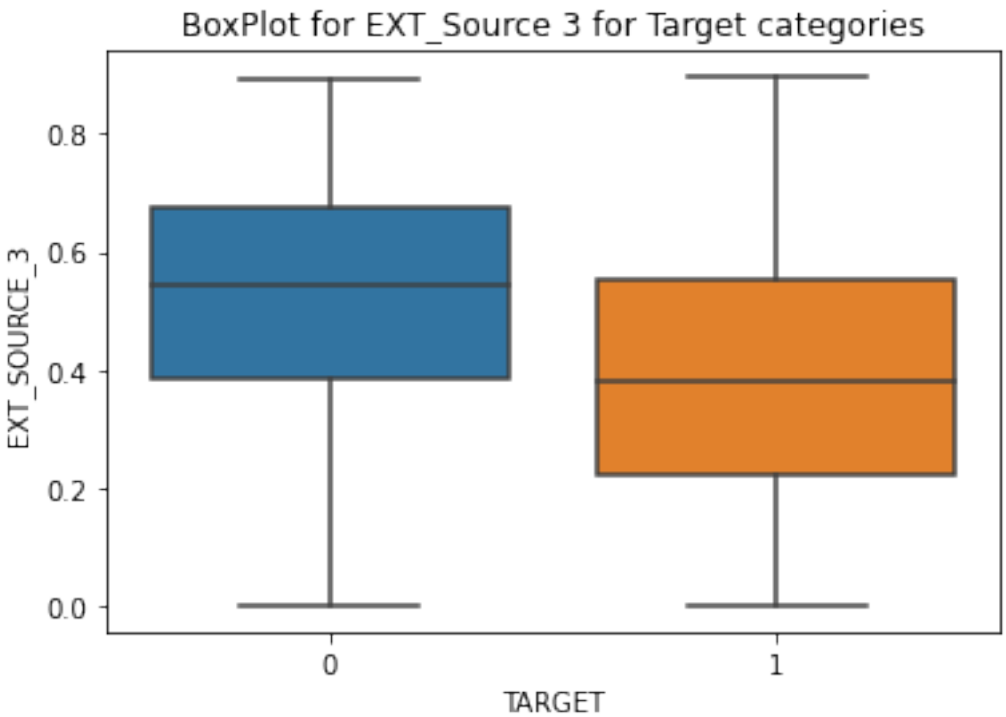

```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="EXT_SOURCE_2", data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for EXT_Source 3 for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for EXT_Source 3 for Target categories')
```




```
#lets see the box plot
ax = sns.boxplot(x="TARGET", y="EXT_SOURCE_3", data=df_train)
#plt.ylim([10,5000000])
plt.title("BoxPlot for EXT_Source 3 for Target categories")
```

```
Text(0.5, 1.0, 'BoxPlot for EXT_Source 3 for Target categories')
```




```
#top 10 correlated column
df_train.corr()
[['EXT_SOURCE_1']].sort_values('EXT_SOURCE_1',ascending=False)[:4]
```

```
               EXT_SOURCE_1
EXT_SOURCE_1       1.000000
Client_Age         0.600333
DAYS_EMPLOYED      0.289848
EXT_SOURCE_2       0.213982
```

```
#top 10 correlated column
df_train.corr()
[['EXT_SOURCE_2']].sort_values('EXT_SOURCE_2',ascending=False)[:4]
```

```
                            EXT_SOURCE_2
EXT_SOURCE_2                    1.000000
EXT_SOURCE_1                    0.213982
REGION_POPULATION_RELATIVE      0.198924
HOUR_APPR_PROCESS_START         0.156863
```

```
#top 10 correlated column
df_train.corr()
[['EXT_SOURCE_3']].sort_values('EXT_SOURCE_3',ascending=False)[:4]
```

```
               EXT_SOURCE_3
EXT_SOURCE_3       1.000000
Client_Age         0.205492
EXT_SOURCE_1       0.186846
DAYS_EMPLOYED      0.113434
```

- EXT_SOURCE_1 & EXT_SOURCE_3 are well correlated with Client_Age
- EXT_SOURCE_2 is slightly correlate with EXT_SOURCE_1
- EXT_SOURCE_3 is slightly correlate with EXT_SOURCE_1

Now we can focus on some building properties

```
#For Avg columns
avg= [col for col in df_train.columns if col.split("_")[-1]=='AVG']
#For median values
median= [col for col in df_train.columns if col.split("_")[-
1]=='MEDI']
#for mode values
mode= [col for col in df_train.columns if col.split("_")[-1]=='MODE']
```

```
len(avg)
```

```
14
```

```
len(median)
```

```
14
```

```
len(mode)
```

```
19
```

```
set([i.split('_')[0] for i in mode]).difference(set([i.split('_')[0]
for i in avg]))
```

```
{'EMERGENCYSTATE', 'FONDKAPREMONT', 'HOUSETYPE', 'TOTALAREA',
'WALLSMATERIAL'}
```

```
# we have already studied the all above columns except TOTALAREA_MODE
```

TOTALAREA_MODE

TOTALAREA of place where client lives

```
df_train['TOTALAREA_MODE'].describe()
```

```
count   159080.000000
mean         0.102547
std          0.107462
min          0.000000
25%          0.041200
50%          0.068800
75%          0.127600
max          1.000000
Name: TOTALAREA_MODE, dtype: float64
```

```
df_train['TOTALAREA_MODE'].isna().sum()
```

```
148431
```

```
df_test['TOTALAREA_MODE'].isna().sum()
```

```
22624
```

```
#lets see how many are zeros
df_train[df_train['TOTALAREA_MODE']==0].shape[0]
```

```
582
```

```
#top 10 correlated column with DAYS_REGISTRATION
df_train.corr()
[['TOTALAREA_MODE']].sort_values('TOTALAREA_MODE',ascending=False)
[:10]
```

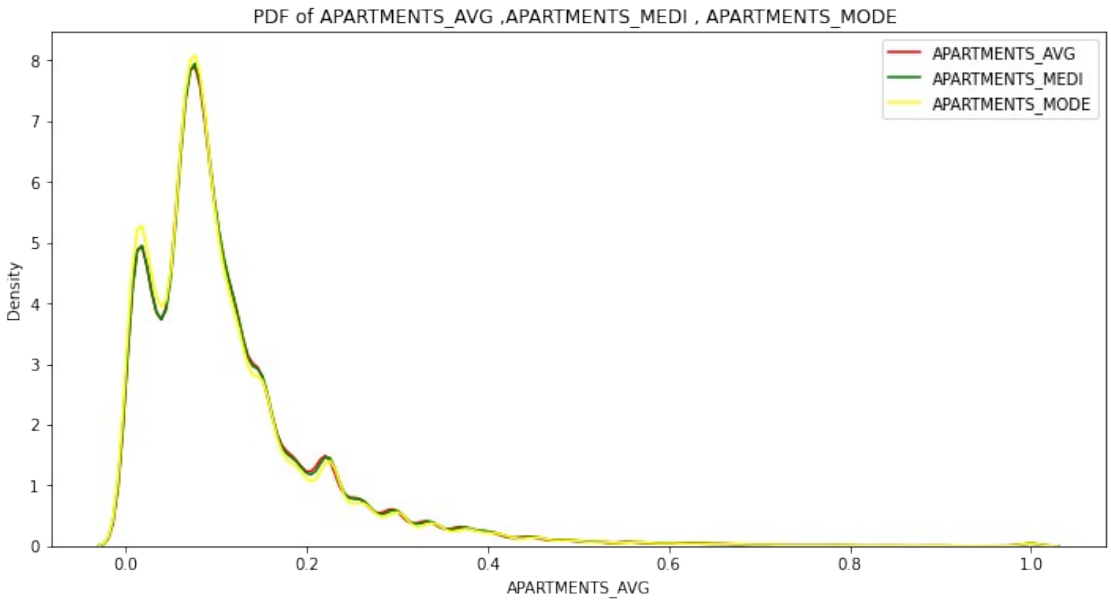
```
                        TOTALAREA_MODE
TOTALAREA_MODE                1.000000
LIVINGAREA_AVG                0.925039
LIVINGAREA_MEDI               0.919363
LIVINGAREA_MODE               0.899247
APARTMENTS_AVG                0.892627
APARTMENTS_MEDI               0.886568
APARTMENTS_MODE               0.863628
LIVINGAPARTMENTS_AVG          0.848028
LIVINGAPARTMENTS_MEDI         0.846105
ELEVATORS_AVG                 0.844628
```

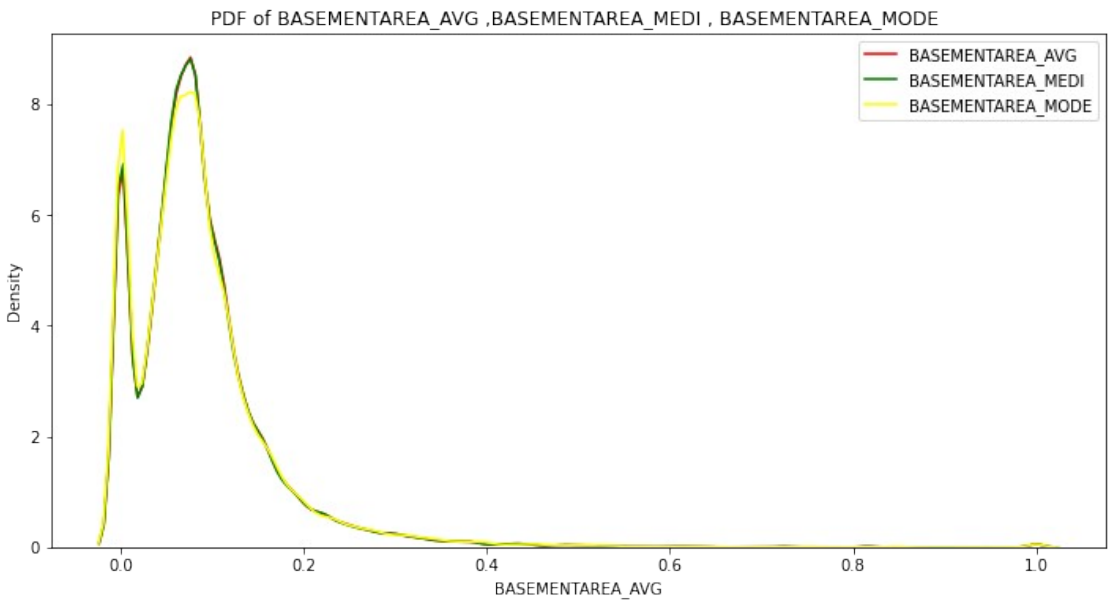
- we can see this column is higly correlated with so many column we can drop it

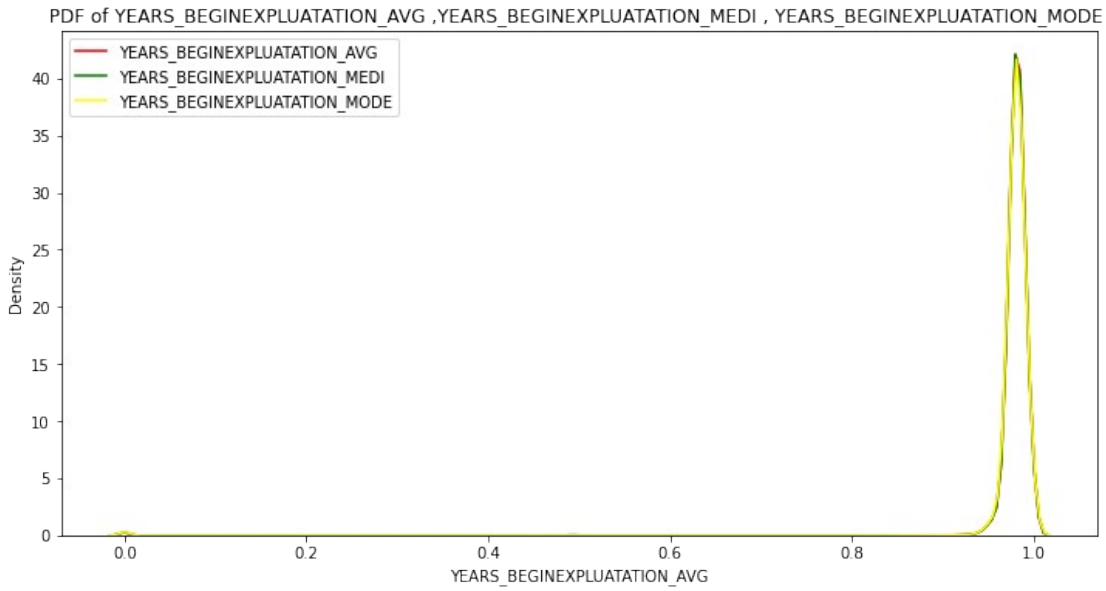
lets see building properties with its mean,meadian and AVG

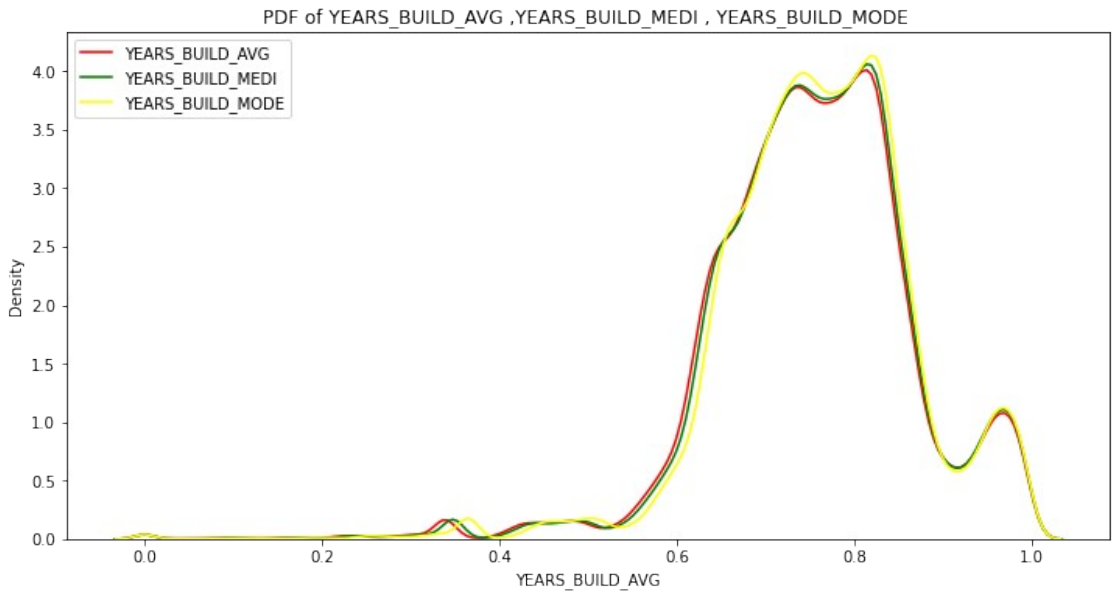
```
def building_properties_stats(feature1,feature2,feature3):
    plt.figure(figsize=(12,6))
    sns.kdeplot(df_train[feature1],label=feature1,color='red')
    sns.kdeplot(df_train[feature2],label=feature2,color="green")
    sns.kdeplot(df_train[feature3],label=feature3,color="yellow")
    #plt.xlim([40000,5000000])
    plt.title(f"PDF of {feature1} ,{feature2} , {feature3} ")
    plt.legend()
    print("\n \n ")
```

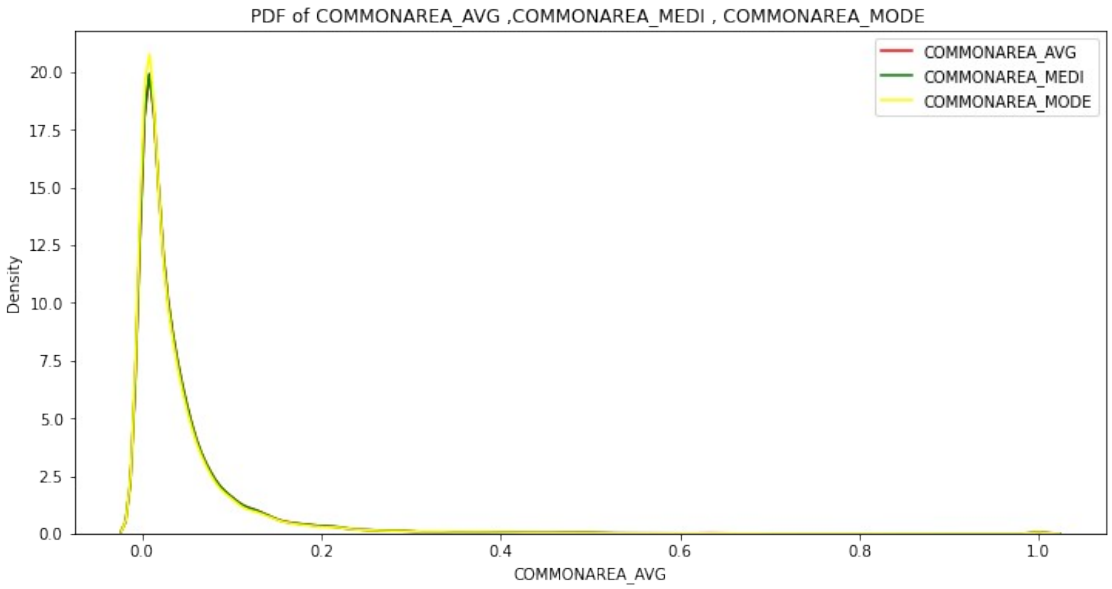
```
for i,j ,k in zip(avg,median,mode[:15]):
    building_properties_stats(i,j,k)
```

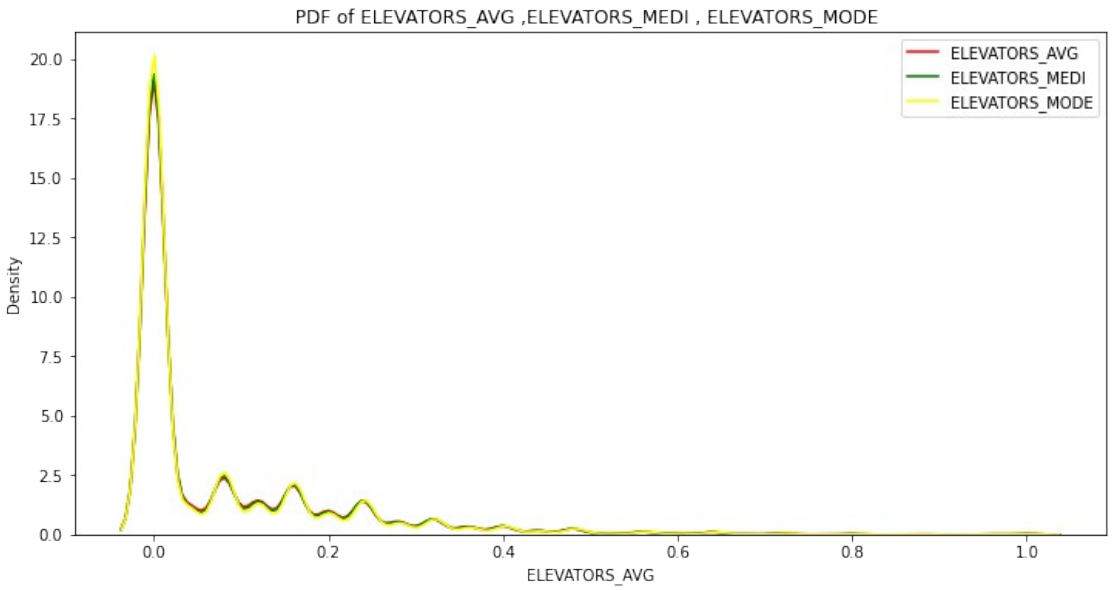

PDF of APARTMENTS_AVG ,APARTMENTS_MEDI , APARTMENTS_MODE
APARTMENTS_AVG
APARTMENTS_MEDI
APARTMENTS_MODE
Density
APARTMENTS_AVG

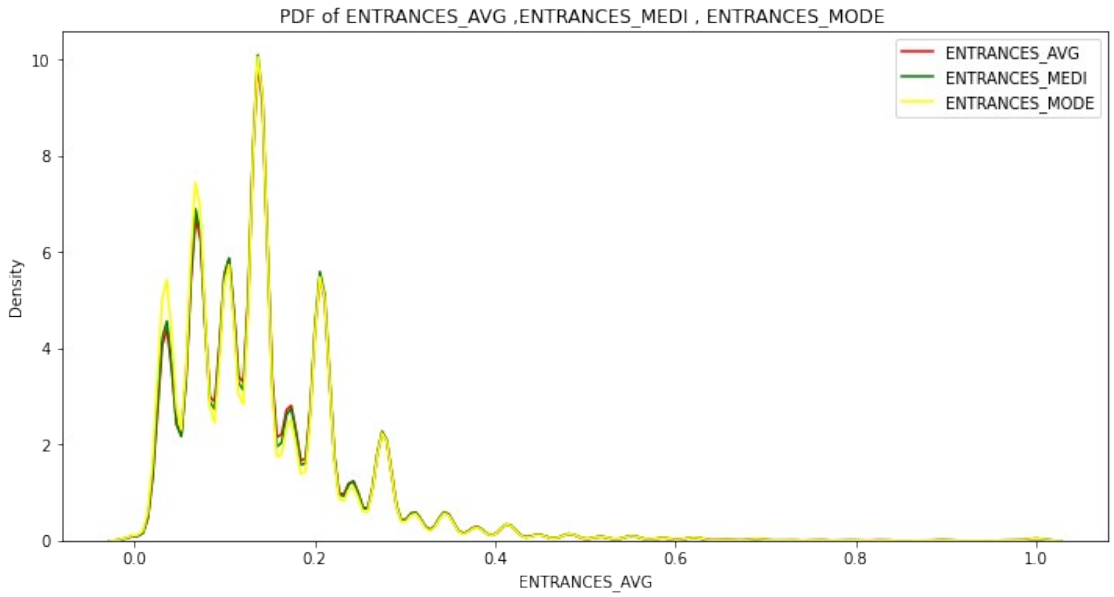

PDF of BASEMENTAREA_AVG ,BASEMENTAREA_MEDI , BASEMENTAREA_MODE
BASEMENTAREA_AVG
BASEMENTAREA_MEDI
BASEMENTAREA_MODE
Density
BASEMENTAREA_AVG

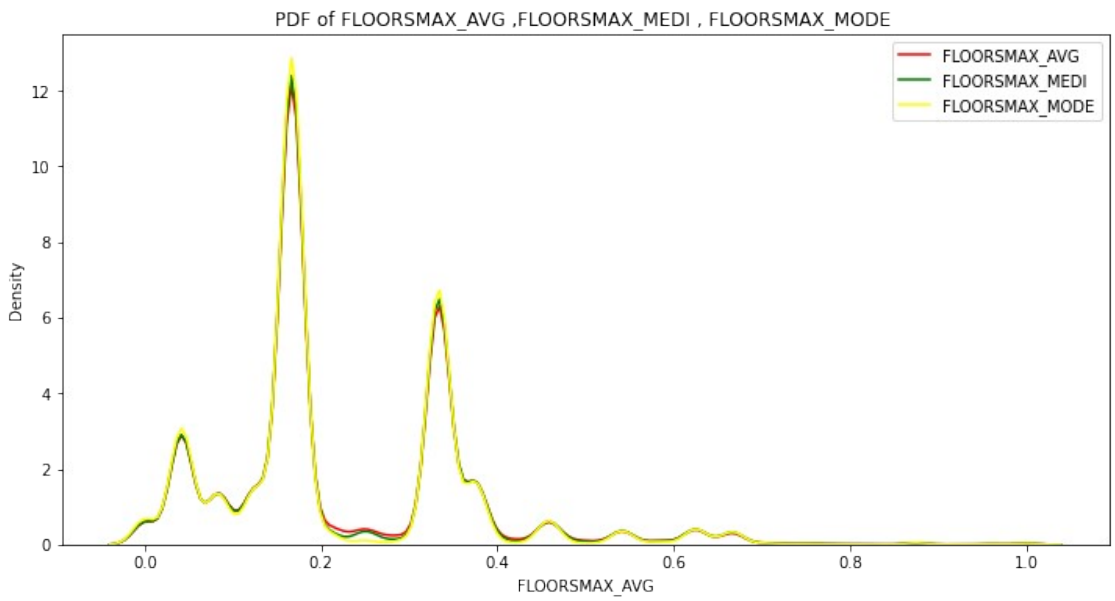

PDF of YEARS_BEGINEXPLUATATION_AVG ,YEARS_BEGINEXPLUATATION_MEDI , YEARS_BEGINEXPLUATATION_MODE
YEARS_BEGINEXPLUATATION_AVG
YEARS_BEGINEXPLUATATION_MEDI
YEARS_BEGINEXPLUATATION_MODE
Density
YEARS_BEGINEXPLUATATION_AVG

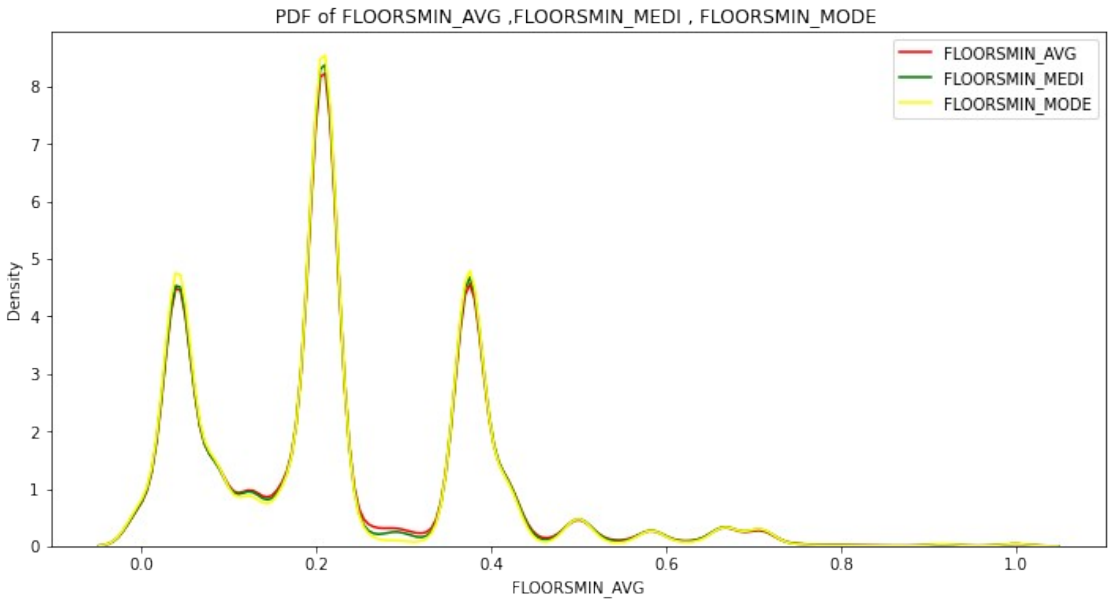

PDF of YEARS_BUILD_AVG ,YEARS_BUILD_MEDI , YEARS_BUILD_MODE
YEARS_BUILD_AVG
YEARS_BUILD_MEDI
YEARS_BUILD_MODE
Density
YEARS_BUILD_AVG

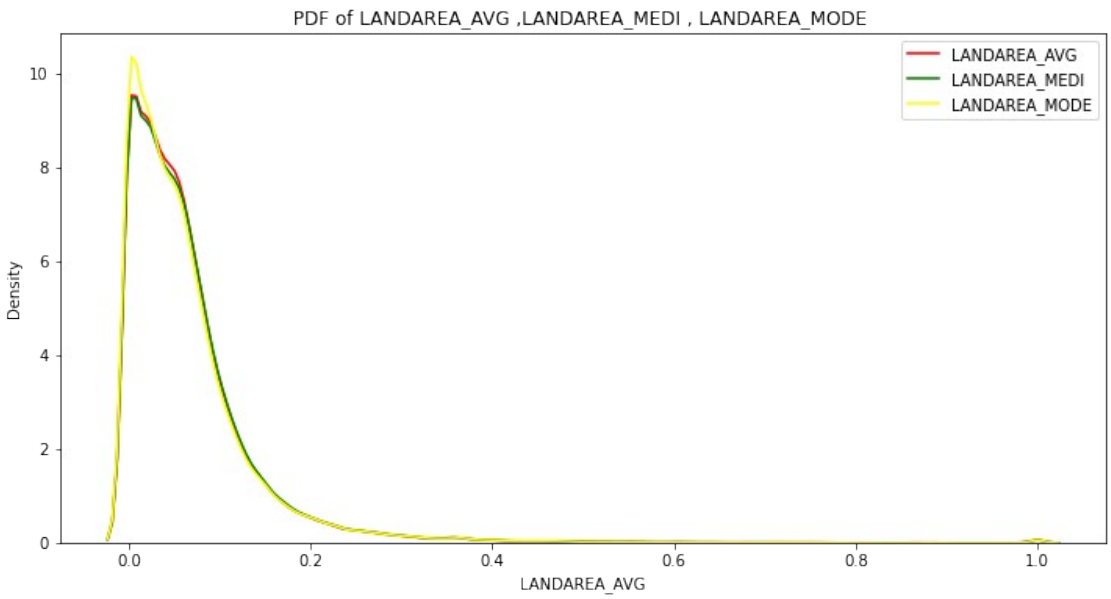

PDF of COMMONAREA_AVG ,COMMONAREA_MEDI , COMMONAREA_MODE
COMMONAREA_AVG
COMMONAREA_MEDI
COMMONAREA_MODE
Density
COMMONAREA_AVG

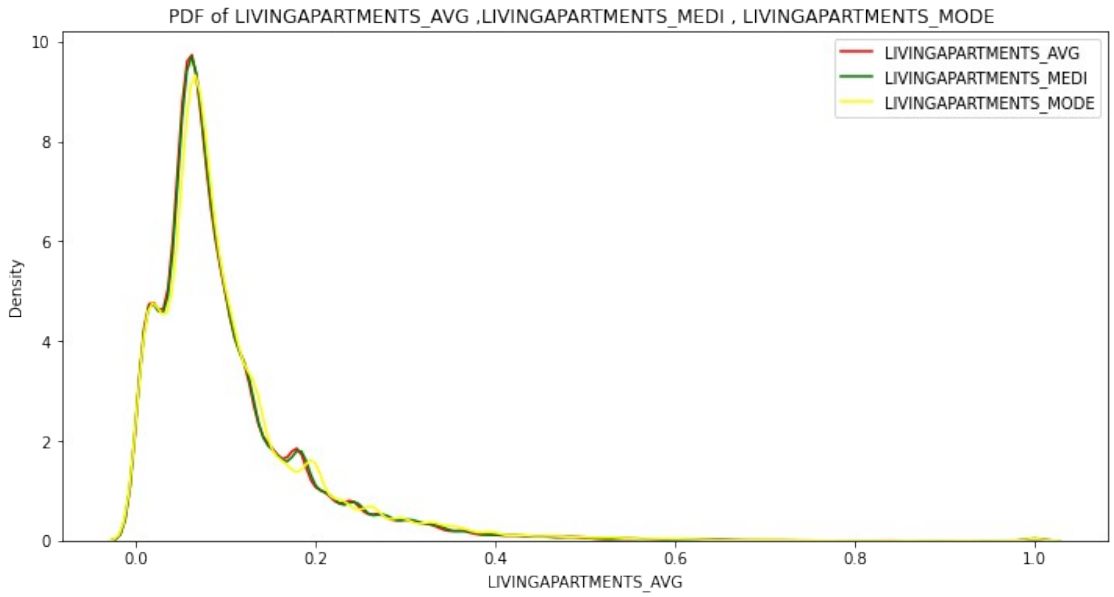

PDF of ELEVATORS_AVG ,ELEVATORS_MEDI , ELEVATORS_MODE
ELEVATORS_AVG
ELEVATORS_MEDI
ELEVATORS_MODE
Density
ELEVATORS_AVG

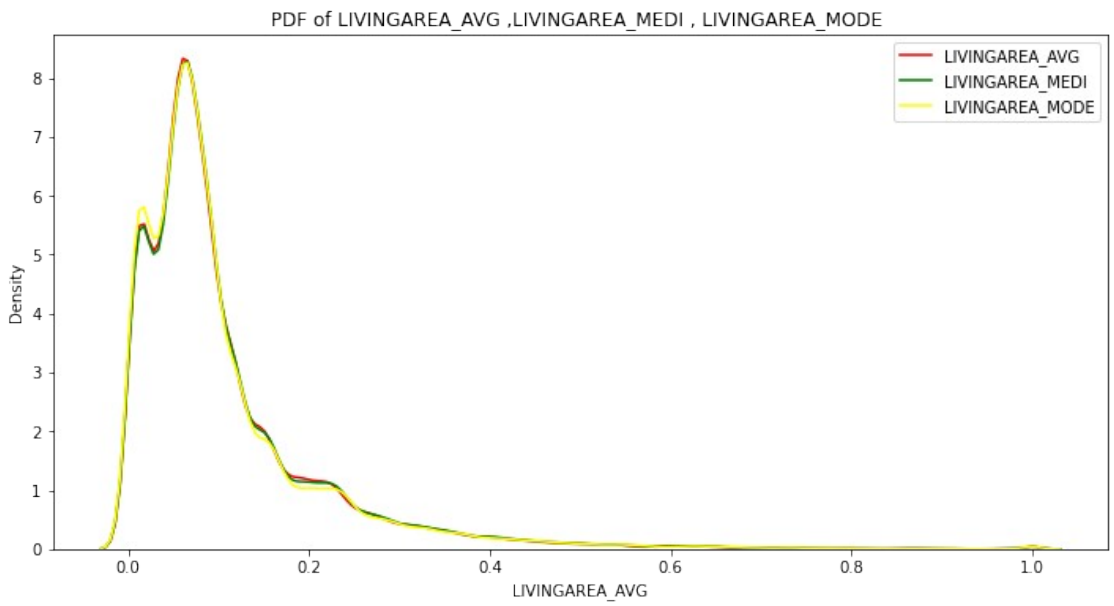

PDF of ENTRANCES_AVG ,ENTRANCES_MEDI , ENTRANCES_MODE
ENTRANCES_AVG
ENTRANCES_MEDI
ENTRANCES_MODE
Density
ENTRANCES_AVG

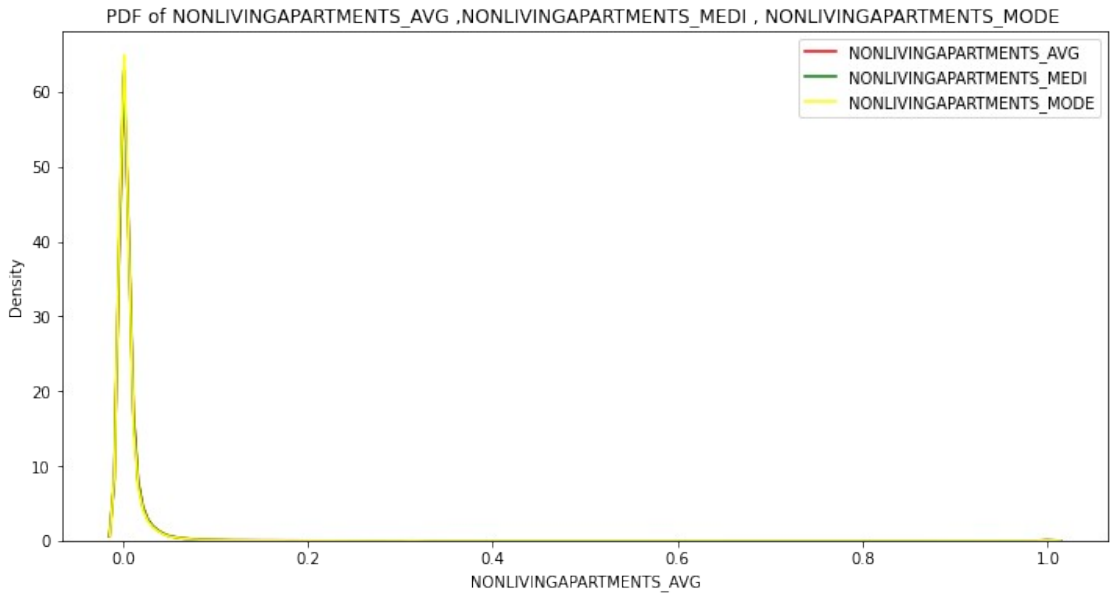

PDF of FLOORSMAX_AVG ,FLOORSMAX_MEDI , FLOORSMAX_MODE
FLOORSMAX_AVG
FLOORSMAX_MEDI
FLOORSMAX_MODE
Density
FLOORSMAX_AVG

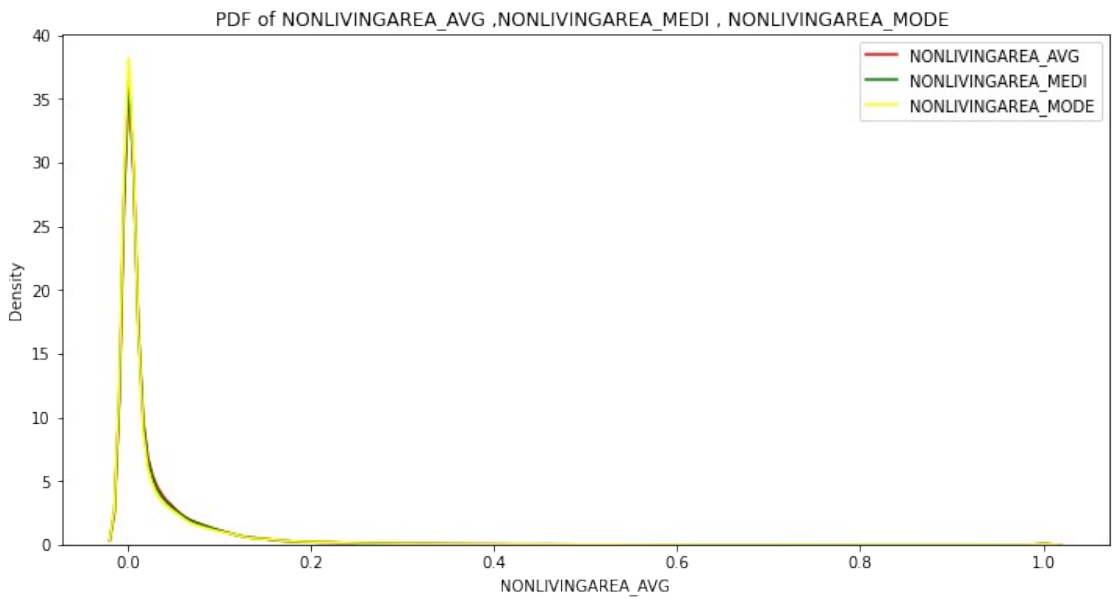

PDF of FLOORSMIN_AVG ,FLOORSMIN_MEDI , FLOORSMIN_MODE
FLOORSMIN_AVG
FLOORSMIN_MEDI
FLOORSMIN_MODE
Density
FLOORSMIN_AVG


PDF of LANDAREA_AVG ,LANDAREA_MEDI , LANDAREA_MODE
LANDAREA_AVG
LANDAREA_MEDI
LANDAREA_MODE
Density
LANDAREA_AVG


PDF of LIVINGAPARTMENTS_AVG ,LIVINGAPARTMENTS_MEDI , LIVINGAPARTMENTS_MODE
LIVINGAPARTMENTS_AVG
LIVINGAPARTMENTS_MEDI
LIVINGAPARTMENTS_MODE
Density
LIVINGAPARTMENTS_AVG


PDF of LIVINGAREA_AVG ,LIVINGAREA_MEDI , LIVINGAREA_MODE
LIVINGAREA_AVG
LIVINGAREA_MEDI
LIVINGAREA_MODE
Density
LIVINGAREA_AVG






```
# ALl features mean ,median and mode are almost same, we can keep
columns which have least NA values

count_na_dict= {}
for i,j ,k in zip(avg,median,mode[:15]):
    count_na_dict[i]= df_train[i].isna().sum()
    count_na_dict[j]= df_train[j].isna().sum()
    count_na_dict[k]= df_train[k].isna().sum()

len(count_na_dict)
```

42

avg

```
['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']
```

```
avg.append('TARGET')
```

```
avg[0]
```

```
'APARTMENTS_AVG'
```

```
df_train[avg].corr()['TARGET'].sort_values(ascending=True)
```

```
FLOORSMAX_AVG                 -0.044003
ELEVATORS_AVG                 -0.034199
FLOORSMIN_AVG                 -0.033614
LIVINGAREA_AVG                -0.032997
APARTMENTS_AVG                -0.029498
LIVINGAPARTMENTS_AVG          -0.025031
BASEMENTAREA_AVG              -0.022746
YEARS_BUILD_AVG               -0.022149
ENTRANCES_AVG                 -0.019172
COMMONAREA_AVG                -0.018550
NONLIVINGAREA_AVG             -0.013578
LANDAREA_AVG                  -0.010885
YEARS_BEGINEXPLUATATION_AVG   -0.009728
NONLIVINGAPARTMENTS_AVG       -0.003176
TARGET                         1.000000
Name: TARGET, dtype: float64
```

```
df_train[avg].std().sort_values()
```

```
NONLIVINGAPARTMENTS_AVG        0.047732
YEARS_BEGINEXPLUATATION_AVG    0.059223
NONLIVINGAREA_AVG              0.069523
COMMONAREA_AVG                 0.076036
LANDAREA_AVG                   0.081184
BASEMENTAREA_AVG               0.082438
LIVINGAPARTMENTS_AVG           0.092576
ENTRANCES_AVG                  0.100049
APARTMENTS_AVG                 0.108240
LIVINGAREA_AVG                 0.110565
```

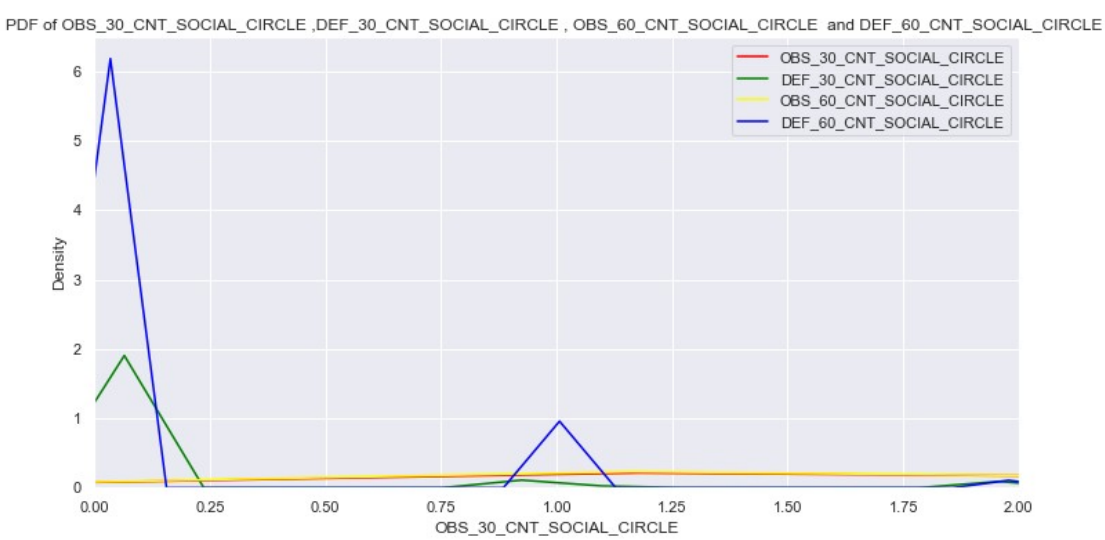
```
YEARS_BUILD_AVG                 0.113280
ELEVATORS_AVG                   0.134576
FLOORSMAX_AVG                   0.144641
FLOORSMIN_AVG                   0.161380
TARGET                          0.272419
dtype: float64
```

So we can see all type of feature have same value of NA values for mean ,median and mode

We can just keep one of type and drop others,Avg will be a better option

lets see SOCIAL_CIRCLE columns

```
social_circle=[ 'OBS_30_CNT_SOCIAL_CIRCLE','DEF_30_CNT_SOCIAL_CIRCLE',
'OBS_60_CNT_SOCIAL_CIRCLE','DEF_60_CNT_SOCIAL_CIRCLE']

sns.set(style="darkgrid")
plt.figure(figsize=(12,6))
sns.kdeplot(df_train[social_circle[0]],label=social_circle[0],color='r
ed')
sns.kdeplot(df_train[social_circle[1]],label=social_circle[1],color="g
reen")
sns.kdeplot(df_train[social_circle[2]],label=social_circle[2],color="y
ellow")
sns.kdeplot(df_train[social_circle[3]],label=social_circle[3],color="b
lue")
plt.xlim([0,2])
plt.title(f"PDF of {social_circle[0]} ,{social_circle[1]} ,
{social_circle[2]}  and {social_circle[3]} ")
plt.legend()
print("\n \n ")
```



```
count_na_dict_social_circle= {}
for i in social_circle:
    count_na_dict_social_circle[i]= df_train[i].isna().sum()
```

```
count_na_dict_social_circle
```

```
{'OBS_30_CNT_SOCIAL_CIRCLE': 1021,
 'DEF_30_CNT_SOCIAL_CIRCLE': 1021,
 'OBS_60_CNT_SOCIAL_CIRCLE': 1021,
 'DEF_60_CNT_SOCIAL_CIRCLE': 1021}
```

```
for i in social_circle:
    print(df_train[['TARGET',i]].corr()['TARGET'])
    print("\n")
```

```
TARGET                      1.000000
OBS_30_CNT_SOCIAL_CIRCLE    0.009131
Name: TARGET, dtype: float64


TARGET                      1.000000
DEF_30_CNT_SOCIAL_CIRCLE    0.032248
Name: TARGET, dtype: float64


TARGET                      1.000000
OBS_60_CNT_SOCIAL_CIRCLE    0.009022
Name: TARGET, dtype: float64


TARGET                      1.000000
DEF_60_CNT_SOCIAL_CIRCLE    0.031276
Name: TARGET, dtype: float64
```

- I think we can drop these columns

Lets see FLAG_DOCUMENT columns

- These are binary columns 1= yes ,0= No, means has client provide the document "N"

```
flag_doc_cols= [col for col in df_train.columns if 'FLAG_DOCUMENT' in
col]
```

```
flag_doc_cols
```

```
['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']
```

```
#Now we need to understand which document is most relevant of a client
being defaulter

count_na_dict_flag_doc= {}
for i in flag_doc_cols:
    count_na_dict_flag_doc[i]= df_train[i].isna().sum()

#count_na_dict_flag_doc
```

- No NA columns

```
#see for each flag document column which have Taregt-wise number of
doument submitted flag
for i in flag_doc_cols:
    print(f"for {i}: defaulters count")
    print(df_train.groupby(['TARGET'])[i].sum())
    print("\n")
```

```
for FLAG_DOCUMENT_2: defaulters count
TARGET
0    9
1    4
Name: FLAG_DOCUMENT_2, dtype: int64


for FLAG_DOCUMENT_3: defaulters count
TARGET
0    199028
1     19312
Name: FLAG_DOCUMENT_3, dtype: int64


for FLAG_DOCUMENT_4: defaulters count
TARGET
```

```
0    25
1     0
Name: FLAG_DOCUMENT_4, dtype: int64


for FLAG_DOCUMENT_5: defaulters count
TARGET
0    4276
1     372
Name: FLAG_DOCUMENT_5, dtype: int64


for FLAG_DOCUMENT_6: defaulters count
TARGET
0    25571
1     1507
Name: FLAG_DOCUMENT_6, dtype: int64


for FLAG_DOCUMENT_7: defaulters count
TARGET
0    56
1     3
Name: FLAG_DOCUMENT_7, dtype: int64


for FLAG_DOCUMENT_8: defaulters count
TARGET
0    23188
1     1836
Name: FLAG_DOCUMENT_8, dtype: int64


for FLAG_DOCUMENT_9: defaulters count
TARGET
0    1124
1      74
Name: FLAG_DOCUMENT_9, dtype: int64


for FLAG_DOCUMENT_10: defaulters count
TARGET
0    7
1    0
Name: FLAG_DOCUMENT_10, dtype: int64


for FLAG_DOCUMENT_11: defaulters count
TARGET
```

```
0    1128
1      75
Name: FLAG_DOCUMENT_11, dtype: int64


for FLAG_DOCUMENT_12: defaulters count
TARGET
0    2
1    0
Name: FLAG_DOCUMENT_12, dtype: int64


for FLAG_DOCUMENT_13: defaulters count
TARGET
0    1054
1      30
Name: FLAG_DOCUMENT_13, dtype: int64


for FLAG_DOCUMENT_14: defaulters count
TARGET
0    873
1     30
Name: FLAG_DOCUMENT_14, dtype: int64


for FLAG_DOCUMENT_15: defaulters count
TARGET
0    361
1     11
Name: FLAG_DOCUMENT_15, dtype: int64


for FLAG_DOCUMENT_16: defaulters count
TARGET
0    2903
1     150
Name: FLAG_DOCUMENT_16, dtype: int64


for FLAG_DOCUMENT_17: defaulters count
TARGET
0    80
1     2
Name: FLAG_DOCUMENT_17, dtype: int64


for FLAG_DOCUMENT_18: defaulters count
TARGET
```

```
0    2358
1     142
Name: FLAG_DOCUMENT_18, dtype: int64


for FLAG_DOCUMENT_19: defaulters count
TARGET
0    171
1     12
Name: FLAG_DOCUMENT_19, dtype: int64


for FLAG_DOCUMENT_20: defaulters count
TARGET
0    143
1     13
Name: FLAG_DOCUMENT_20, dtype: int64


for FLAG_DOCUMENT_21: defaulters count
TARGET
0    89
1    14
Name: FLAG_DOCUMENT_21, dtype: int64
```

- The above data says

For any flag doument n:

0 --> For non defaulters clients How many clients have submitted the Flagged document n

1 --> For defaulters clients How many clients have submitted the Flagged document n

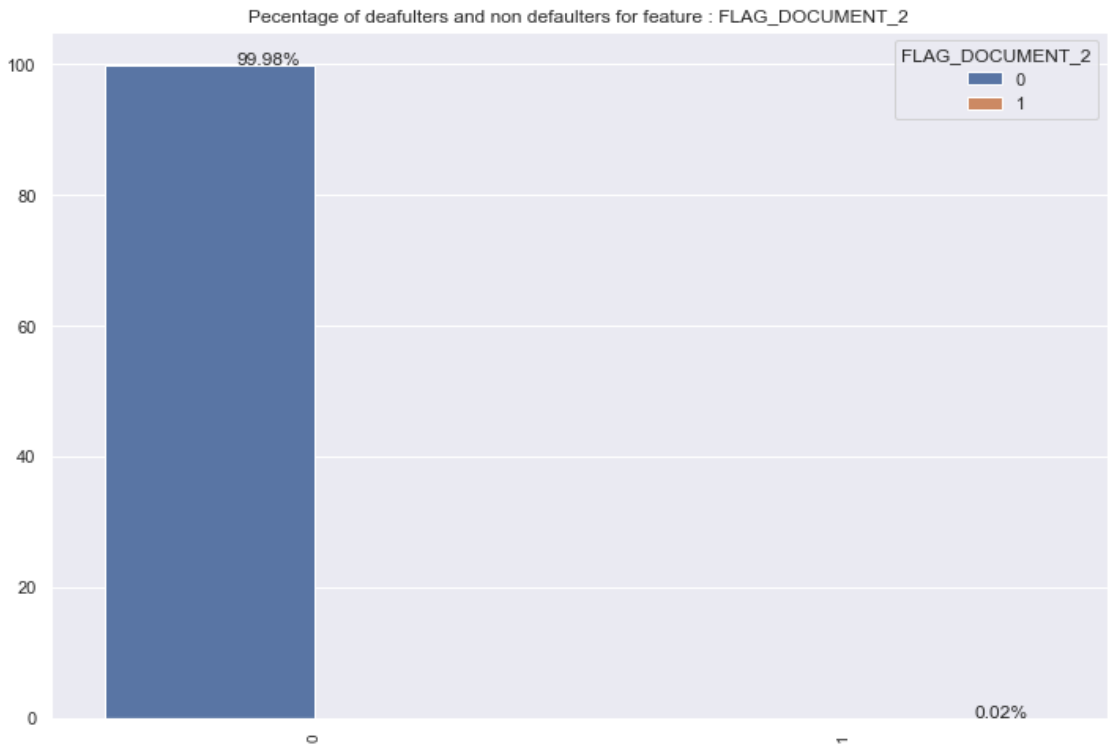
We see these are the flag documents have some good variation between defaulters and Non -defaulters

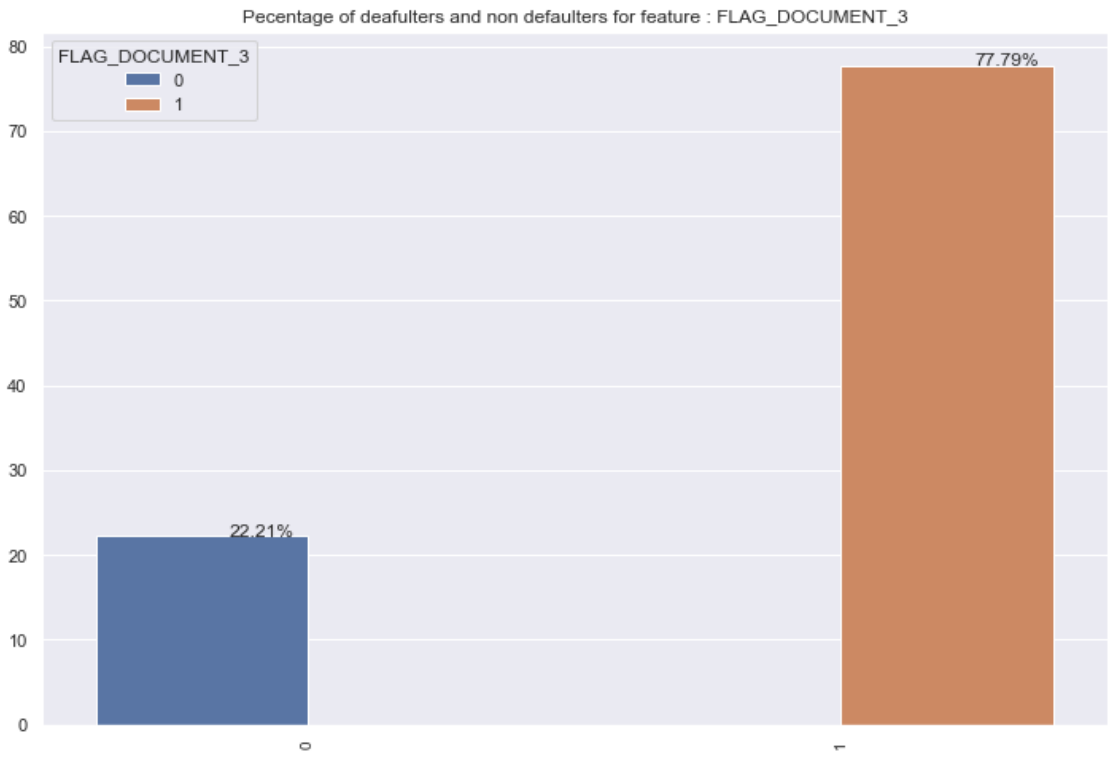
- FLAG_DOCUMENT_3
- FLAG_DOCUMENT_5
- FLAG_DOCUMENT_6
- FLAG_DOCUMENT_8
- FLAG_DOCUMENT_9
- FLAG_DOCUMENT_11
- FLAG_DOCUMENT_16

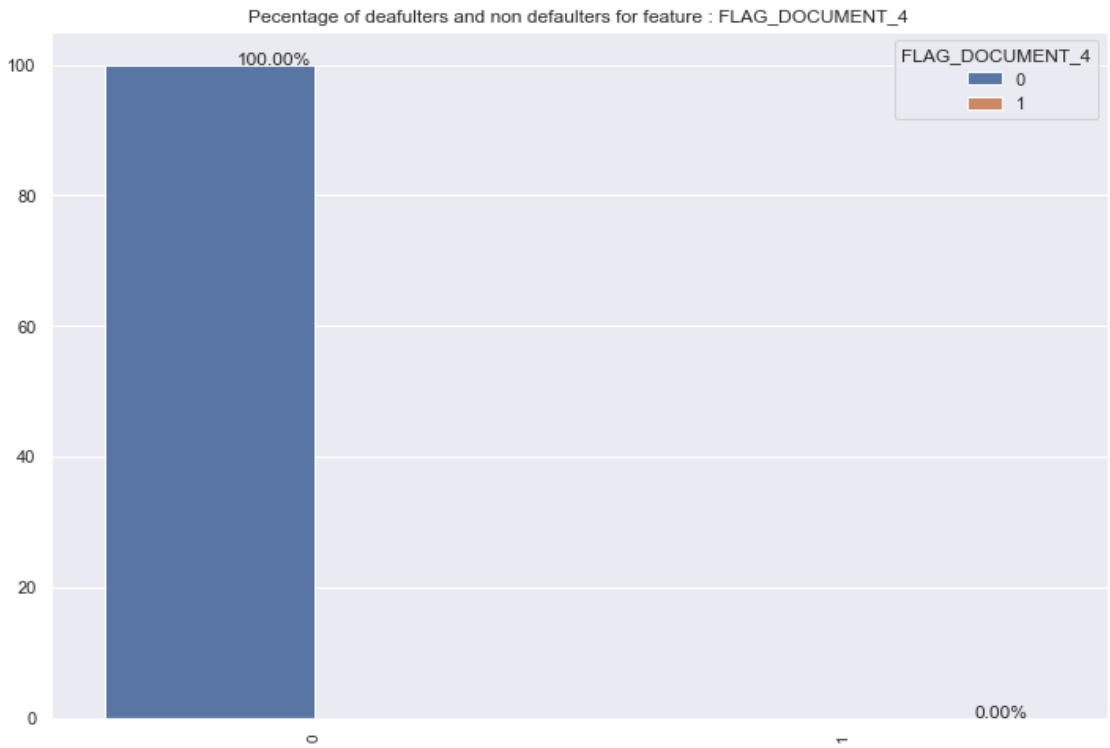
- FLAG_DOCUMENT_18

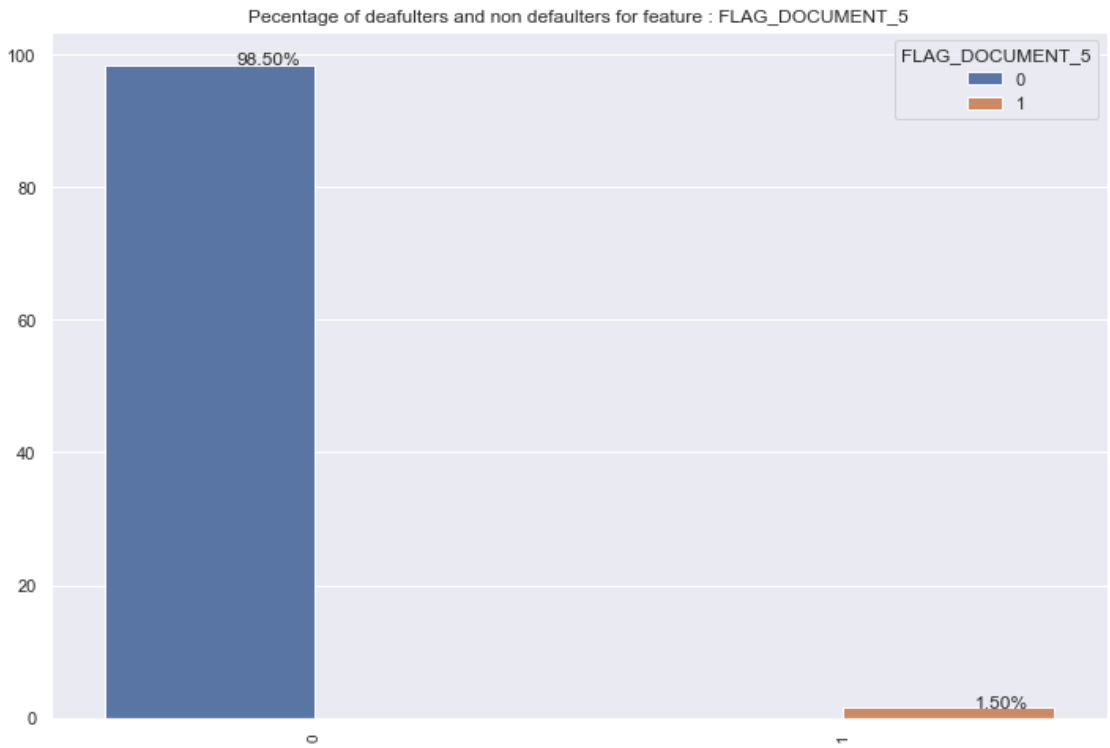
We will consider those and leave rest

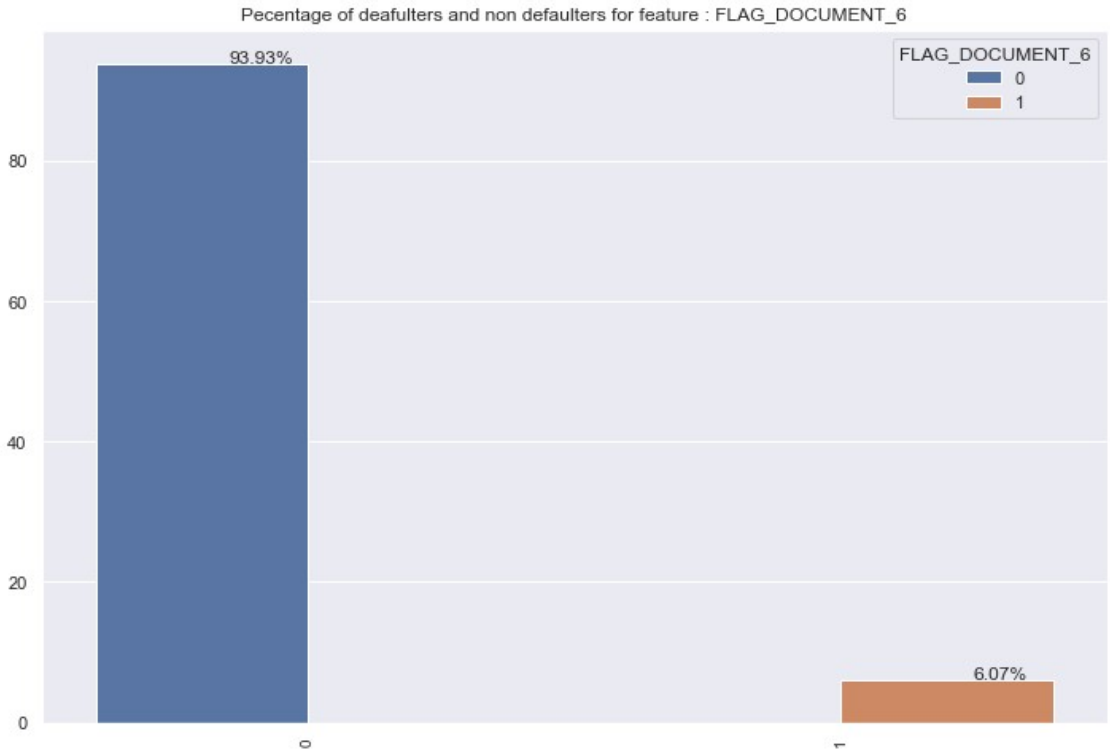
```
for i in flag_doc_cols:
    plot_for_categorical_defaulters(df_train,i)
    print("\n \n \n")
```

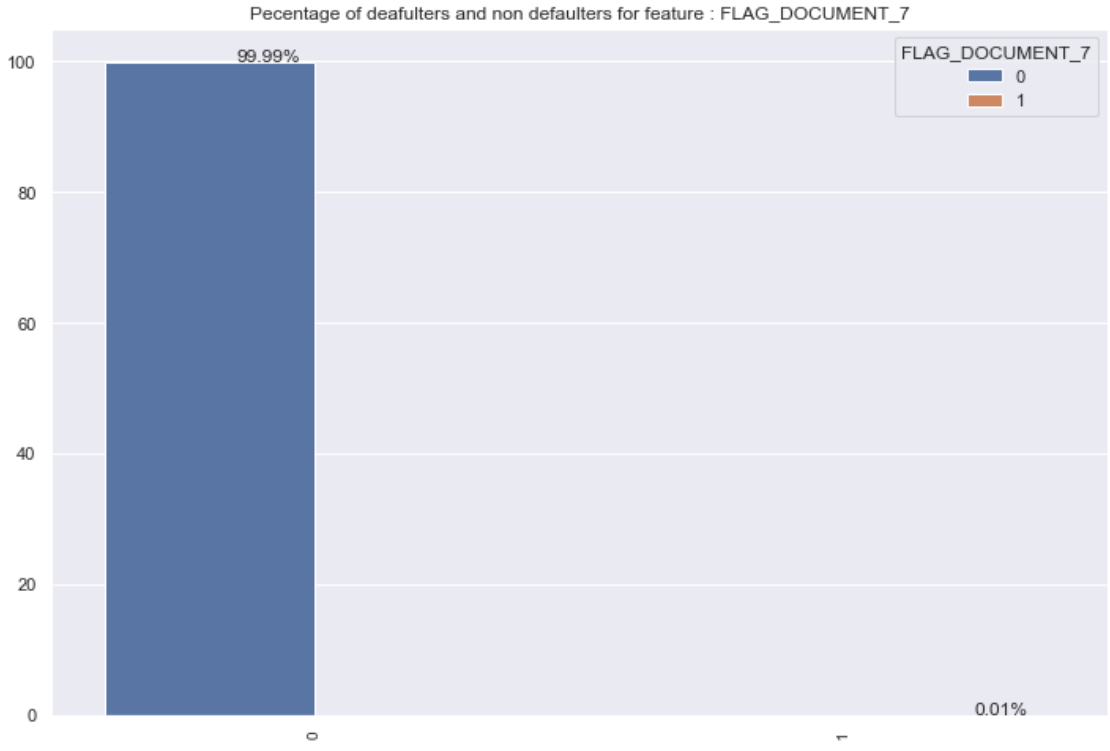

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_2
FLAG_DOCUMENT_2
0
1
99.98%
0.02%
100
80
60
40
20
0
0
1

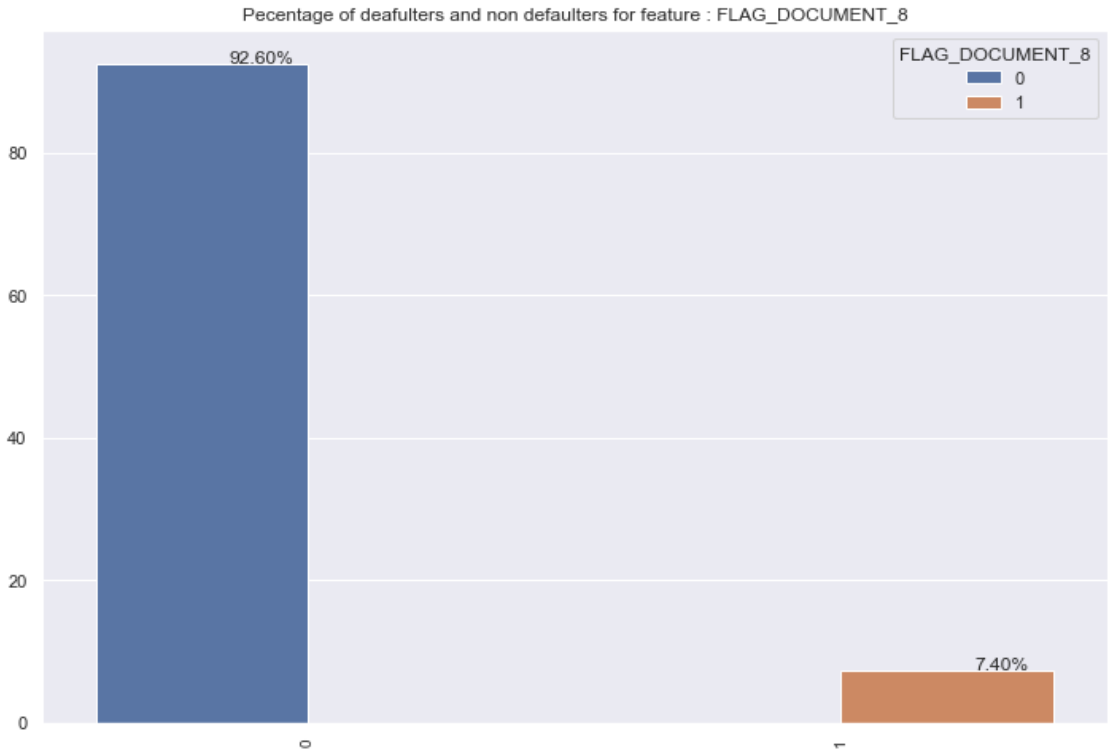

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_3
FLAG_DOCUMENT_3
0
1
22.21%
77.79%
80
70
60
50
40
30
20
10
0
0
1

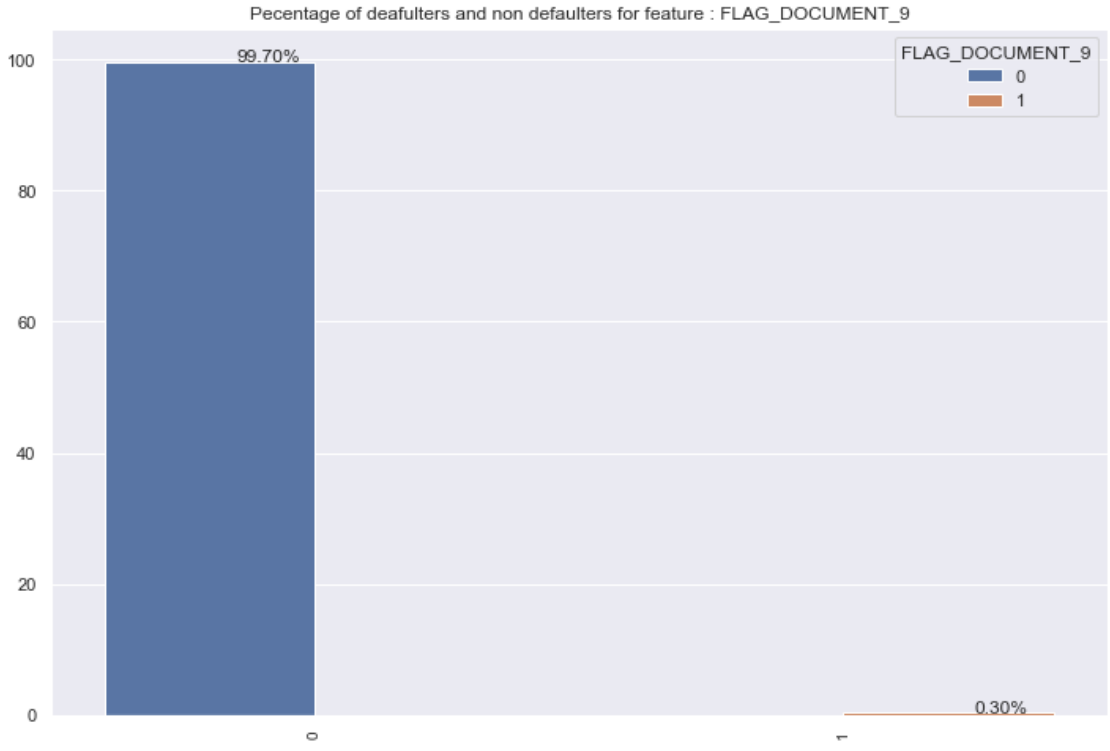

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_4
FLAG_DOCUMENT_4
0
1
100.00%
0.00%
100
80
60
40
20
0
0
1

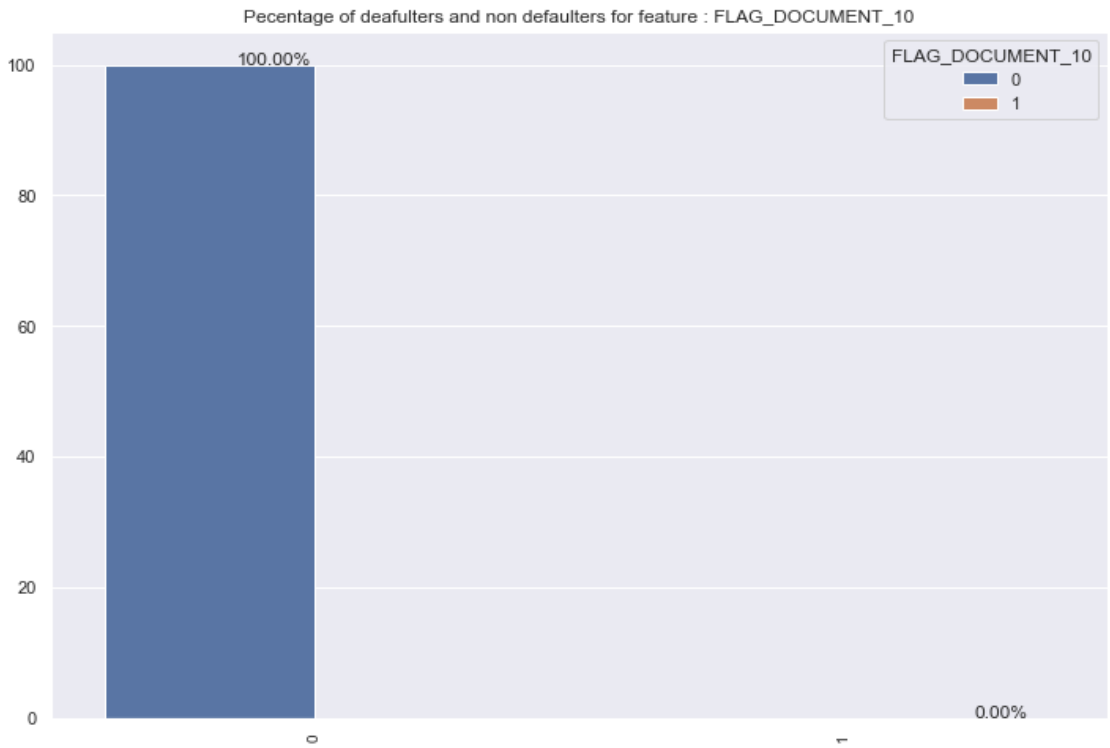

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_5
FLAG_DOCUMENT_5
0
1
98.50%
1.50%
100
80
60
40
20
0
0
1

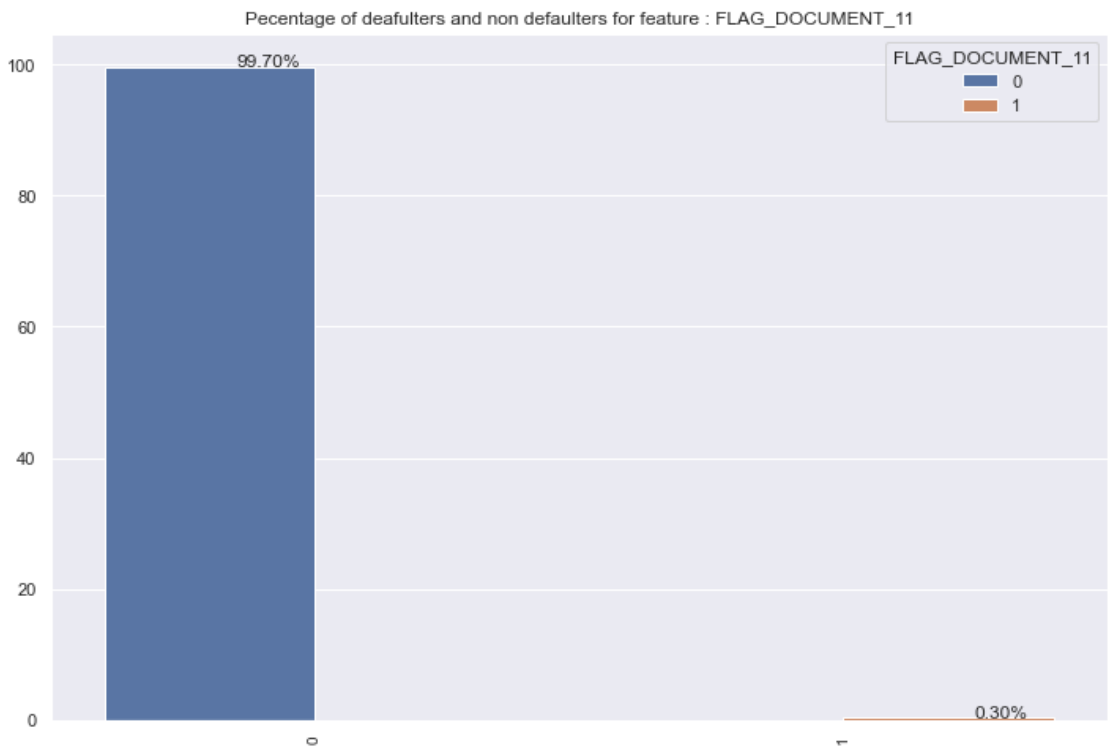

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_6
FLAG_DOCUMENT_6
0
1
93.93%
6.07%
0
20
40
60
80
0
1

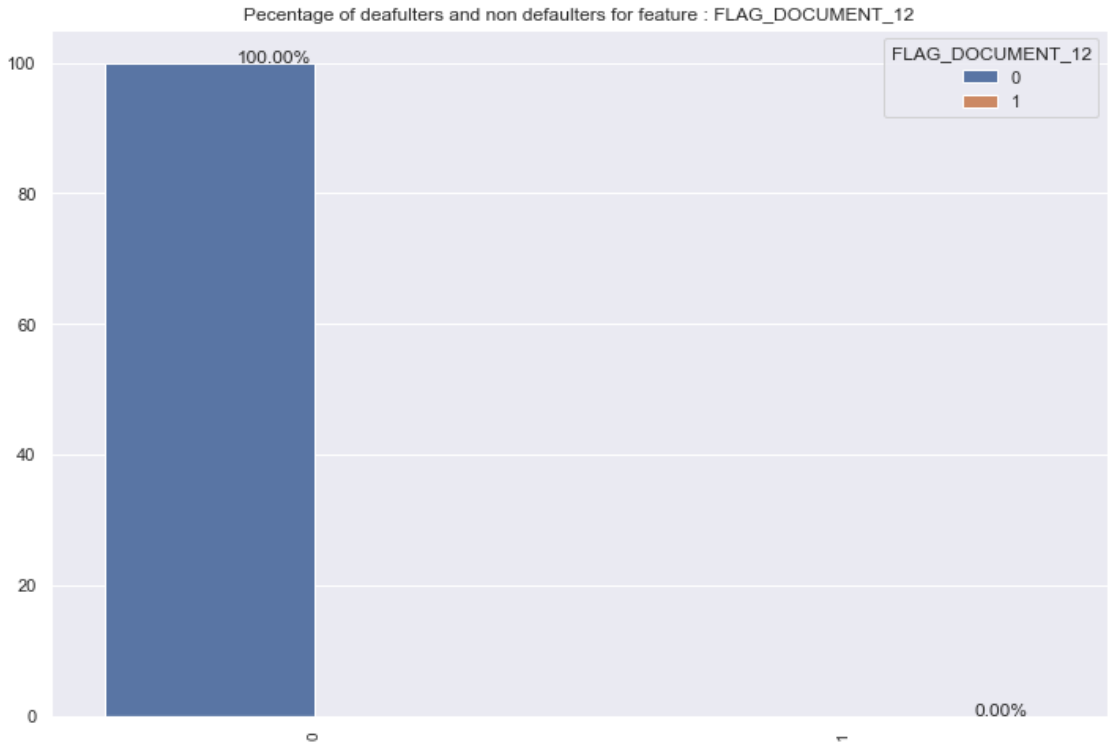

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_7
FLAG_DOCUMENT_7
0
1
99.99%
0.01%
0
20
40
60
80
100
0
1

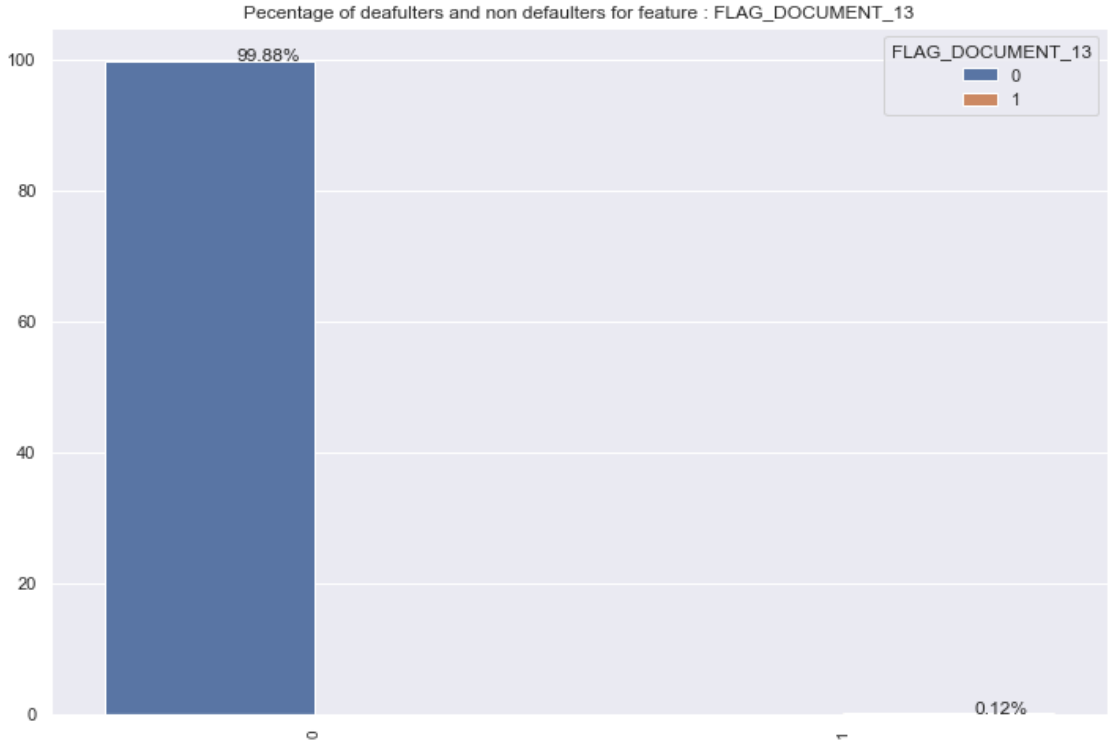

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_8
FLAG_DOCUMENT_8
0
1
92.60%
7.40%
0
20
40
60
80
0
1

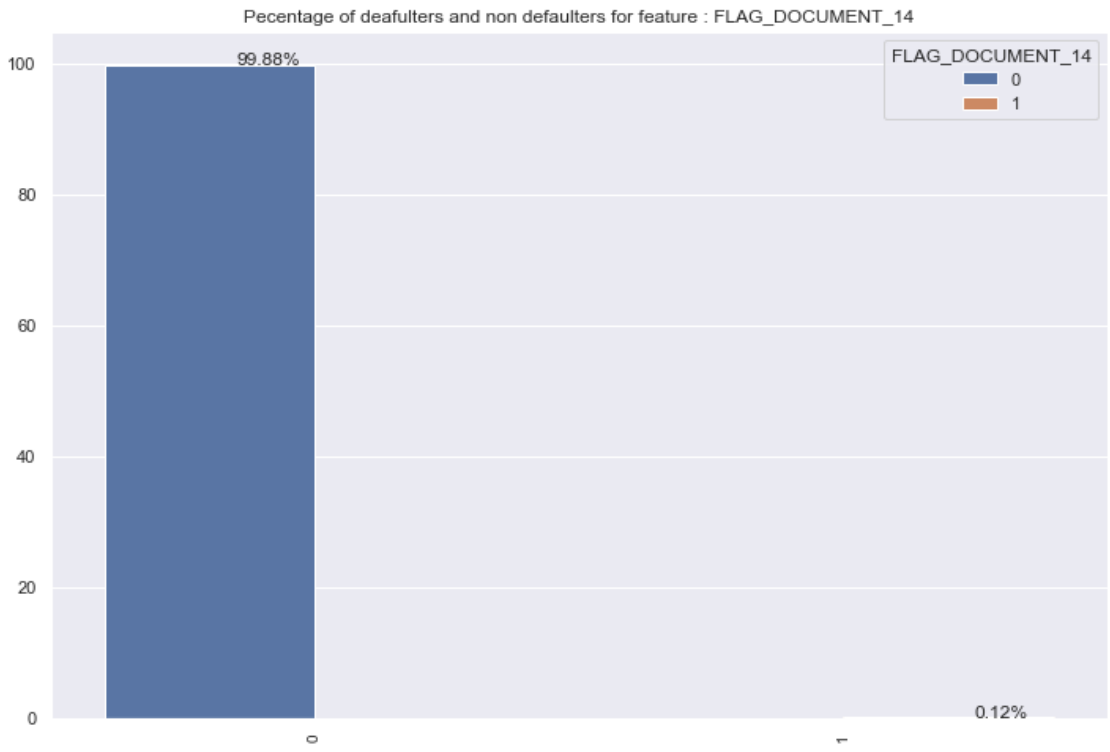

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_9
FLAG_DOCUMENT_9
0
1
99.70%
0.30%
0
20
40
60
80
100
0
1

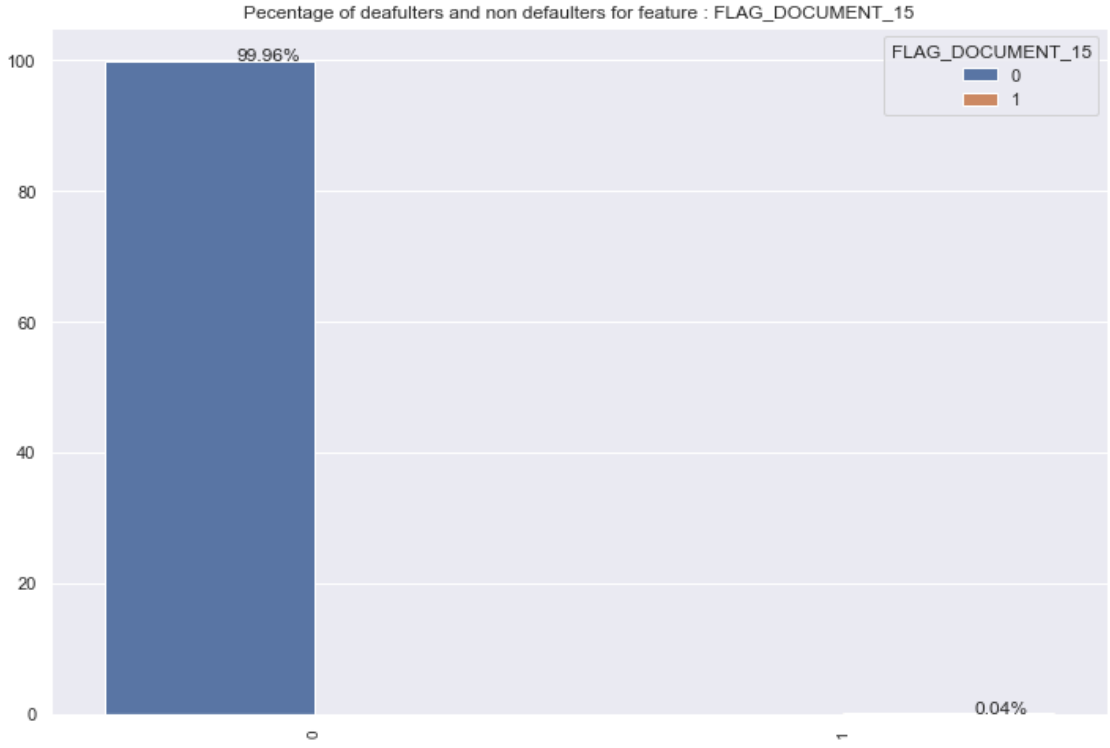

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_10
FLAG_DOCUMENT_10
0
1
100.00%
0.00%
100
80
60
40
20
0
0
1

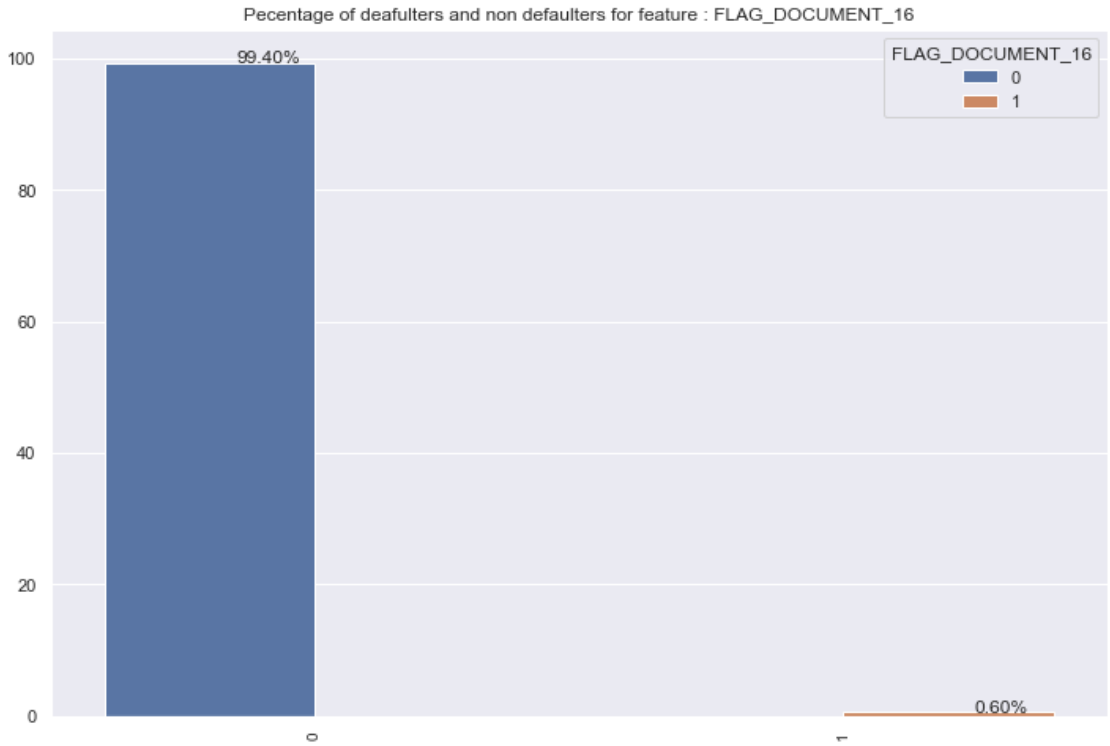

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_11
FLAG_DOCUMENT_11
0
1
99.70%
0.30%
100
80
60
40
20
0
0
1

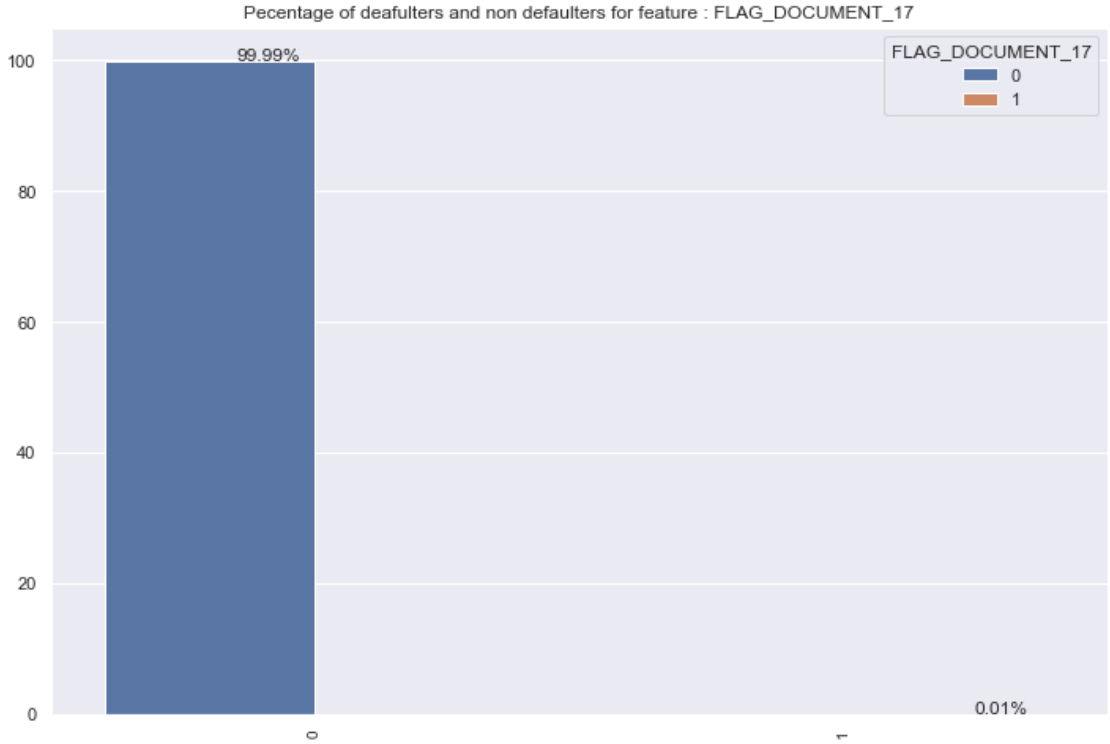

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_12
FLAG_DOCUMENT_12
0
1
100.00%
0.00%
100
80
60
40
20
0
0
1

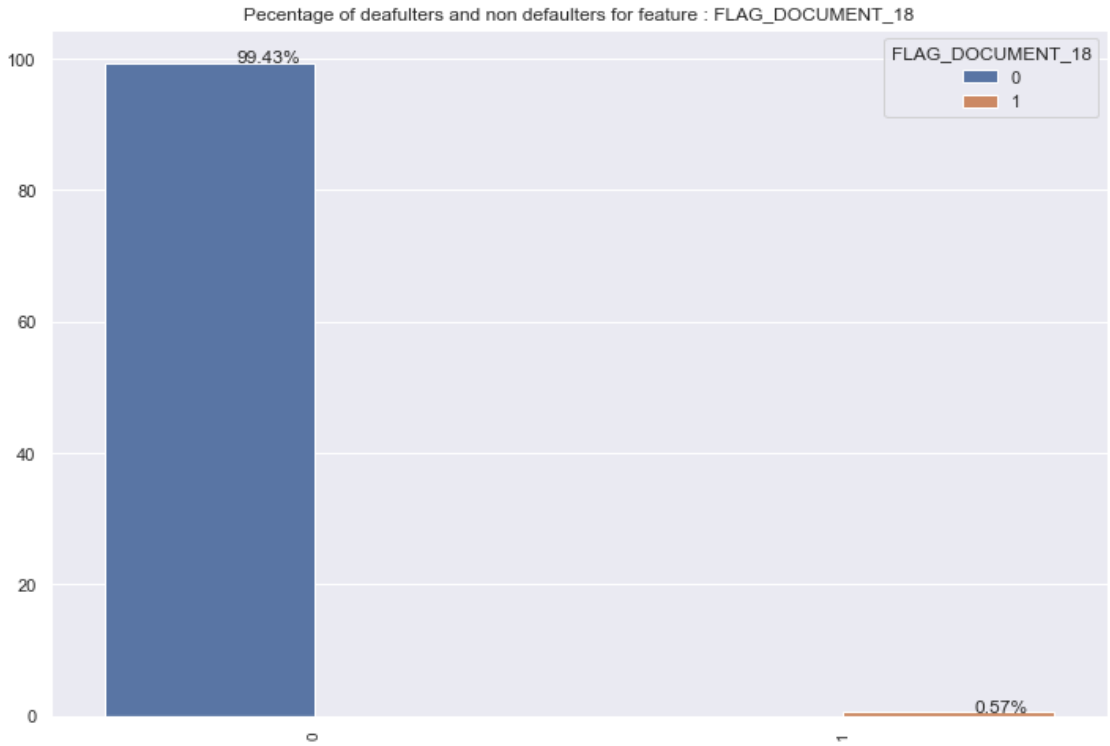

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_13
FLAG_DOCUMENT_13
0
1
99.88%
0.12%
100
80
60
40
20
0
0
1

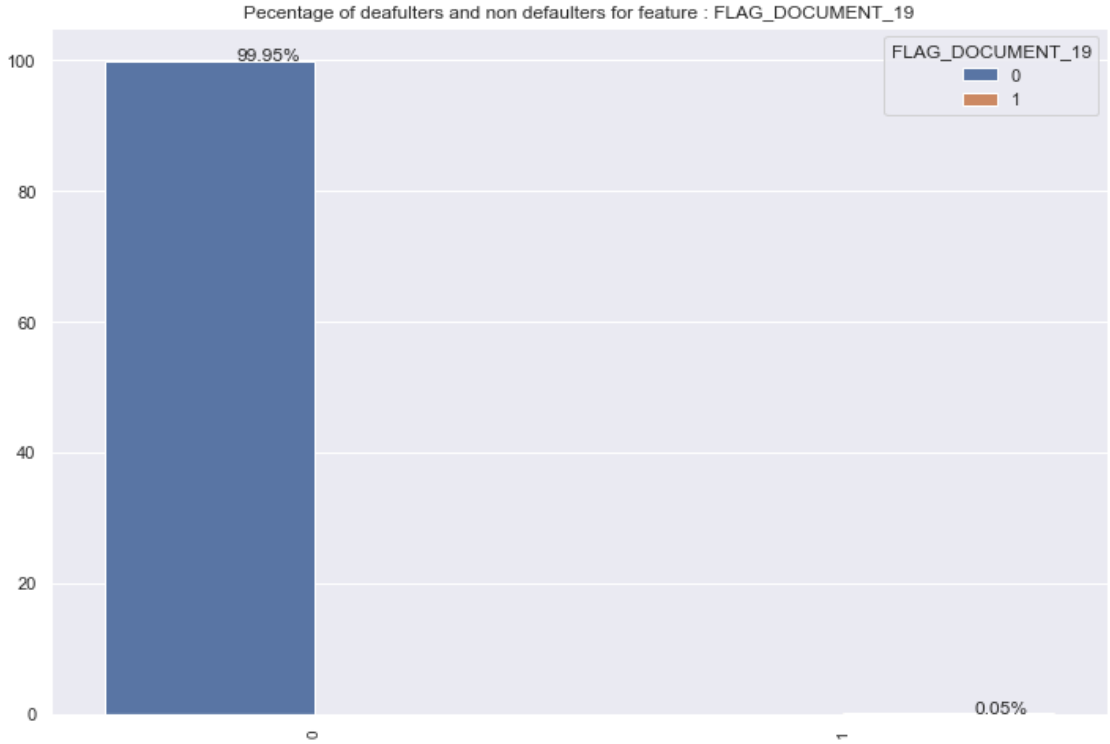

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_14
FLAG_DOCUMENT_14
0
1
99.88%
0.12%
100
80
60
40
20
0
0
1

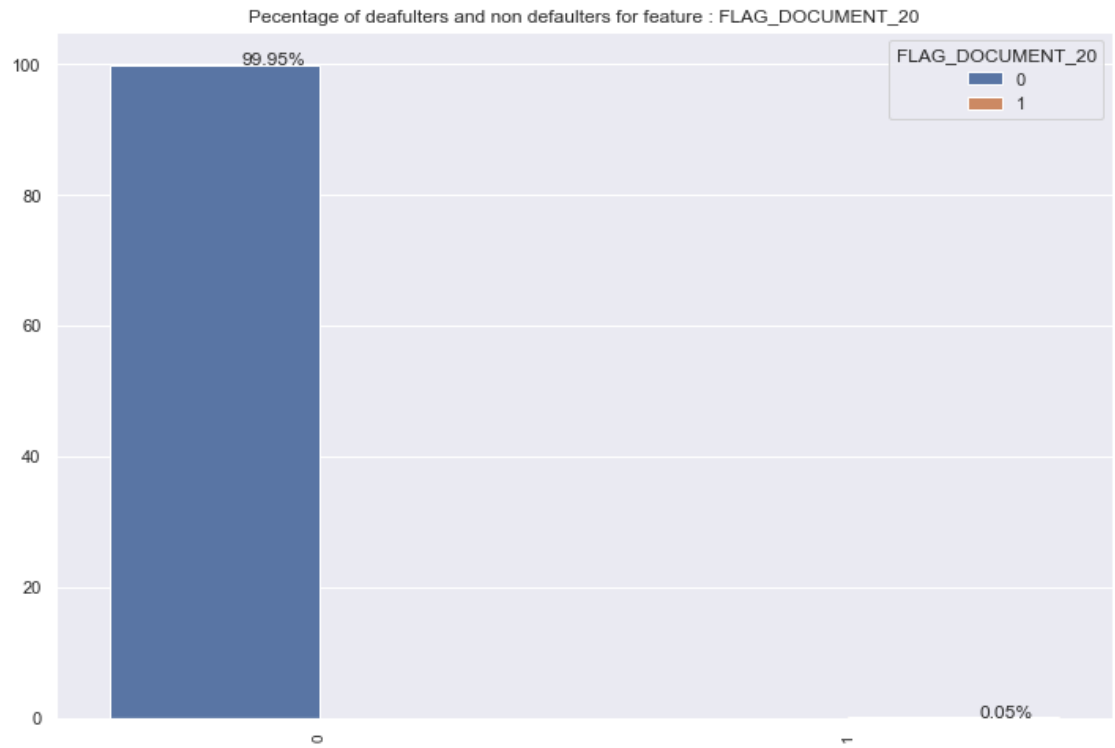

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_15
FLAG_DOCUMENT_15
0
1
99.96%
0.04%
100
80
60
40
20
0
0
1

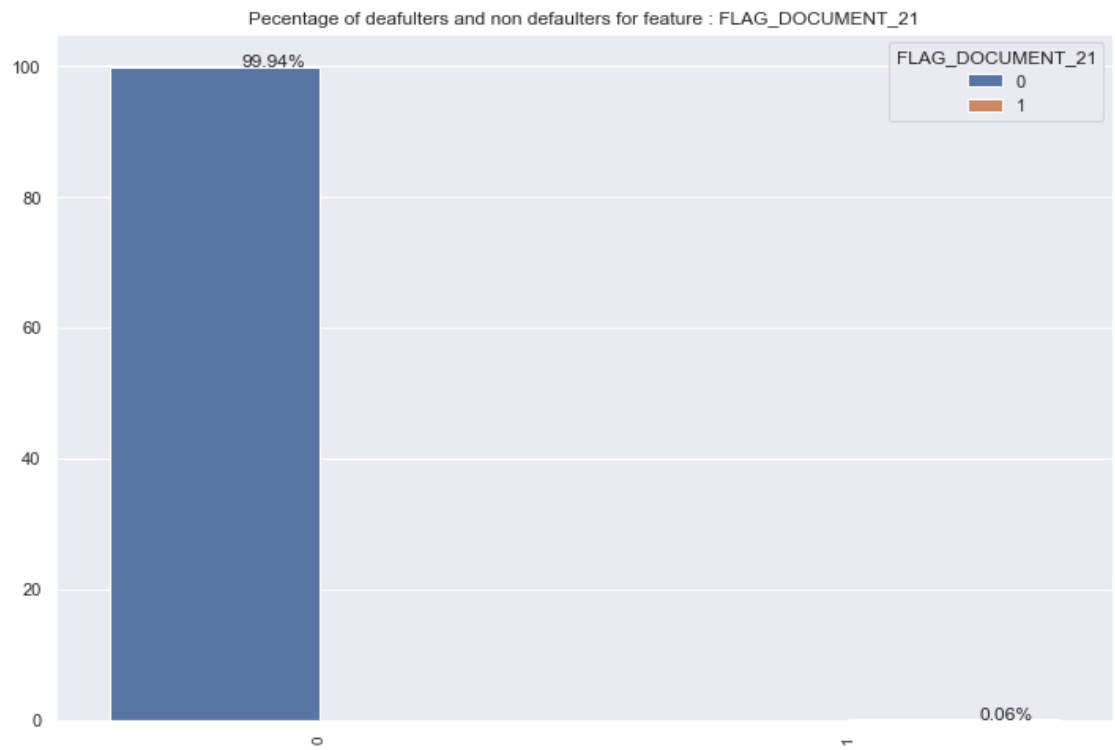

Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_16
FLAG_DOCUMENT_16
0
1
99.40%
0.60%
100
80
60
40
20
0
0
1


Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_17
FLAG_DOCUMENT_17
0
1
99.99%
0.01%
100
80
60
40
20
0
0
1


Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_18
FLAG_DOCUMENT_18
0
1
99.43%
0.57%
100
80
60
40
20
0
0
1


Pecentage of deafulters and non defaulters for feature : FLAG_DOCUMENT_19
FLAG_DOCUMENT_19
0
1
99.95%
0.05%
100
80
60
40
20
0
0
1





- we can see with help of above images , where there is no very big gap between both value of flag document, that flag will have more variations of both Target categories

```
df_train[flag_doc_cols].std().sort_values(ascending=False)
```

```
FLAG_DOCUMENT_3    0.453752
FLAG_DOCUMENT_6    0.283376
FLAG_DOCUMENT_8    0.273412
FLAG_DOCUMENT_5    0.122010
FLAG_DOCUMENT_16   0.099144
FLAG_DOCUMENT_18   0.089798
FLAG_DOCUMENT_11   0.062424
FLAG_DOCUMENT_9    0.062295
FLAG_DOCUMENT_13   0.059268
FLAG_DOCUMENT_14   0.054110
FLAG_DOCUMENT_15   0.034760
FLAG_DOCUMENT_19   0.024387
FLAG_DOCUMENT_20   0.022518
FLAG_DOCUMENT_21   0.018299
FLAG_DOCUMENT_17   0.016327
FLAG_DOCUMENT_7    0.013850
FLAG_DOCUMENT_4    0.009016
FLAG_DOCUMENT_2    0.006502
FLAG_DOCUMENT_10   0.004771
FLAG_DOCUMENT_12   0.002550
dtype: float64
```

We will look down to these columns now

- 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

```
AMT_REQ_CREDIT_cols= [col for col in df_train.columns if
'AMT_REQ_CREDIT' in col]
```

```
AMT_REQ_CREDIT_cols
```

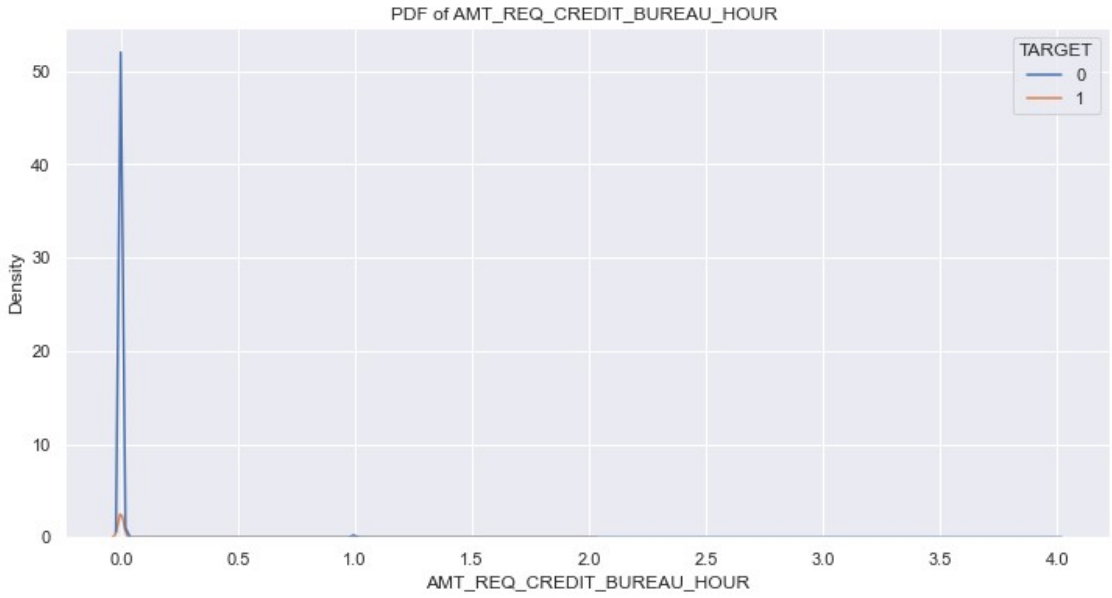
```
['AMT_REQ_CREDIT_BUREAU_HOUR',
 'AMT_REQ_CREDIT_BUREAU_DAY',
 'AMT_REQ_CREDIT_BUREAU_WEEK',
 'AMT_REQ_CREDIT_BUREAU_MON',
 'AMT_REQ_CREDIT_BUREAU_QRT',
 'AMT_REQ_CREDIT_BUREAU_YEAR']
```

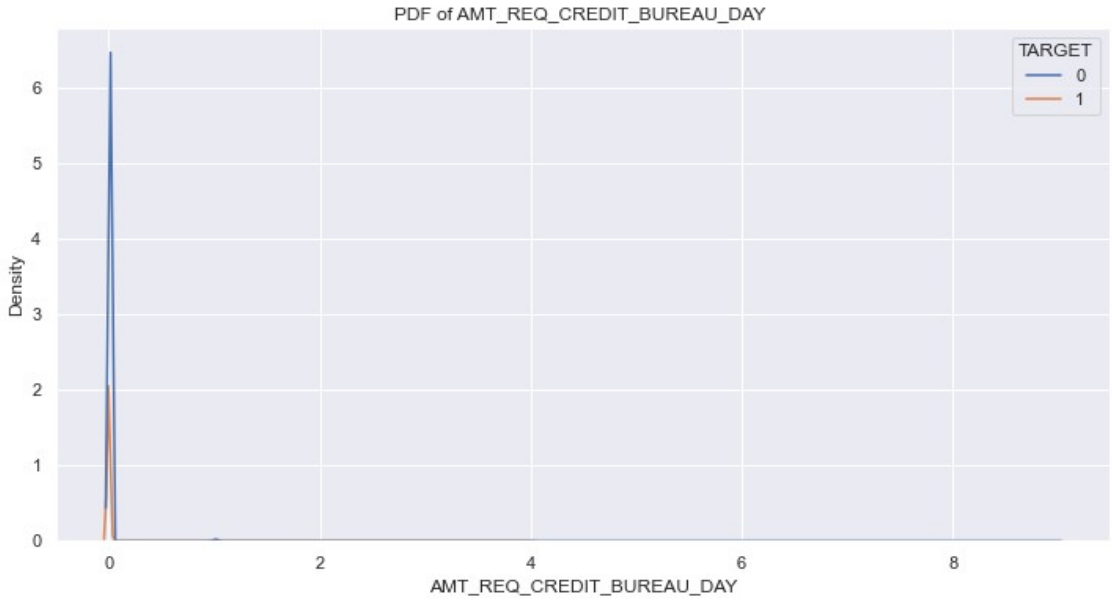
```
df_train[AMT_REQ_CREDIT_cols].std().sort_values(ascending=False)
```

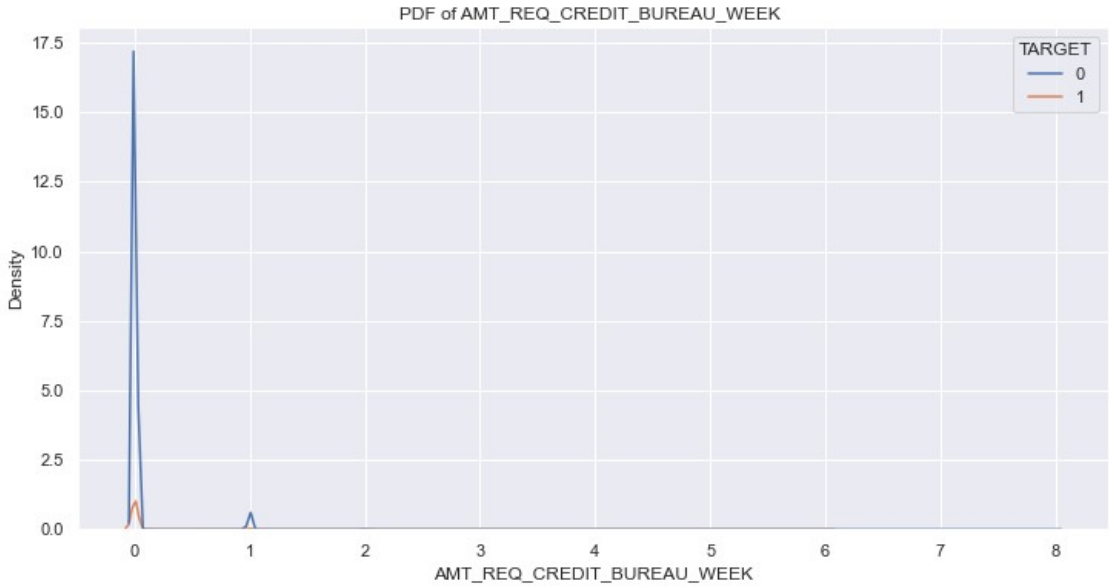
```
AMT_REQ_CREDIT_BUREAU_YEAR   1.869295
AMT_REQ_CREDIT_BUREAU_MON    0.916002
AMT_REQ_CREDIT_BUREAU_QRT    0.794056
AMT_REQ_CREDIT_BUREAU_WEEK   0.204685
AMT_REQ_CREDIT_BUREAU_DAY    0.110757
```

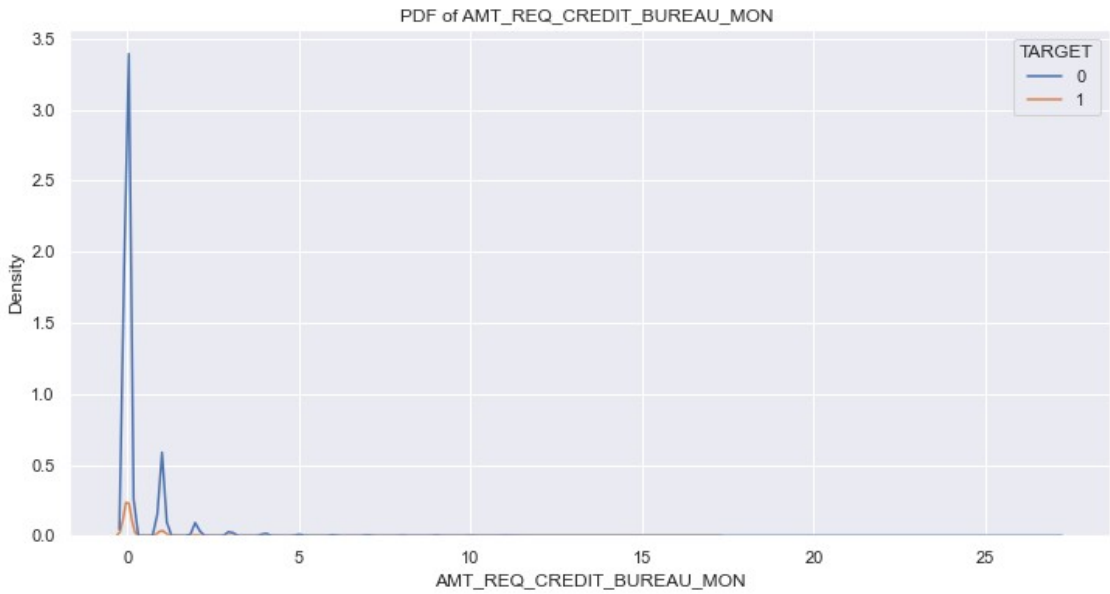
```
AMT_REQ_CREDIT_BUREAU_HOUR   0.083849
dtype: float64
```

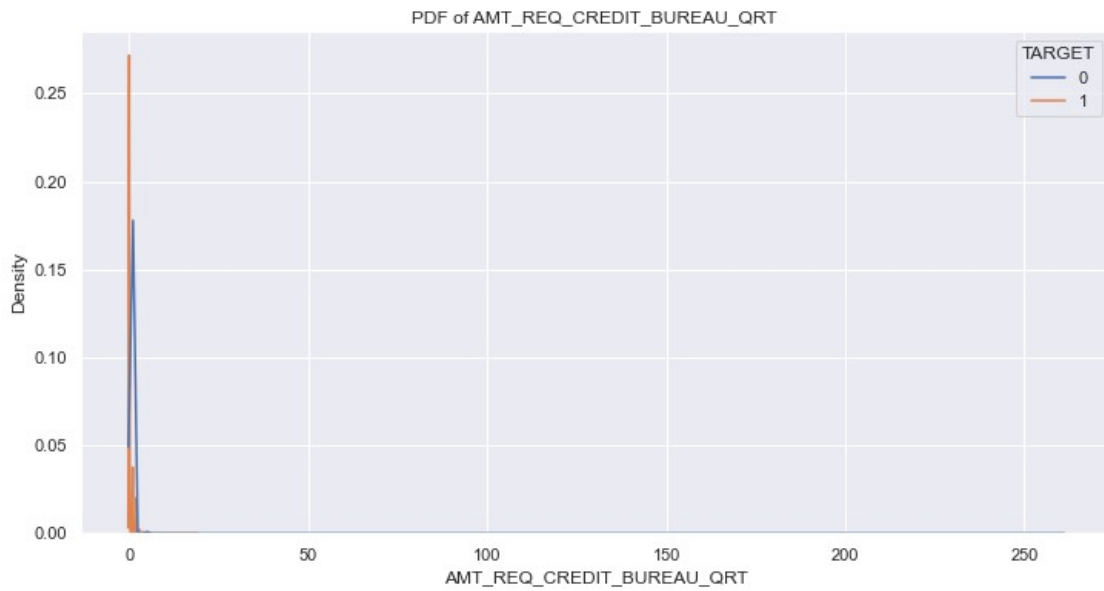
```
for i in range(len(AMT_REQ_CREDIT_cols)):
#sns.set(style="darkgrid")
    plt.figure(figsize=(12,6))

sns.kdeplot(df_train[AMT_REQ_CREDIT_cols[i]],label=AMT_REQ_CREDIT_cols
[i],hue=df_train['TARGET'])
    #plt.xlim([0,2])
    plt.title(f"PDF of {AMT_REQ_CREDIT_cols[i]} ")
    #plt.legend()
    print("\n \n ")
```

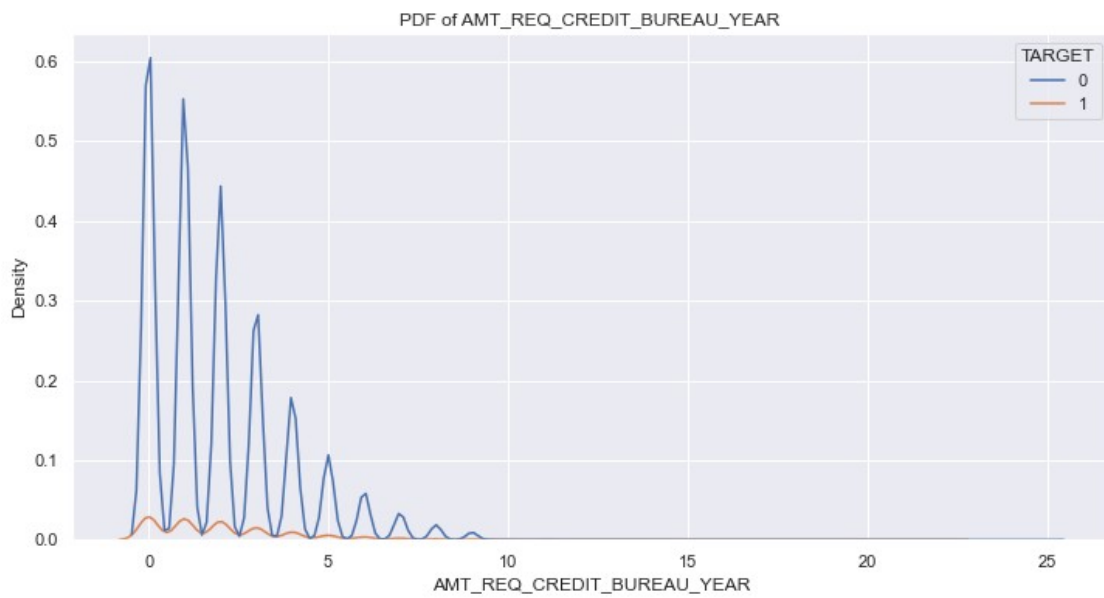

PDF of AMT_REQ_CREDIT_BUREAU_HOUR
TARGET
0
1
Density
0
10
20
30
40
50
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
AMT_REQ_CREDIT_BUREAU_HOUR


PDF of AMT_REQ_CREDIT_BUREAU_DAY
TARGET
0
1
Density
0
1
2
3
4
5
6
0
2
4
6
8
AMT_REQ_CREDIT_BUREAU_DAY


PDF of AMT_REQ_CREDIT_BUREAU_WEEK
TARGET
0
1
Density
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
2
3
4
5
6
7
8
AMT_REQ_CREDIT_BUREAU_WEEK


PDF of AMT_REQ_CREDIT_BUREAU_MON
TARGET
0
1
Density
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
5
10
15
20
25
AMT_REQ_CREDIT_BUREAU_MON





I am not sure about these columns here we can see those later

```
#Now we will try to see and encode the categorical features now,let
see the categoricla columns now
```

```
categorical_features
```

```
['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']



from sklearn.preprocessing import LabelEncoder
def encode_fillna_categorical_columns(df):
    #NAME_CONTRACT_TYPE
    NAME_CONTRACT_TYPE_encode= {'Cash loans': 1, 'Revolving loans': 0}
    df['NAME_CONTRACT_TYPE']=
df['NAME_CONTRACT_TYPE'].map(NAME_CONTRACT_TYPE_encode)
    #CODE_GENDER
    df['CODE_GENDER']= df['CODE_GENDER'].map({'F':1,'M':2,'XNA':0})
    #FLAG_OWN_CAR
    df['FLAG_OWN_CAR']= df['FLAG_OWN_CAR'].map({'N':1,'Y':0})
    #FLAG_OWN_REALTY
    df['FLAG_OWN_REALTY']= df['FLAG_OWN_REALTY'].map({'N':0,'Y':1})
    #NAME_TYPE_SUITE
    df['NAME_TYPE_SUITE'].fillna('Unaccompanied',inplace=True)
    le_NAME_TYPE_SUITE=LabelEncoder()
    le_NAME_TYPE_SUITE.fit(df_train['NAME_TYPE_SUITE'])

df['NAME_TYPE_SUITE']=le_NAME_TYPE_SUITE.transform(df['NAME_TYPE_SUITE
'])
    #NAME_INCOME_TYPE
    le_NAME_INCOME_TYPE=LabelEncoder()
    le_NAME_INCOME_TYPE.fit(df_train['NAME_INCOME_TYPE'])

df['NAME_INCOME_TYPE']=le_NAME_INCOME_TYPE.transform(df['NAME_INCOME_T
YPE'])
    #NAME_EDUCATION_TYPE
    map_NAME_EDUCATION_TYPE={'Secondary / secondary special': 4,
                             'Higher education': 2,
                             'Incomplete higher': 3,
                             'Lower secondary': 5,
                             'Academic degree': 1}

    df['NAME_EDUCATION_TYPE']=
df['NAME_EDUCATION_TYPE'].map(map_NAME_EDUCATION_TYPE)
    #NAME_FAMILY_STATUS
    map_NAME_FAMILY_STATUS= {'Married': 5,
                             'Single / not married': 2,
                             'Civil marriage': 3,
```

```
                                'Separated': 2,
                                'Widow': 1,
                                'Unknown': 0}
    df['NAME_FAMILY_STATUS']=
df['NAME_FAMILY_STATUS'].map(map_NAME_FAMILY_STATUS)


    #NAME_HOUSING_TYPE
    map_NAME_HOUSING_TYPE={'House / apartment': 5,
                        'With parents': 4,
                        'Municipal apartment': 2,
                        'Rented apartment': 3, 'Office apartment': 1,
                        'Co-op apartment': 1}

    df['NAME_HOUSING_TYPE']=
df['NAME_HOUSING_TYPE'].map(map_NAME_HOUSING_TYPE)
    #OCCUPATION_TYPE
    map_OCCUPATION_TYPE= {'IT staff': 0,
                          'HR staff': 0,
                          'Realty agents': 0,
                          'Secretaries': 0,
                          'Waiters/barmen staff': 0,
                          'Private service staff': 0,
                          'Low-skill Laborers': 1,
                          'Cleaning staff': 1,
                          'Accountants': 1,
                          'Medicine staff': 2,
                          'Cooking staff': 2,
                          'High skill tech staff': 2,
                          'Security staff': 2,
                          'Managers': 3,
                          'Core staff': 3,
                          'Drivers': 3,
                          'Sales staff': 4,
                          'Laborers': 5}


    df['OCCUPATION_TYPE']=
df['OCCUPATION_TYPE'].map(map_OCCUPATION_TYPE)#
    #WEEKDAY_APPR_PROCESS_START: we will just drop this column because
its no use
    df.drop('WEEKDAY_APPR_PROCESS_START',axis=1)
    #ORGANIZATION_TYPE,we can catregorize in it in 3
    df['OCCUPATION_TYPE']=
df['ORGANIZATION_TYPE'].map(map_org_type_encode)
    #FONDKAPREMONT_MODE
    df['FONDKAPREMONT_MODE'].fillna('Not specified',inplace=True)
    #we can label encode it
    le_FONDKAPREMONT_MODE=LabelEncoder()
    le_FONDKAPREMONT_MODE.fit(df_train['FONDKAPREMONT_MODE'])
```

```
df['FONDKAPREMONT_MODE']=le_FONDKAPREMONT_MODE.transform(df['FONDKAPRE
MONT_MODE'])
    #HOUSETYPE_MODE
    map_HOUSETYPE_MODE={'Not specified':2,'block of flats':1,'specific
housing': 0,'terraced house': 0}
    df['HOUSETYPE_MODE']= df['HOUSETYPE_MODE'].map(map_HOUSETYPE_MODE)
    #WALLSMATERIAL_MODE
    df['WALLSMATERIAL_MODE']=df['WALLSMATERIAL_MODE'].fillna("Not
Specified")

df['WALLSMATERIAL_MODE']=df['WALLSMATERIAL_MODE'].map(map_wallsmateria
l_type)
    #EMERGENCYSTATE_MODE
    df['EMERGENCYSTATE_MODE']=df['EMERGENCYSTATE_MODE'].fillna("No")

df['EMERGENCYSTATE_MODE']=df['EMERGENCYSTATE_MODE'].map({'Yes':1,'No':
0})
    return df
    print("done")

def fillna_numerical_columns(df):
    #AMT_GOODS_PRICE : fill it with same value of AMT_credit

df['AMT_GOODS_PRICE']=df['AMT_GOODS_PRICE'].fillna(df['AMT_CREDIT'])
    #OWN_CAR_AGE,fillna with zero because they dont have car
    df['OWN_CAR_AGE']= df['OWN_CAR_AGE'].fillna(0)
    #DAYS_LAST_PHONE_CHANGE

df['DAYS_LAST_PHONE_CHANGE']=df['DAYS_LAST_PHONE_CHANGE'].fillna(0)
    #CNT_FAM_MEMBERS
    df['CNT_FAM_MEMBERS']=df['CNT_FAM_MEMBERS'].fillna(0)
    #AMT_ANNUITY;we suppose that where AMT_ANNUITY is equal to
AMT_CREDIT without considering interest and repayment
    #period as 1 year
    df['AMT_ANNUITY']=df['AMT_ANNUITY'].fillna(df['AMT_CREDIT'])
    #fill na for 'EXT_SOURCE_1','EXT_SOURCE_2','EXT_SOURCE_3' with 0
i.e zero income from thate externel income source
    df['EXT_SOURCE_1']= df['EXT_SOURCE_1'].fillna(0)
    df['EXT_SOURCE_2']= df_train['EXT_SOURCE_2'].fillna(0)
    df['EXT_SOURCE_3']= df_train['EXT_SOURCE_3'].fillna(0)
    #TOTALAREA_MODE: filling it with  highest correlation colum
LIVINGAREA_AVG
    df['TOTALAREA_MODE']=
df['TOTALAREA_MODE'].fillna(df['LIVINGAREA_AVG'])
    #DAYS_LAST_PHONE_CHANGE
    df['DAYS_LAST_PHONE_CHANGE'].fillna(0,inplace=True)
    #I am not filling NA for the Building Area and other column
because , I think we will drop it as it dont show any use wrt Target
```

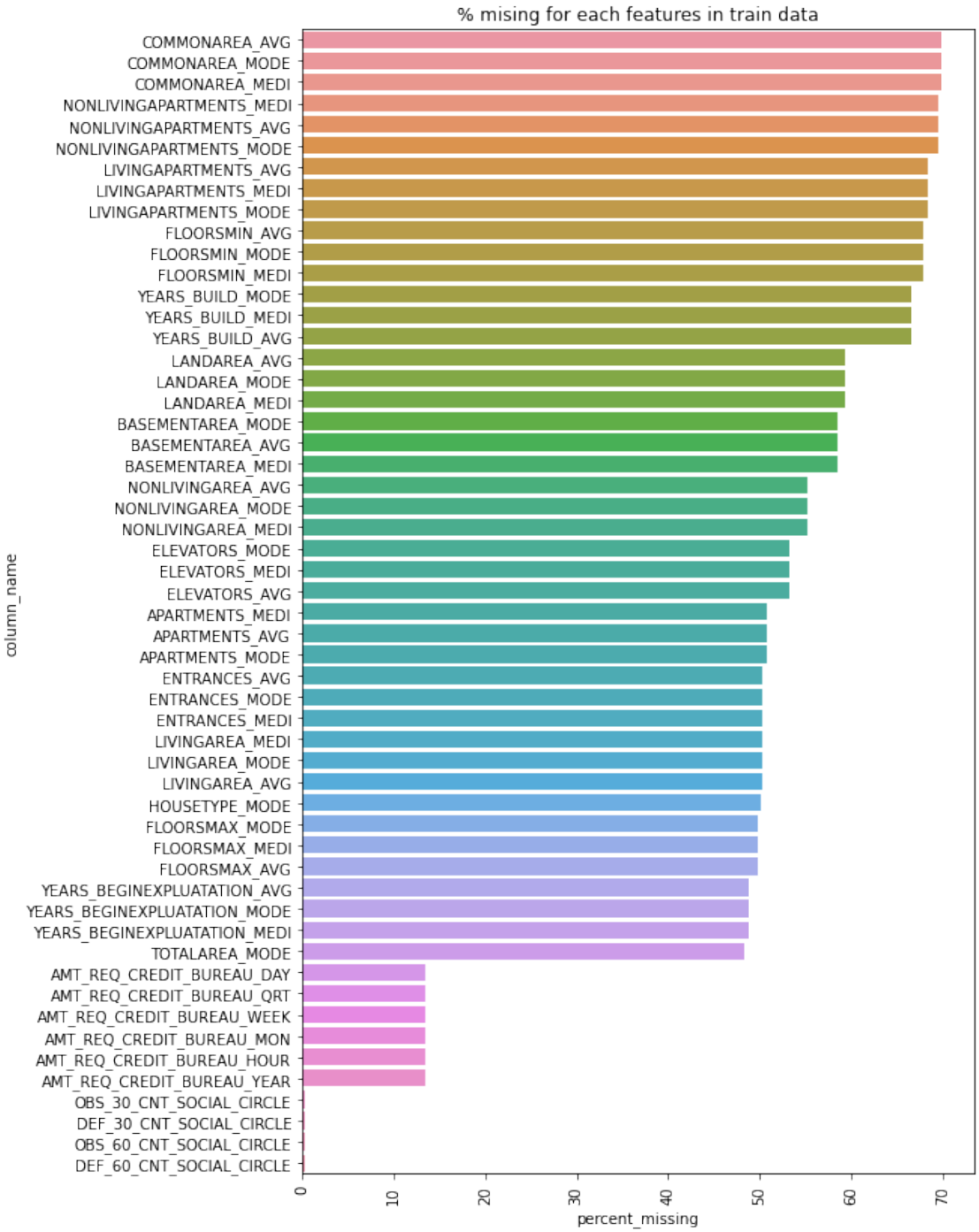
```
    return df
    print("done")

df_train=pd.read_csv(r'home-credit-default-risk/
application_train.csv')
df_train=encode_fillna_categorical_columns(df_train)
df_train=fillna_numerical_columns(df_train)



df_train.head()
```

```
   SK_ID_CURR  TARGET  NAME_CONTRACT_TYPE  CODE_GENDER
FLAG_OWN_CAR  \
0      100002       1                   1            2             1

1      100003       0                   1            1             1

2      100004       0                   0            2             0

3      100006       0                   1            1             1

4      100007       0                   1            2             1


   FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT
AMT_ANNUITY  \
0                1             0          202500.0    406597.5
24700.5
1                0             0          270000.0   1293502.5
35698.5
2                1             0           67500.0    135000.0
6750.0
3                1             0          135000.0    312682.5
29686.5
4                1             0          121500.0    513000.0
21865.5

   ...  FLAG_DOCUMENT_18  FLAG_DOCUMENT_19  FLAG_DOCUMENT_20  \
0  ...                 0                 0                 0
1  ...                 0                 0                 0
2  ...                 0                 0                 0
3  ...                 0                 0                 0
4  ...                 0                 0                 0

   FLAG_DOCUMENT_21  AMT_REQ_CREDIT_BUREAU_HOUR
AMT_REQ_CREDIT_BUREAU_DAY  \
0                 0                         0.0
0.0
1                 0                         0.0
```

```
0.0
2                  0                          0.0
0.0
3                  0                          NaN
NaN
4                  0                          0.0
0.0

   AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                         0.0                        0.0
1                         0.0                        0.0
2                         0.0                        0.0
3                         NaN                        NaN
4                         0.0                        0.0

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                         1.0
1                        0.0                         0.0
2                        0.0                         0.0
3                        NaN                         NaN
4                        0.0                         0.0

[5 rows x 122 columns]
```

```
percent_missing = df_train.isnull().sum() * 100 / len(df_train)

missing_value_df_train = pd.DataFrame({'column_name':
df_train.columns,
                                       'percent_missing':
percent_missing.values},index=None)

#missing_value_df_train

missing_value_df_train.sort_values('percent_missing',ascending=False,i
nplace=True)

missing_value_df_train=missing_value_df_train[missing_value_df_train['
percent_missing']>0]

missing_value_df_train.reset_index(inplace=True,drop=True)

#Lets plot it
plt.figure(figsize=(8,14))
g=sns.barplot(y='column_name',x='percent_missing',data=missing_value_d
f_train)
plt.xticks(rotation=90)
plt.title("% mising for each features in train data")
```

```
Text(0.5, 1.0, '% mising for each features in train data')
```



So now as we have done all detailed EDA about application data,and important imputation,Lets see another data as well

```
os.listdir('home-credit-default-risk/')
```

```
['application_test.csv',
 'application_train.csv',
 'bureau.csv',
 'bureau_balance.csv',
```

```
 'credit_card_balance.csv',
 'HomeCredit_columns_description.csv',
 'installments_payments.csv',
 'POS_CASH_balance.csv',
 'previous_application.csv',
 'sample_submission.csv']
```

```
bureau= pd.read_csv('home-credit-default-risk/bureau.csv')
bureau_balance=
pd.read_csv('home-credit-default-risk/bureau_balance.csv')
credit_card_balance=
pd.read_csv('home-credit-default-risk/credit_card_balance.csv')
installments_payments=
pd.read_csv('home-credit-default-risk/installments_payments.csv')
POS_CASH_balance=
pd.read_csv('home-credit-default-risk/POS_CASH_balance.csv')
previous_application=
pd.read_csv('home-credit-default-risk/previous_application.csv')
```

bureau.csv

This table consist the Credit Bureau records for the customers ,intimated by the CREDIT BUREAU

```
#Shape of Table
bureau.shape
```

```
(1716428, 17)
```

```
bureau.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1716428 entries, 0 to 1716427
Data columns (total 17 columns):
 #   Column                  Dtype
---  ------                  -----
 0   SK_ID_CURR              int64
 1   SK_ID_BUREAU            int64
 2   CREDIT_ACTIVE           object
 3   CREDIT_CURRENCY         object
 4   DAYS_CREDIT             int64
 5   CREDIT_DAY_OVERDUE      int64
 6   DAYS_CREDIT_ENDDATE     float64
 7   DAYS_ENDDATE_FACT       float64
 8   AMT_CREDIT_MAX_OVERDUE  float64
 9   CNT_CREDIT_PROLONG      int64
 10  AMT_CREDIT_SUM          float64
 11  AMT_CREDIT_SUM_DEBT     float64
 12  AMT_CREDIT_SUM_LIMIT    float64
 13  AMT_CREDIT_SUM_OVERDUE  float64
```

```
 14  CREDIT_TYPE             object
 15  DAYS_CREDIT_UPDATE      int64
 16  AMT_ANNUITY             float64
dtypes: float64(8), int64(6), object(3)
memory usage: 222.6+ MB
```

```
bureau.head()
```

```
   SK_ID_CURR  SK_ID_BUREAU CREDIT_ACTIVE CREDIT_CURRENCY  DAYS_CREDIT
\
0      215354       5714462        Closed      currency 1         -497

1      215354       5714463        Active      currency 1         -208

2      215354       5714464        Active      currency 1         -203

3      215354       5714465        Active      currency 1         -203

4      215354       5714466        Active      currency 1         -629


   CREDIT_DAY_OVERDUE  DAYS_CREDIT_ENDDATE  DAYS_ENDDATE_FACT  \
0                   0               -153.0             -153.0
1                   0               1075.0                NaN
2                   0                528.0                NaN
3                   0                  NaN                NaN
4                   0               1197.0                NaN

   AMT_CREDIT_MAX_OVERDUE  CNT_CREDIT_PROLONG  AMT_CREDIT_SUM  \
0                     NaN                   0         91323.0
1                     NaN                   0        225000.0
2                     NaN                   0        464323.5
3                     NaN                   0         90000.0
4                 77674.5                   0       2700000.0

   AMT_CREDIT_SUM_DEBT  AMT_CREDIT_SUM_LIMIT
AMT_CREDIT_SUM_OVERDUE  \
0                  0.0                   NaN                      0.0

1             171342.0                   NaN                      0.0

2                  NaN                   NaN                      0.0

3                  NaN                   NaN                      0.0

4                  NaN                   NaN                      0.0


       CREDIT_TYPE  DAYS_CREDIT_UPDATE  AMT_ANNUITY
```

```
0  Consumer credit                -131          NaN
1      Credit card                 -20          NaN
2  Consumer credit                 -16          NaN
3      Credit card                 -16          NaN
4  Consumer credit                 -21          NaN
```

```
col_descriptions[col_descriptions['Table']=='bureau.csv']
```

```
          Table                      Row  \
125  bureau.csv              SK_ID_CURR
126  bureau.csv            SK_BUREAU_ID
127  bureau.csv           CREDIT_ACTIVE
128  bureau.csv         CREDIT_CURRENCY
129  bureau.csv             DAYS_CREDIT
130  bureau.csv      CREDIT_DAY_OVERDUE
131  bureau.csv     DAYS_CREDIT_ENDDATE
132  bureau.csv       DAYS_ENDDATE_FACT
133  bureau.csv  AMT_CREDIT_MAX_OVERDUE
134  bureau.csv      CNT_CREDIT_PROLONG
135  bureau.csv          AMT_CREDIT_SUM
136  bureau.csv     AMT_CREDIT_SUM_DEBT
137  bureau.csv    AMT_CREDIT_SUM_LIMIT
138  bureau.csv  AMT_CREDIT_SUM_OVERDUE
139  bureau.csv             CREDIT_TYPE
140  bureau.csv      DAYS_CREDIT_UPDATE
141  bureau.csv             AMT_ANNUITY


Description  \
125  ID of loan in our sample - one loan in our sample can have 0,1,2
or more related previous credits in credit bureau
126            Recoded ID of previous Credit Bureau credit related to
our loan (unique coding for each loan application)
127
Status of the Credit Bureau (CB) reported credits
128
Recoded currency of the Credit Bureau credit
129                                  How many days before current
application did client apply for Credit Bureau credit
130                       Number of days past due on CB credit at the
time of application for related loan in our sample
131                                 Remaining duration of CB credit
(in days) at the time of application in Home Credit
132                        Days since CB credit ended at the time of
application in Home Credit (only for closed credit)
133                Maximal amount overdue on the Credit Bureau credit
so far (at application date of loan in our sample)
134                                                              How
many times was the Credit Bureau credit prolonged
135
```

```
Current credit amount for the Credit Bureau credit
136
Current debt on Credit Bureau credit
137                                                        Current
credit limit of credit card reported in Credit Bureau
138
Current amount overdue on Credit Bureau credit
139
Type of Credit Bureau credit (Car, cash,...)
140                       How many days before loan application did
last information about the Credit Bureau credit come
141
Annuity of the Credit Bureau credit

                                   Special
125                                 hashed
126                                 hashed
127                                    NaN
128                                recoded
129  time only relative to the application
130                                    NaN
131  time only relative to the application
132  time only relative to the application
133                                    NaN
134                                    NaN
135                                    NaN
136                                    NaN
137                                    NaN
138                                    NaN
139                                    NaN
140  time only relative to the application
141                                    NaN
```

```
bureau.describe()
```

```
         SK_ID_CURR  SK_ID_BUREAU   DAYS_CREDIT  CREDIT_DAY_OVERDUE  \
count  1.716428e+06  1.716428e+06  1.716428e+06        1.716428e+06
mean   2.782149e+05  5.924434e+06 -1.142108e+03        8.181666e-01
std    1.029386e+05  5.322657e+05  7.951649e+02        3.654443e+01
min    1.000010e+05  5.000000e+06 -2.922000e+03        0.000000e+00
25%    1.888668e+05  5.463954e+06 -1.666000e+03        0.000000e+00
50%    2.780550e+05  5.926304e+06 -9.870000e+02        0.000000e+00
75%    3.674260e+05  6.385681e+06 -4.740000e+02        0.000000e+00
max    4.562550e+05  6.843457e+06  0.000000e+00        2.792000e+03

       DAYS_CREDIT_ENDDATE  DAYS_ENDDATE_FACT  AMT_CREDIT_MAX_OVERDUE
\
count         1.610875e+06       1.082775e+06            5.919400e+05

mean          5.105174e+02      -1.017437e+03            3.825418e+03
```

std	4.994220e+03	7.140106e+02	2.060316e+05
min	-4.206000e+04	-4.202300e+04	0.000000e+00
25%	-1.138000e+03	-1.489000e+03	0.000000e+00
50%	-3.300000e+02	-8.970000e+02	0.000000e+00
75%	4.740000e+02	-4.250000e+02	0.000000e+00
max	3.119900e+04	0.000000e+00	1.159872e+08

	CNT_CREDIT_PROLONG	AMT_CREDIT_SUM	AMT_CREDIT_SUM_DEBT \
count	1.716428e+06	1.716415e+06	1.458759e+06
mean	6.410406e-03	3.549946e+05	1.370851e+05
std	9.622391e-02	1.149811e+06	6.774011e+05
min	0.000000e+00	0.000000e+00	-4.705600e+06
25%	0.000000e+00	5.130000e+04	0.000000e+00
50%	0.000000e+00	1.255185e+05	0.000000e+00
75%	0.000000e+00	3.150000e+05	4.015350e+04
max	9.000000e+00	5.850000e+08	1.701000e+08

	AMT_CREDIT_SUM_LIMIT	AMT_CREDIT_SUM_OVERDUE	DAYS_CREDIT_UPDATE \
count	1.124648e+06	1.716428e+06	1.716428e+06
mean	6.229515e+03	3.791276e+01	-5.937483e+02
std	4.503203e+04	5.937650e+03	7.207473e+02
min	-5.864061e+05	0.000000e+00	-4.194700e+04
25%	0.000000e+00	0.000000e+00	-9.080000e+02
50%	0.000000e+00	0.000000e+00	-3.950000e+02
75%	0.000000e+00	0.000000e+00	-3.300000e+01
max	4.705600e+06	3.756681e+06	3.720000e+02

	AMT_ANNUITY
count	4.896370e+05
mean	1.571276e+04
std	3.258269e+05
min	0.000000e+00
25%	0.000000e+00

```
50%    0.000000e+00
75%    1.350000e+04
max    1.184534e+08
```

```
#Null columns
bureau.isna().sum()
```

```
SK_ID_CURR                      0
SK_ID_BUREAU                    0
CREDIT_ACTIVE                   0
CREDIT_CURRENCY                 0
DAYS_CREDIT                     0
CREDIT_DAY_OVERDUE              0
DAYS_CREDIT_ENDDATE        105553
DAYS_ENDDATE_FACT          633653
AMT_CREDIT_MAX_OVERDUE    1124488
CNT_CREDIT_PROLONG              0
AMT_CREDIT_SUM                 13
AMT_CREDIT_SUM_DEBT        257669
AMT_CREDIT_SUM_LIMIT       591780
AMT_CREDIT_SUM_OVERDUE          0
CREDIT_TYPE                     0
DAYS_CREDIT_UPDATE              0
AMT_ANNUITY               1226791
dtype: int64
```

SK_ID_CURR

ID of loan in our sample - one loan in our sample can have 0,1,2 or more related previous credits in credit bureau

```
#Unique values
bureau.SK_ID_CURR.nunique()
```

```
305811
```

- SO we have lots of dupliacte for SK_ID_CURR,This is foreign key for this table and primary key for application -train|test.csv

SK_ID_BUREAU

Recoded ID of previous Credit Bureau credit related to our loan (unique coding for each loan application)

```
#Unique values
bureau.SK_ID_BUREAU.nunique()
```

```
1716428
```

```
bureau_cat_cols=bureau_merged.select_dtypes(include=[object]).columns.
to_list()
```

```
#Lets see its coorrelation

sns.heatmap(bureau_merged.drop(bureau_cat_cols,axis=1).corr(),cmap="Yl
GnBu")
```

- No column is corelated to Target
- High correlation between
  - DAYS_CRDIT and DAYS_FACT
  - DAYS_CRDIT and DAYS_CREDIT_UPDATE
  - AMT_CREDIT_DEBT and AMT_CREDIT_SUM
  - DAYS_CREDIT_UPDATE AND DAYS_ENDDATE_FACT

CREDIT_ACTIVE

Status of the Credit Bureau (CB) reported credits

```
bureau['CREDIT_ACTIVE'].isna().sum()
```

```
0
```

```
bureau['CREDIT_ACTIVE'].value_counts()
```

```
Closed      1079273
Active       630607
Sold           6527
Bad debt         21
Name: CREDIT_ACTIVE, dtype: int64
```

```
#lets add Target column in order to analyze the bureau data

bureau_merged =
df_train[['SK_ID_CURR','TARGET']].merge(bureau,how='left',on='SK_ID_CU
RR')
```

```
bureau_merged.head()
```

```
   SK_ID_CURR  TARGET  SK_ID_BUREAU CREDIT_ACTIVE CREDIT_CURRENCY  \
0      100002       1     6158904.0        Closed      currency 1
1      100002       1     6158905.0        Closed      currency 1
2      100002       1     6158906.0        Closed      currency 1
3      100002       1     6158907.0        Closed      currency 1
4      100002       1     6158908.0        Closed      currency 1
```

```
   DAYS_CREDIT  CREDIT_DAY_OVERDUE  DAYS_CREDIT_ENDDATE
DAYS_ENDDATE_FACT  \
0      -1125.0                0.0              -1038.0            -
1038.0
1       -476.0                0.0                  NaN              -
48.0
2      -1437.0                0.0              -1072.0            -
1185.0
3      -1121.0                0.0               -911.0             -
911.0
4       -645.0                0.0                 85.0              -
36.0

   AMT_CREDIT_MAX_OVERDUE  CNT_CREDIT_PROLONG  AMT_CREDIT_SUM  \
0                     NaN                 0.0         40761.0
1                     NaN                 0.0             0.0
2                   0.000                 0.0        135000.0
3                3321.000                 0.0         19071.0
4                5043.645                 0.0        120735.0

   AMT_CREDIT_SUM_DEBT  AMT_CREDIT_SUM_LIMIT
AMT_CREDIT_SUM_OVERDUE  \
0                  NaN                   NaN                      0.0

1                  0.0                   NaN                      0.0

2                  0.0                   0.0                      0.0

3                  NaN                   NaN                      0.0

4                  0.0                   0.0                      0.0


       CREDIT_TYPE  DAYS_CREDIT_UPDATE  AMT_ANNUITY
0      Credit card             -1038.0          0.0
1      Credit card               -47.0          NaN
2  Consumer credit             -1185.0          0.0
3  Consumer credit              -906.0          0.0
4  Consumer credit               -34.0          0.0
```

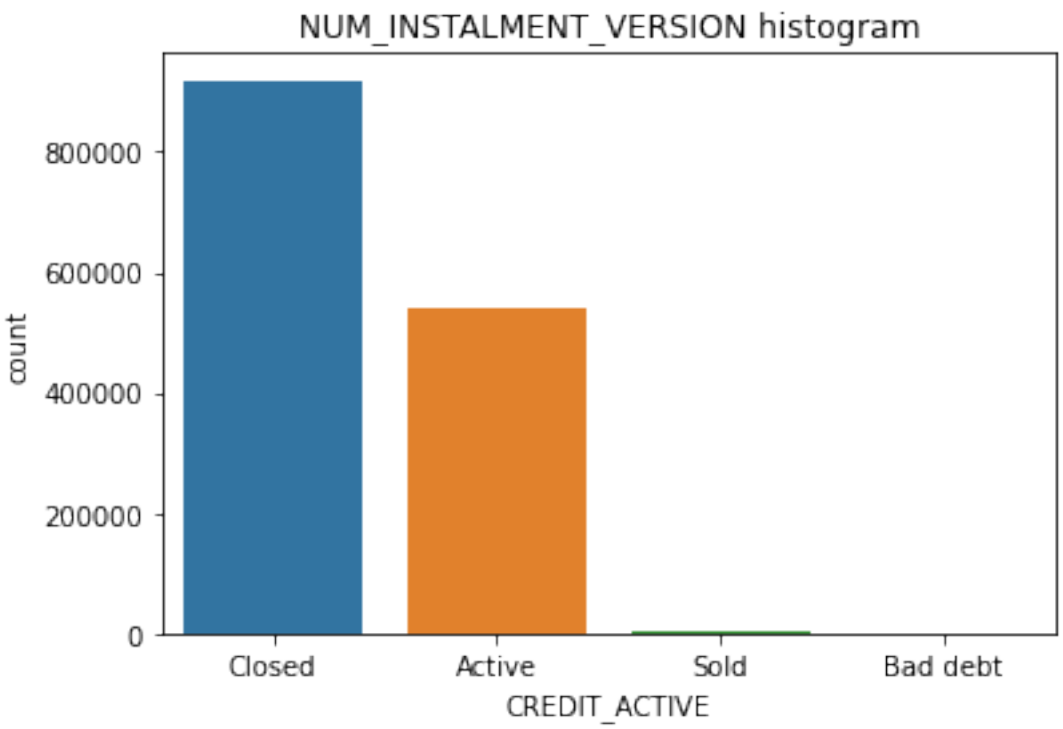
```
bureau_merged.groupby(['TARGET','CREDIT_ACTIVE'])
['CREDIT_ACTIVE'].size()
```

```
TARGET  CREDIT_ACTIVE
0       Active           491583
        Bad debt             16
        Closed           854124
        Sold               5082
```

```
1       Active            50336
        Bad debt              4
        Closed            63609
        Sold                571
Name: CREDIT_ACTIVE, dtype: int64
```

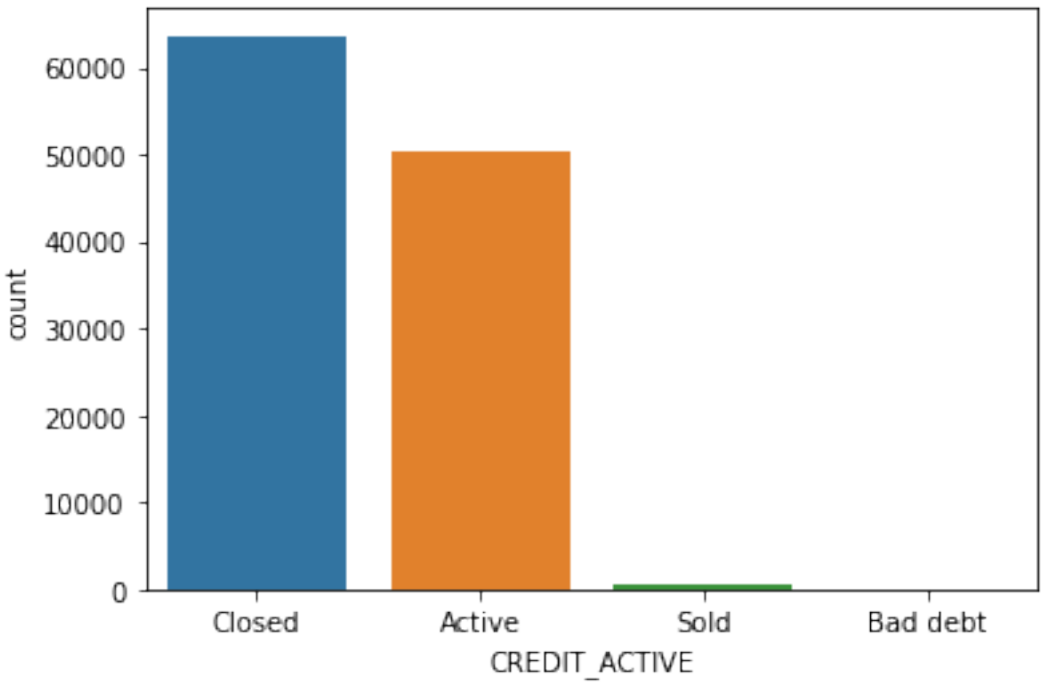
```
sns.countplot(bureau_merged['CREDIT_ACTIVE'])
plt.title("NUM_INSTALMENT_VERSION histogram")
```

```
Text(0.5, 1.0, 'NUM_INSTALMENT_VERSION histogram')
```



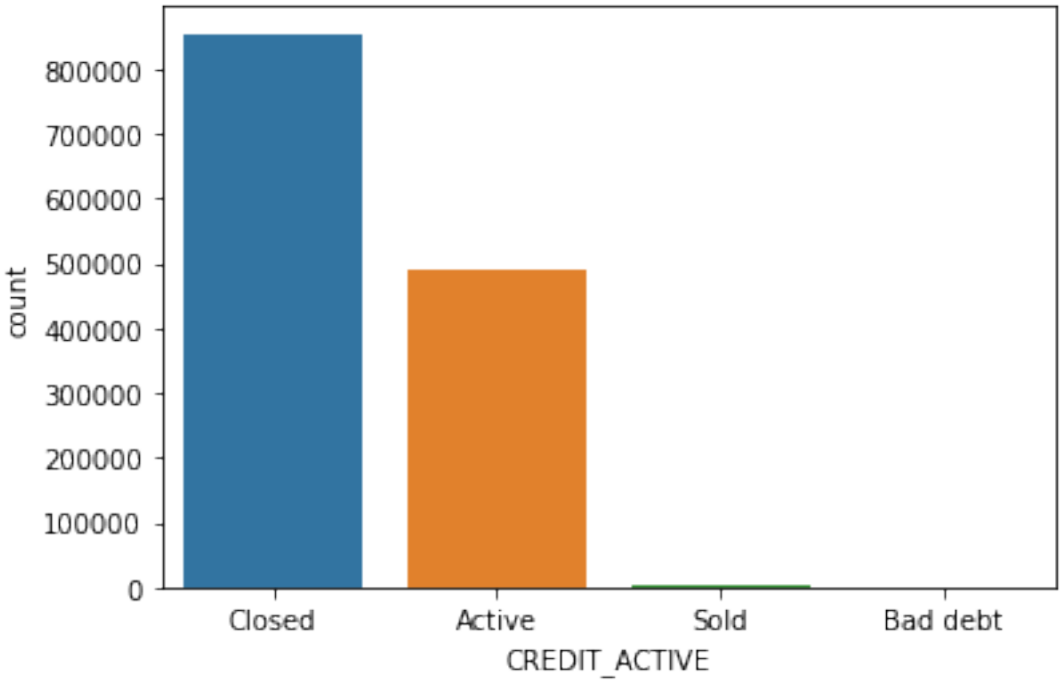

```
sns.countplot(bureau_merged[bureau_merged['TARGET']==1]
['CREDIT_ACTIVE'])
```

```
<AxesSubplot:xlabel='CREDIT_ACTIVE', ylabel='count'>
```



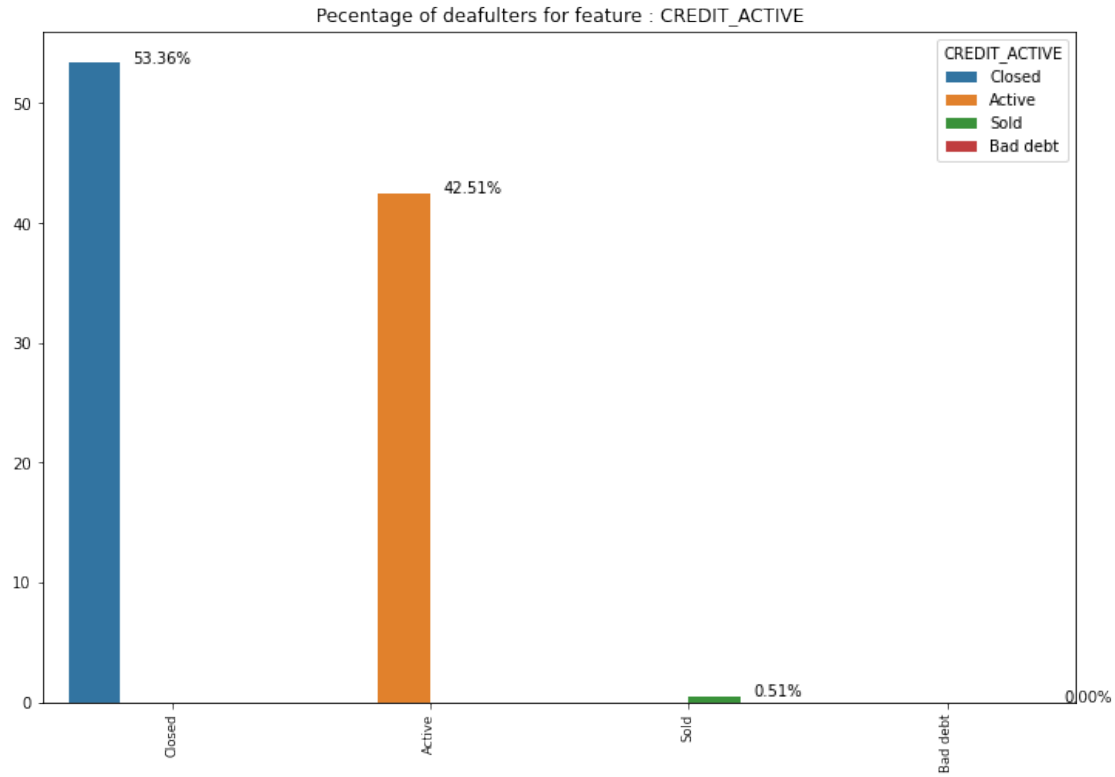

```
sns.countplot(bureau_merged[bureau_merged['TARGET']==0]
['CREDIT_ACTIVE'])
```

```
<AxesSubplot:xlabel='CREDIT_ACTIVE', ylabel='count'>
```




```
plot_for_categorical_defaulters(bureau_merged,'CREDIT_ACTIVE')
```

```
{'Closed': 53.361992051261254,
 'Active': 42.505474896585284,
 'Sold': 0.5109903479600941,
 'Bad debt': 0.0}
```




```
#we can see for Target = 0 : Non defaulters
#Active percrentage is
491583/(854124)
```

```
0.5755405538305913
```

```
#we can see for Target = 1 : defaulters
#Active percrentage is
50336/(63609)
```

```
0.791334559574903
```

```
#So Active/closed ratio is more for the defaulters
```

DAYS_CREDIT

How many days before current application did client apply for Credit Bureau credit

```
bureau_merged['DAYS_CREDIT'].describe()
```

```
count    1.465325e+06
mean    -1.143600e+03
```

```
std      8.000200e+02
min     -2.922000e+03
25%     -1.671000e+03
50%     -9.850000e+02
75%     -4.700000e+02
max      0.000000e+00
Name: DAYS_CREDIT, dtype: float64
```

```
bureau_merged['DAYS_CREDIT'].isna().sum()
```

```
44020
```

```
# total 44020 are null
```

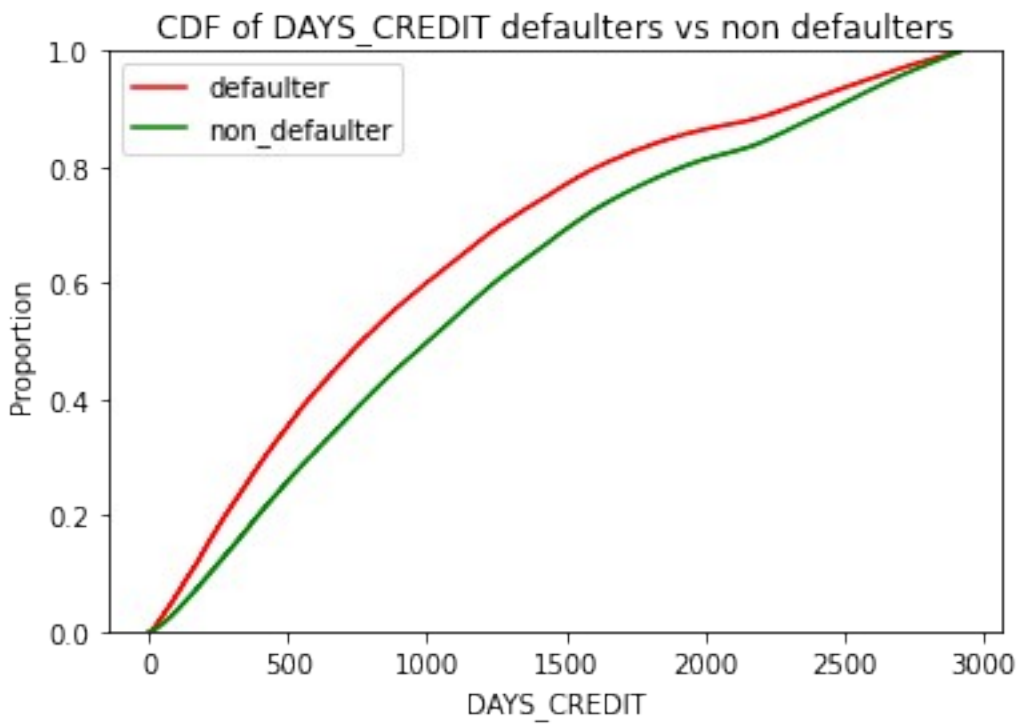
```
bureau_merged[bureau_merged['DAYS_CREDIT'].isna()].isna().sum()
```

```
SK_ID_CURR                    0
TARGET                        0
SK_ID_BUREAU              44020
CREDIT_ACTIVE             44020
CREDIT_CURRENCY           44020
DAYS_CREDIT               44020
CREDIT_DAY_OVERDUE        44020
DAYS_CREDIT_ENDDATE       44020
DAYS_ENDDATE_FACT         44020
AMT_CREDIT_MAX_OVERDUE    44020
CNT_CREDIT_PROLONG        44020
AMT_CREDIT_SUM            44020
AMT_CREDIT_SUM_DEBT       44020
AMT_CREDIT_SUM_LIMIT      44020
AMT_CREDIT_SUM_OVERDUE    44020
CREDIT_TYPE               44020
DAYS_CREDIT_UPDATE        44020
AMT_ANNUITY               44020
dtype: int64
```

```
#SO where DAYS_CREDIT is NULL all other columns values are also
NULL,There are total 44020 of rows as suchhere
# I think these record are for those customer whose credit bureau
record are not available in the Bureau table
```

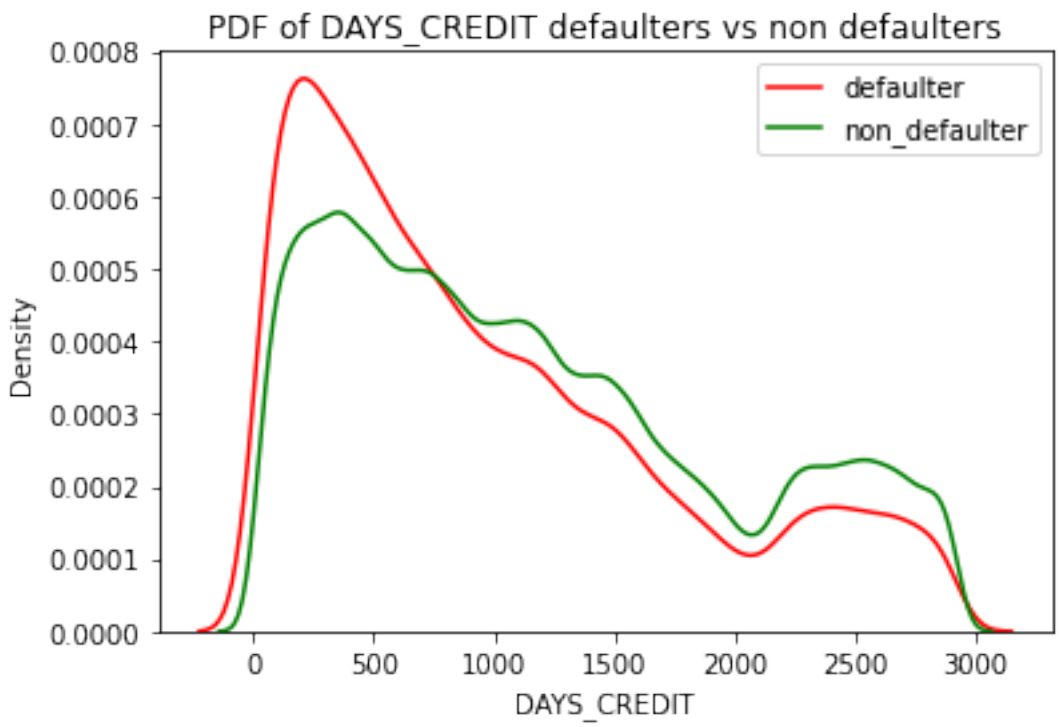
```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(-1*bureau_merged[bureau_merged['TARGET']==1]
['DAYS_CREDIT'],label='defaulter',color='red')
sns.ecdfplot(-1*bureau_merged[bureau_merged['TARGET']==0]
['DAYS_CREDIT'],label='non_defaulter',color='green')
#plt.xlim([40000,1000000])
plt.title("CDF of DAYS_CREDIT defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d149011a00>
```




```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(-1*bureau_merged[bureau_merged['TARGET']==1]
['DAYS_CREDIT'],label='defaulter',color='red')
sns.kdeplot(-1*bureau_merged[bureau_merged['TARGET']==0]
['DAYS_CREDIT'],label='non_defaulter',color='green')
#plt.xlim([40000,1000000])
plt.title("PDF of DAYS_CREDIT defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

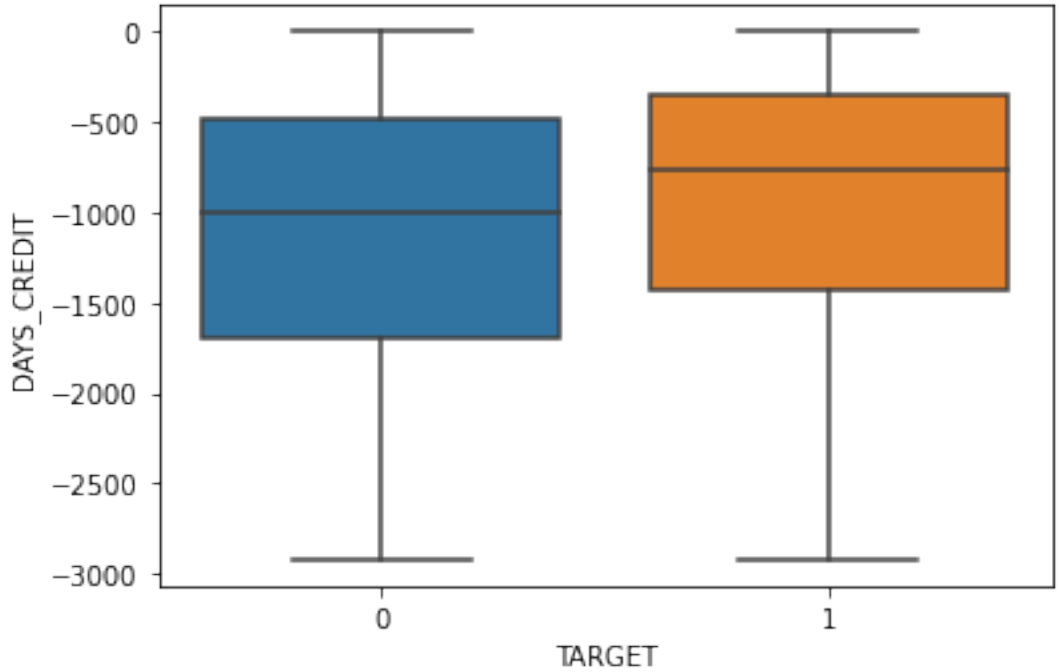
```
<matplotlib.legend.Legend at 0x1d149039bb0>
```




```
#we can see there aree more defaulters lies in between -1 tp -800
DAYS_CREDIT
```

```
sns.boxplot(x='TARGET',y='DAYS_CREDIT',data=bureau_merged)
plt.title("Boxplot for DAYS_CREDIT")
```

```
<AxesSubplot:xlabel='TARGET', ylabel='DAYS_CREDIT'>
```



- DAYS_CREDIT is lower for defaulters than non defaulters

CREDIT_DAY_OVERDUE

Number of days past due on CB credit at the time of application for related loan in our sample

```
bureau_merged['CREDIT_DAY_OVERDUE'].describe()
```

```
count    1.465325e+06
mean     8.887612e-01
std      3.824707e+01
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      0.000000e+00
max      2.792000e+03
Name: CREDIT_DAY_OVERDUE, dtype: float64
```

```
#bureau_merged['CREDIT_DAY_OVERDUE'].value_counts()
```

```
bureau_merged['CREDIT_DAY_OVERDUE'].isna().sum()
```

```
44020
```

```
# total 44020 are null,same as DAYS_CREDIT were NULL
```

```
bureau_merged['CREDIT_DAY_OVERDUE'].notnull().sum()
```

```
1465325
```

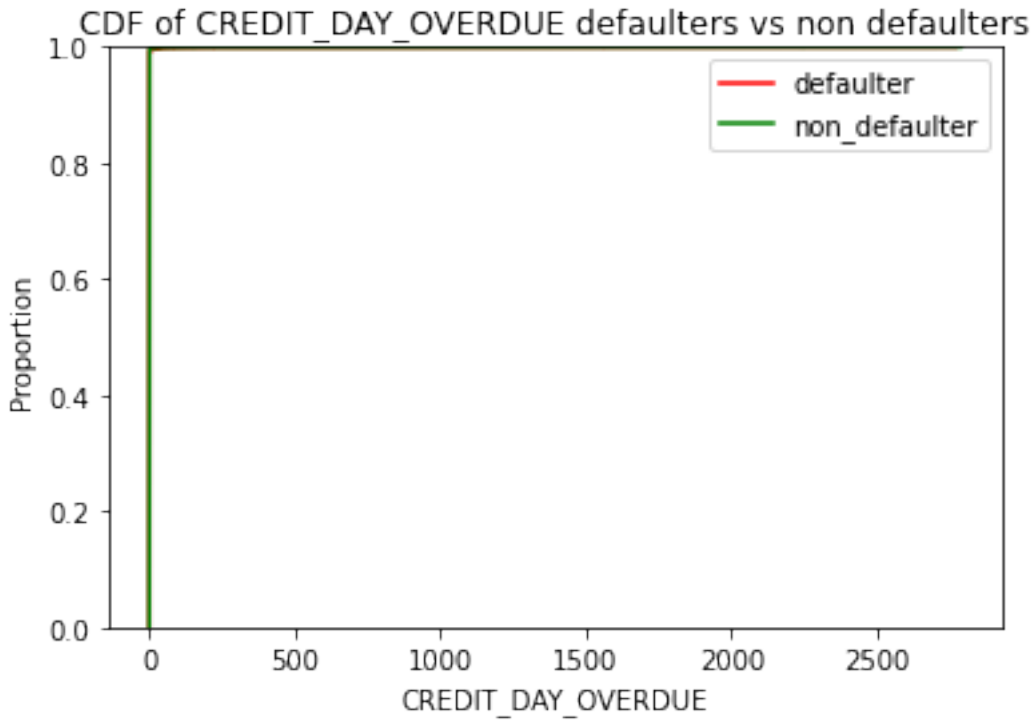
```
bureau_merged[bureau_merged['CREDIT_DAY_OVERDUE'].isna()].isna().sum()
```

```
SK_ID_CURR                    0
TARGET                        0
SK_ID_BUREAU              44020
CREDIT_ACTIVE             44020
CREDIT_CURRENCY           44020
DAYS_CREDIT               44020
CREDIT_DAY_OVERDUE        44020
DAYS_CREDIT_ENDDATE       44020
DAYS_ENDDATE_FACT         44020
AMT_CREDIT_MAX_OVERDUE    44020
CNT_CREDIT_PROLONG        44020
AMT_CREDIT_SUM            44020
AMT_CREDIT_SUM_DEBT       44020
AMT_CREDIT_SUM_LIMIT      44020
AMT_CREDIT_SUM_OVERDUE    44020
CREDIT_TYPE               44020
DAYS_CREDIT_UPDATE        44020
```

```
AMT_ANNUITY                44020
dtype: int64
```

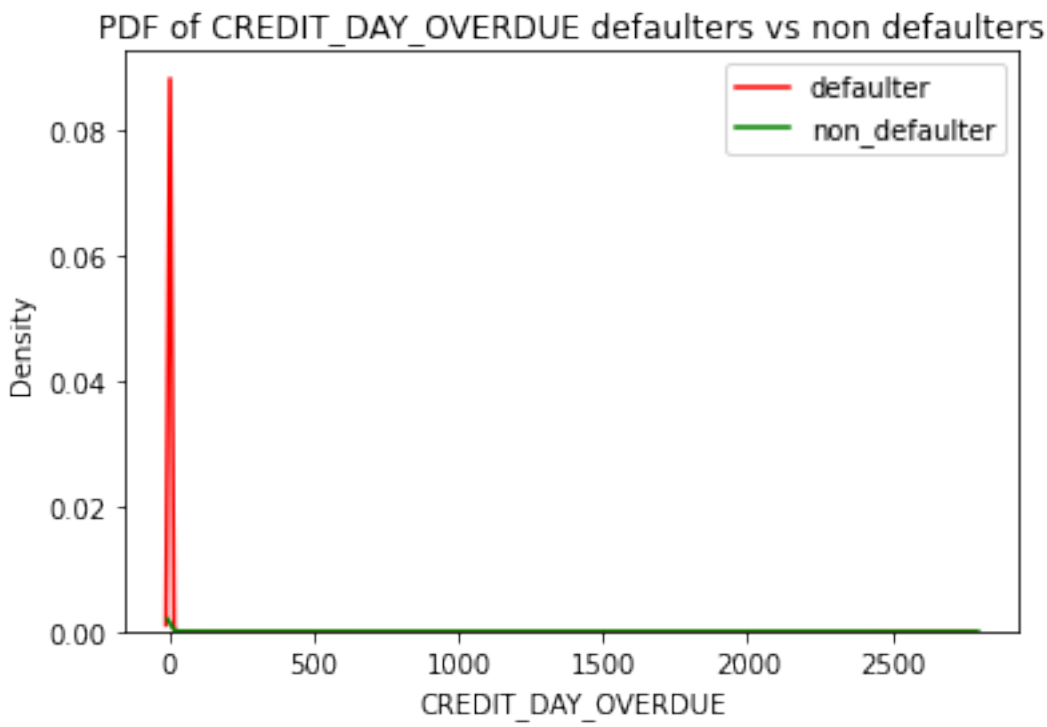
```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==1]
['CREDIT_DAY_OVERDUE'],label='defaulter',color='red')
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==0]
['CREDIT_DAY_OVERDUE'],label='non_defaulter',color='green')
#plt.xlim([0,300])
plt.title("CDF of CREDIT_DAY_OVERDUE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d148a72f40>
```




```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==1]
['CREDIT_DAY_OVERDUE'],label='defaulter',color='red')
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==0]
['CREDIT_DAY_OVERDUE'],label='non_defaulter',color='green')
#plt.xlim([40000,1000000])
plt.title("PDF of CREDIT_DAY_OVERDUE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

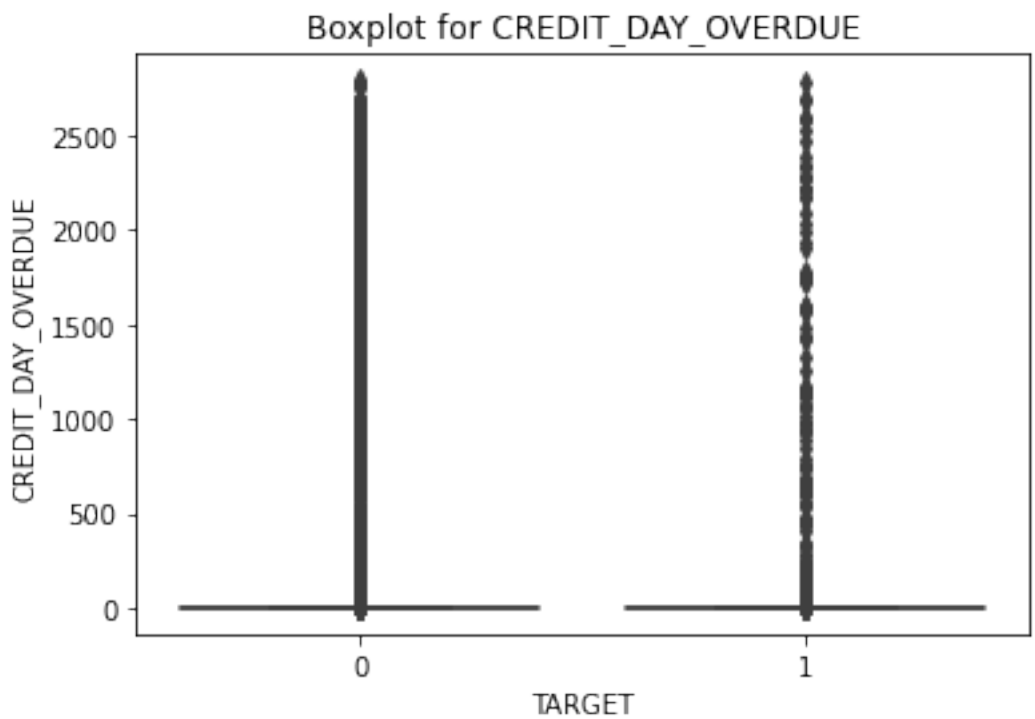
```
<matplotlib.legend.Legend at 0x1d2392f4220>
```




```
#we can see there aree most values are just zero
```

```
sns.boxplot(x='TARGET',y='CREDIT_DAY_OVERDUE',data=bureau_merged)
plt.title("Boxplot for CREDIT_DAY_OVERDUE")
```

```
Text(0.5, 1.0, 'Boxplot for CREDIT_DAY_OVERDUE')
```



```
#we can there are total 1465325 not null values here in which 1461619
```

```
1461619/1465325
```

```
0.9974708682374217
```

```
#So 99.75 % are just zeros
```

DAYS_CREDIT_ENDDATE

Remaining duration of CB credit (in days) at the time of application in Home Credit

```
bureau_merged['DAYS_CREDIT_ENDDATE'].describe()
```

```
count    1.376227e+06
mean     4.931322e+02
std      4.964905e+03
min     -4.206000e+04
25%     -1.144000e+03
50%     -3.340000e+02
75%      4.730000e+02
max      3.119900e+04
Name: DAYS_CREDIT_ENDDATE, dtype: float64
```

```
bureau_merged['CREDIT_DAY_OVERDUE'].isna().sum()
```

```
44020
```

```
# total 44020 are null,same as DAYS_CREDIT were NULL
```

```
bureau_merged['DAYS_CREDIT_ENDDATE'].notnull().sum()
```

```
1376227
```

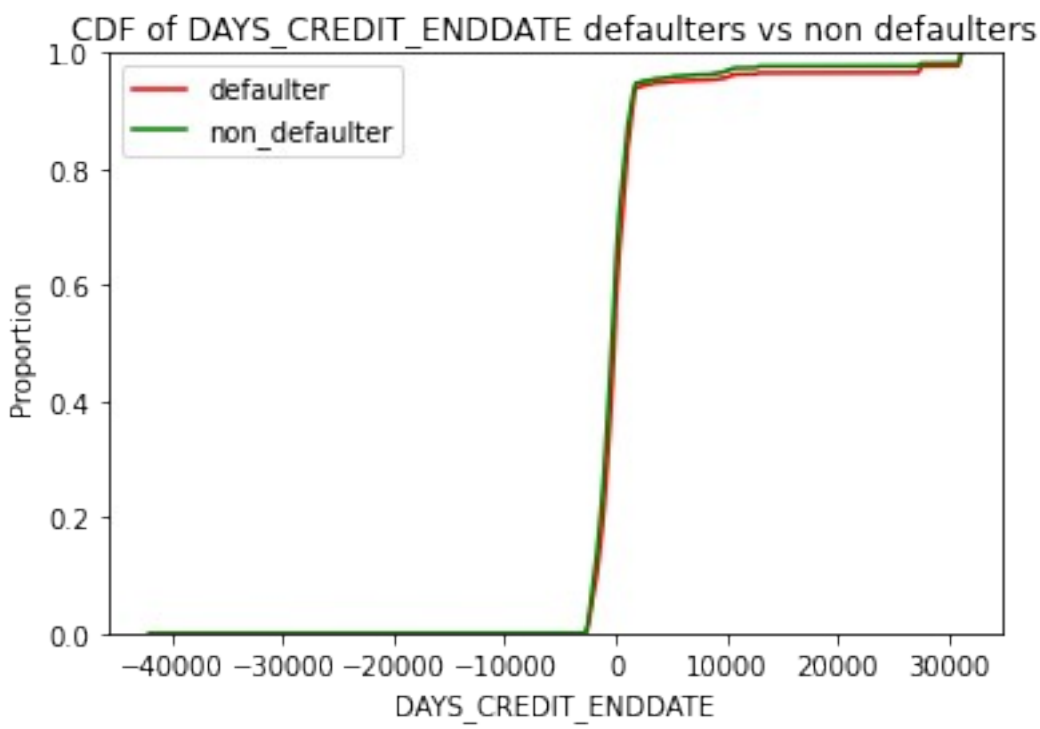
```
bureau_merged[bureau_merged['DAYS_CREDIT_ENDDATE'].isna()].isna().sum(
)
```

```
SK_ID_CURR                     0
TARGET                         0
SK_ID_BUREAU               44020
CREDIT_ACTIVE              44020
CREDIT_CURRENCY            44020
DAYS_CREDIT                44020
CREDIT_DAY_OVERDUE         44020
DAYS_CREDIT_ENDDATE       133118
DAYS_ENDDATE_FACT         106319
AMT_CREDIT_MAX_OVERDUE     84930
CNT_CREDIT_PROLONG         44020
AMT_CREDIT_SUM             44023
AMT_CREDIT_SUM_DEBT        64912
AMT_CREDIT_SUM_LIMIT       82277
AMT_CREDIT_SUM_OVERDUE     44020
```

```
CREDIT_TYPE                 44020
DAYS_CREDIT_UPDATE          44020
AMT_ANNUITY                109505
dtype: int64
```

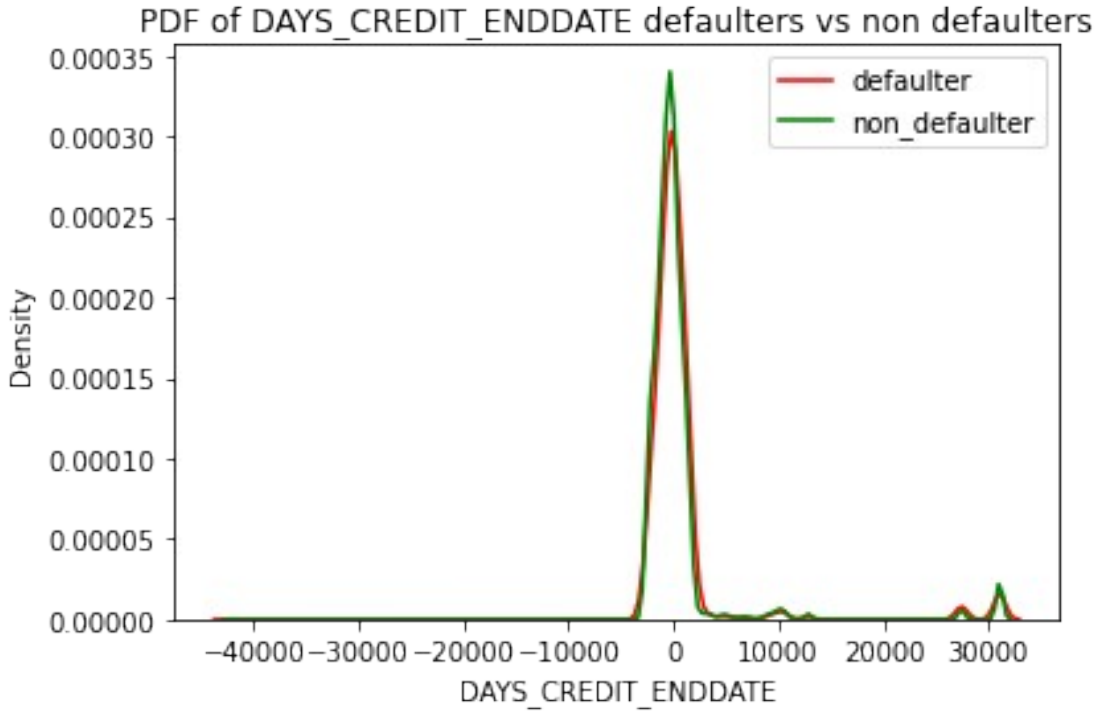
```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==1]
['DAYS_CREDIT_ENDDATE'],label='defaulter',color='red')
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==0]
['DAYS_CREDIT_ENDDATE'],label='non_defaulter',color='green')
#plt.xlim([0,300])
plt.title("CDF of DAYS_CREDIT_ENDDATE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d1266885b0>
```



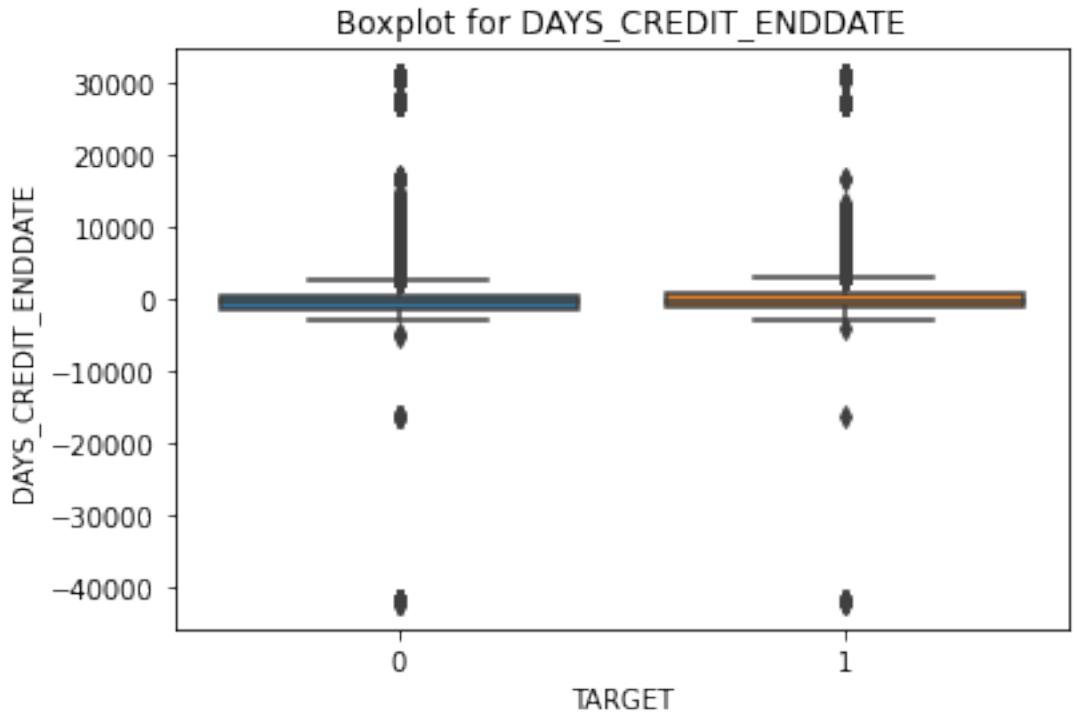

```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==1]
['DAYS_CREDIT_ENDDATE'],label='defaulter',color='red')
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==0]
['DAYS_CREDIT_ENDDATE'],label='non_defaulter',color='green')
#plt.xlim([40000,1000000])
plt.title("PDF of DAYS_CREDIT_ENDDATE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d148fcf9a0>
```




```
#we can see there aree most values are just zero

sns.boxplot(x='TARGET',y='DAYS_CREDIT_ENDDATE',data=bureau_merged)
plt.title("Boxplot for DAYS_CREDIT_ENDDATE")
```

```
Text(0.5, 1.0, 'Boxplot for DAYS_CREDIT_ENDDATE')
```



- DAYS_CREDIT_ENDDATE distribution looks same for both categoires for TARGET

DAYS_ENDDATE_FACT

Days since CB credit ended at the time of application in Home Credit (only for closed credit)

```
bureau_merged['DAYS_ENDDATE_FACT'].describe()
```

```
count    920652.000000
mean      -1022.765286
std         718.371749
min      -42023.000000
25%       -1503.000000
50%        -900.000000
75%        -427.000000
max           0.000000
Name: DAYS_ENDDATE_FACT, dtype: float64
```

```
bureau_merged['DAYS_ENDDATE_FACT'].isna().sum()
```

```
588693
```

```
# total 588693 are null
```

```
bureau_merged['DAYS_ENDDATE_FACT'].notnull().sum()
```

```
920652
```

```
bureau_merged[bureau_merged['DAYS_ENDDATE_FACT'].isna()].isna().sum()
```

```
SK_ID_CURR                     0
TARGET                         0
SK_ID_BUREAU               44020
CREDIT_ACTIVE              44020
CREDIT_CURRENCY            44020
DAYS_CREDIT                44020
CREDIT_DAY_OVERDUE         44020
DAYS_CREDIT_ENDDATE       106319
DAYS_ENDDATE_FACT         588693
AMT_CREDIT_MAX_OVERDUE    397482
CNT_CREDIT_PROLONG         44020
AMT_CREDIT_SUM             44022
AMT_CREDIT_SUM_DEBT       112408
AMT_CREDIT_SUM_LIMIT      193482
AMT_CREDIT_SUM_OVERDUE     44020
CREDIT_TYPE                44020
DAYS_CREDIT_UPDATE         44020
AMT_ANNUITY               457152
dtype: int64
```

```
bureau_merged[bureau_merged['DAYS_ENDDATE_FACT'].isna()].head(20)
```

```
    SK_ID_CURR  TARGET  SK_ID_BUREAU CREDIT_ACTIVE CREDIT_CURRENCY  \
5       100002       1     6158909.0        Active      currency 1
6       100002       1     6158903.0        Active      currency 1
11      100003       0     5885880.0        Active      currency 1
14      100006       0           NaN           NaN             NaN
17      100008       0     6491434.0        Active      currency 1
26      100009       0     5651003.0        Active      currency 1
27      100009       0     5651004.0        Active      currency 1
31      100009       0     5651009.0        Active      currency 1
32      100009       0     5651010.0        Active      currency 1
37      100010       0     5576630.0        Active      currency 1
43      100012       0           NaN           NaN             NaN
50      100014       0     5326772.0        Active      currency 1
51      100014       0     5326773.0        Active      currency 1
56      100016       0     5217355.0        Active      currency 1
58      100016       0     5217357.0        Active      currency 1
59      100016       0     5217358.0        Active      currency 1
62      100016       0     5217354.0        Active      currency 1
69      100018       0           NaN           NaN             NaN
70      100019       0     5832095.0        Active      currency 1
71      100019       0     5832096.0        Active      currency 1

    DAYS_CREDIT  CREDIT_DAY_OVERDUE  DAYS_CREDIT_ENDDATE
DAYS_ENDDATE_FACT  \
5        -103.0                 0.0                  NaN
NaN
6       -1042.0                 0.0                780.0
NaN
11       -606.0                 0.0               1216.0
NaN
14          NaN                 NaN                  NaN
NaN
17        -78.0                 0.0                471.0
NaN
26       -239.0                 0.0               -209.0
NaN
27       -424.0                 0.0               1402.0
NaN
31      -1293.0                 0.0                  NaN
NaN
32       -411.0                 0.0                  NaN
NaN
37      -1138.0                 0.0                689.0
NaN
43          NaN                 NaN                  NaN
NaN
50       -423.0                 0.0                704.0
NaN
```

```
51       -376.0                 0.0                723.0
NaN
56       -141.0                 0.0                223.0
NaN
58       -128.0                 0.0                236.0
NaN
59       -142.0                 0.0                223.0
NaN
62       -262.0                 0.0                845.0
NaN
69          NaN                 NaN                  NaN
NaN
70       -495.0                 0.0              10463.0
NaN
71       -495.0                 0.0                419.0
NaN

    AMT_CREDIT_MAX_OVERDUE  CNT_CREDIT_PROLONG  AMT_CREDIT_SUM  \
5                    40.50                 0.0       31988.565
6                      NaN                 0.0      450000.000
11                    0.00                 0.0      810000.000
14                     NaN                 NaN             NaN
17                    0.00                 0.0      267606.000
26                     NaN                 0.0      337500.000
27                     NaN                 0.0      642861.000
31                     NaN                 0.0      187200.000
32                     NaN                 0.0      360000.000
37                     NaN                 0.0      675000.000
43                     NaN                 NaN             NaN
50                12752.28                 0.0      571500.000
51                    0.00                 0.0      434250.000
56                     NaN                 0.0       88996.500
58                     NaN                 0.0       91264.500
59                     NaN                 0.0       88996.500
62                     NaN                 0.0       67500.000
69                     NaN                 NaN             NaN
70                    0.00                 0.0      270000.000
71                    0.00                 0.0      450000.000

    AMT_CREDIT_SUM_DEBT  AMT_CREDIT_SUM_LIMIT  AMT_CREDIT_SUM_OVERDUE
\
5                  0.00             31988.565                     0.0

6             245781.00                 0.000                     0.0

11                 0.00            810000.000                     0.0

14                  NaN                   NaN                     NaN

17            240057.00                 0.000                     0.0
```

26	326628.00	0.000	0.0
27	557959.50	0.000	0.0
31	0.00	NaN	0.0
32	192762.00	NaN	0.0
37	348007.50	0.000	0.0
43	NaN	NaN	NaN
50	420201.00	0.000	0.0
51	338013.00	0.000	0.0
56	63724.50	0.000	0.0
58	NaN	NaN	0.0
59	NaN	NaN	0.0
62	0.00	0.000	0.0
69	NaN	NaN	NaN
70	0.00	270000.000	0.0
71	245470.14	0.000	0.0

	CREDIT_TYPE	DAYS_CREDIT_UPDATE	AMT_ANNUITY
5	Credit card	-24.0	0.0
6	Consumer credit	-7.0	0.0
11	Credit card	-43.0	NaN
14	NaN	NaN	NaN
17	Consumer credit	-16.0	NaN
26	Credit card	-86.0	NaN
27	Consumer credit	-27.0	NaN
31	Credit card	-847.0	NaN
32	Consumer credit	-23.0	NaN
37	Consumer credit	-18.0	NaN
43	NaN	NaN	NaN
50	Consumer credit	-25.0	NaN
51	Consumer credit	-34.0	NaN
56	Consumer credit	-8.0	NaN
58	Consumer credit	-3.0	NaN

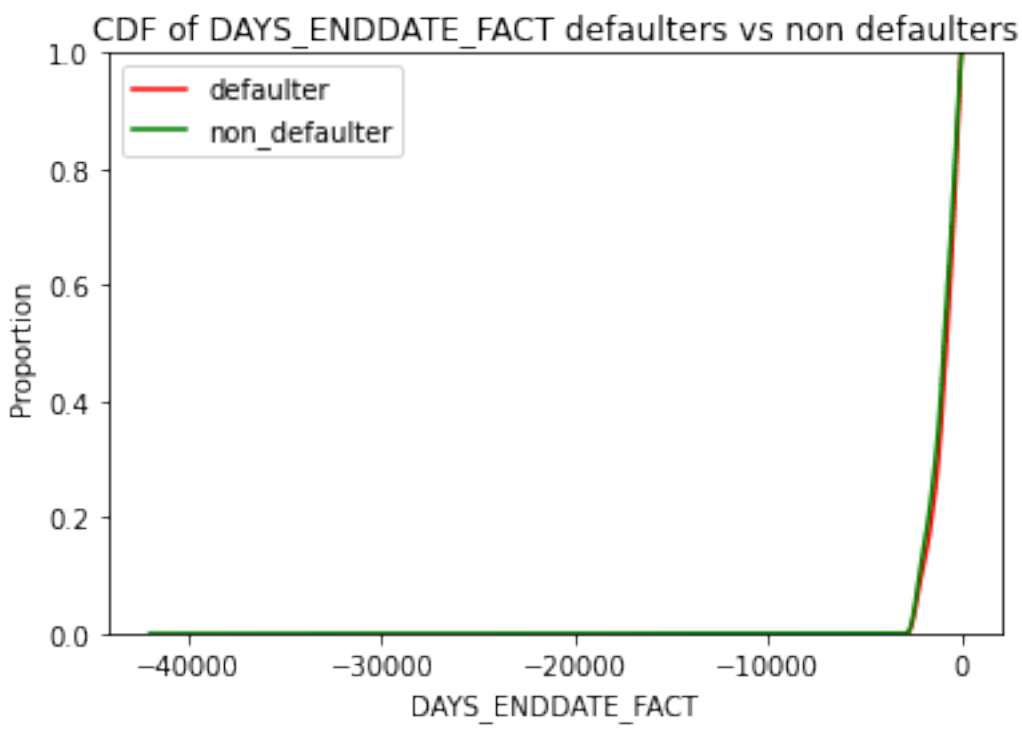
```
59  Consumer credit                -3.0          NaN
62      Credit card               -46.0          NaN
69              NaN                 NaN          NaN
70      Credit card               -46.0      27000.0
71  Consumer credit                -7.0      27000.0
```

```
bureau_merged[bureau_merged['DAYS_ENDDATE_FACT'].isna()]
['CREDIT_ACTIVE'].value_counts()
```

```
Active      540412
Sold          4157
Closed          93
Bad debt        11
Name: CREDIT_ACTIVE, dtype: int64
```

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==1]
['DAYS_ENDDATE_FACT'],label='defaulter',color='red')
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==0]
['DAYS_ENDDATE_FACT'],label='non_defaulter',color='green')
#plt.xlim([0,300])
plt.title("CDF of DAYS_ENDDATE_FACT defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

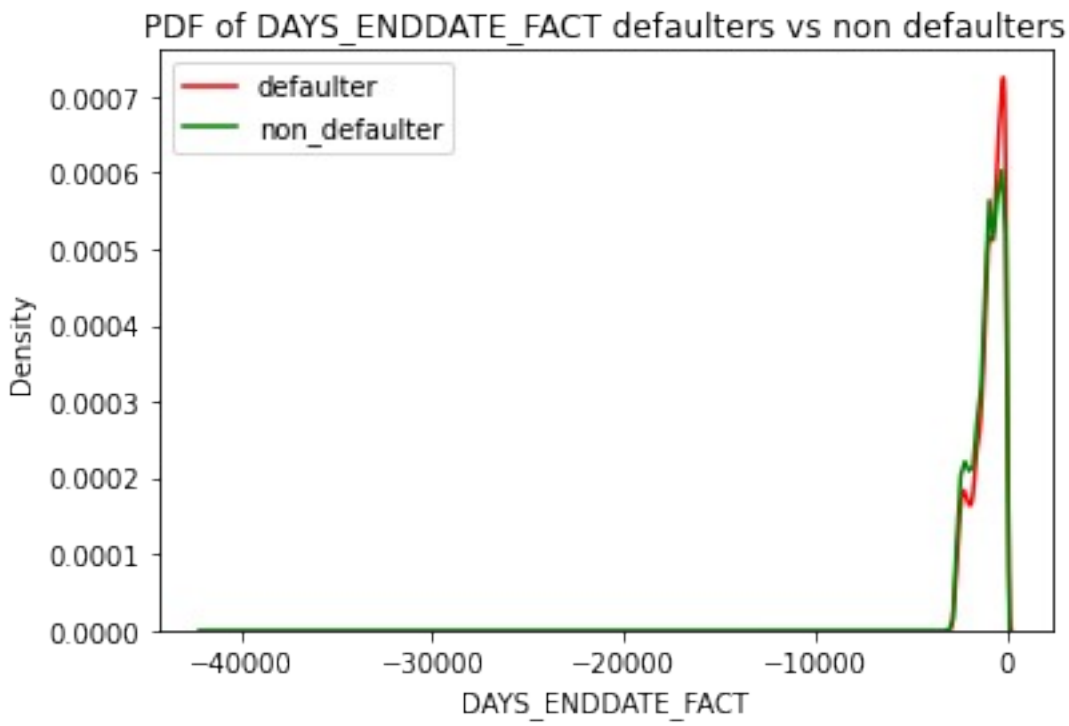
```
<matplotlib.legend.Legend at 0x1d20e0dfd90>
```




```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==1]
```

```
['DAYS_ENDDATE_FACT'],label='defaulter',color='red')
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==0]
['DAYS_ENDDATE_FACT'],label='non_defaulter',color='green')
#plt.xlim([40000,1000000])
plt.title("PDF of DAYS_ENDDATE_FACT defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

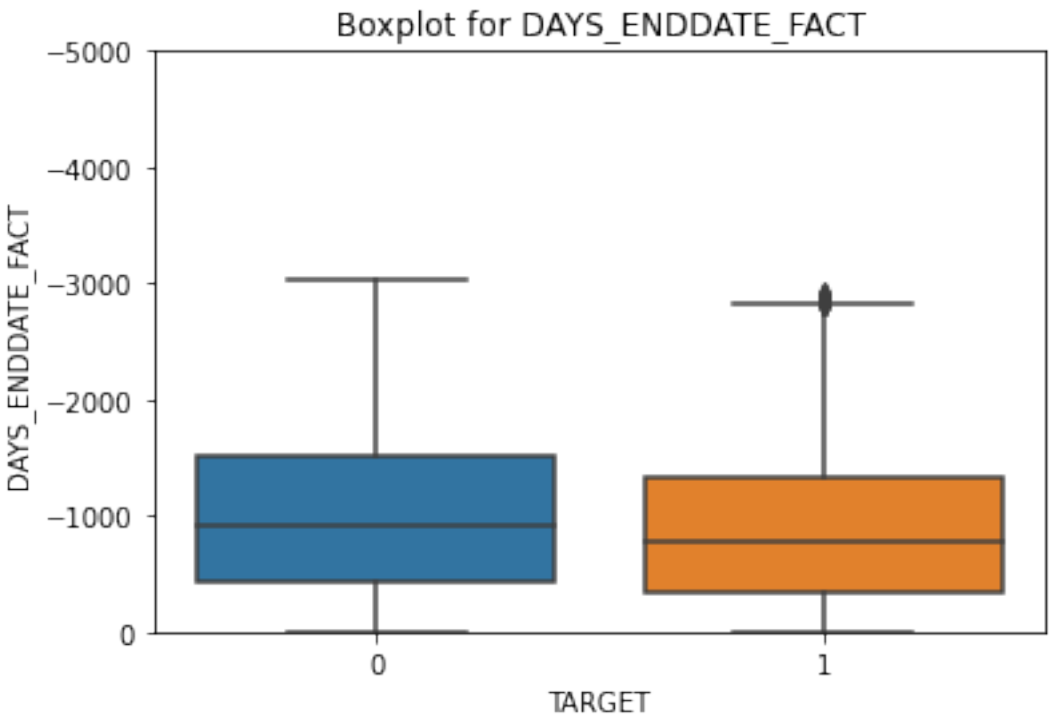
```
<matplotlib.legend.Legend at 0x1d148de43d0>
```




```
#we can see there aree most values are just around zero and very few
have more values ,that is also for non defaulters
```

```
sns.boxplot(x='TARGET',y='DAYS_ENDDATE_FACT',data=bureau_merged)
plt.ylim([0,-5000])
plt.title("Boxplot for DAYS_ENDDATE_FACT")
```

```
Text(0.5, 1.0, 'Boxplot for DAYS_ENDDATE_FACT')
```



- DAYS_ENDDATE_FACT distribution for defaulters looks lower than non defaulters ones

AMT_CREDIT_MAX_OVERDUE

Maximal amount overdue on the Credit Bureau credit so far (at application date of loan in our sample)

```
bureau_merged['AMT_CREDIT_MAX_OVERDUE'].describe()
```

```
count    5.167800e+05
mean     3.942852e+03
std      2.196856e+05
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      0.000000e+00
max      1.159872e+08
Name: AMT_CREDIT_MAX_OVERDUE, dtype: float64
```

```
#bureau_merged['AMT_CREDIT_MAX_OVERDUE'].value_counts()
```

```
bureau_merged['AMT_CREDIT_MAX_OVERDUE'].isna().sum()
```

```
992565
```

```
# total 992565 are null
```

```
bureau_merged['AMT_CREDIT_MAX_OVERDUE'].notnull().sum()
```

```
516780
```

```
bureau_merged[bureau_merged['AMT_CREDIT_MAX_OVERDUE'].isna()].isna().s
um()
```

```
SK_ID_CURR                     0
TARGET                         0
SK_ID_BUREAU               44020
CREDIT_ACTIVE              44020
CREDIT_CURRENCY            44020
DAYS_CREDIT                44020
CREDIT_DAY_OVERDUE         44020
DAYS_CREDIT_ENDDATE        84930
DAYS_ENDDATE_FACT         397482
AMT_CREDIT_MAX_OVERDUE    992565
CNT_CREDIT_PROLONG         44020
AMT_CREDIT_SUM             44023
AMT_CREDIT_SUM_DEBT       196569
AMT_CREDIT_SUM_LIMIT      460357
AMT_CREDIT_SUM_OVERDUE     44020
CREDIT_TYPE                44020
DAYS_CREDIT_UPDATE         44020
AMT_ANNUITY               749019
dtype: int64
```

```
bureau_merged[bureau_merged['AMT_CREDIT_MAX_OVERDUE'].isna()].head(20)
```

```
    SK_ID_CURR  TARGET  SK_ID_BUREAU CREDIT_ACTIVE CREDIT_CURRENCY  \
0       100002       1     6158904.0        Closed      currency 1
1       100002       1     6158905.0        Closed      currency 1
6       100002       1     6158903.0        Active      currency 1
12      100004       0     6829133.0        Closed      currency 1
14      100006       0           NaN           NaN             NaN
16      100008       0     6491433.0        Closed      currency 1
18      100008       0     6491432.0        Closed      currency 1
19      100009       0     5651008.0        Closed      currency 1
20      100009       0     5650996.0        Closed      currency 1
21      100009       0     5650997.0        Closed      currency 1
22      100009       0     5650998.0        Closed      currency 1
23      100009       0     5650999.0        Closed      currency 1
24      100009       0     5651000.0        Closed      currency 1
25      100009       0     5651002.0        Closed      currency 1
26      100009       0     5651003.0        Active      currency 1
27      100009       0     5651004.0        Active      currency 1
28      100009       0     5651005.0        Closed      currency 1
29      100009       0     5651006.0        Closed      currency 1
30      100009       0     5651007.0        Closed      currency 1
31      100009       0     5651009.0        Active      currency 1
```

```
    DAYS_CREDIT  CREDIT_DAY_OVERDUE  DAYS_CREDIT_ENDDATE
DAYS_ENDDATE_FACT  \
0       -1125.0                 0.0              -1038.0            -
1038.0
1        -476.0                 0.0                  NaN
-48.0
6       -1042.0                 0.0                780.0
NaN
12      -1326.0                 0.0               -595.0             -
683.0
14          NaN                 NaN                  NaN
NaN
16      -1097.0                 0.0               -853.0            -
1028.0
18      -1097.0                 0.0               -792.0             -
790.0
19      -1259.0                 0.0              -1080.0             -
986.0
20       -684.0                 0.0               -313.0             -
313.0
21       -782.0                 0.0               1044.0             -
415.0
22      -1428.0                 0.0              -1245.0            -
1244.0
23      -1241.0                 0.0              -1056.0            -
1055.0
24      -1686.0                 0.0                139.0             -
819.0
25      -1765.0                 0.0              -1581.0            -
1580.0
26       -239.0                 0.0               -209.0
NaN
27       -424.0                 0.0               1402.0
NaN
28      -2882.0                 0.0              -2152.0            -
2152.0
29      -1427.0                 0.0              -1245.0            -
1245.0
30      -1879.0                 0.0              -1699.0             -
986.0
31      -1293.0                 0.0                  NaN
NaN

    AMT_CREDIT_MAX_OVERDUE  CNT_CREDIT_PROLONG  AMT_CREDIT_SUM  \
0                      NaN                 0.0         40761.0
1                      NaN                 0.0             0.0
6                      NaN                 0.0        450000.0
12                     NaN                 0.0         94500.0
14                     NaN                 NaN             NaN
16                     NaN                 0.0        105705.0
```

18	NaN	0.0	95134.5
19	NaN	0.0	114799.5
20	NaN	0.0	61965.0
21	NaN	0.0	351000.0
22	NaN	0.0	93465.0
23	NaN	0.0	37710.0
24	NaN	0.0	1777500.0
25	NaN	0.0	66825.0
26	NaN	0.0	337500.0
27	NaN	0.0	642861.0
28	NaN	0.0	247500.0
29	NaN	0.0	93465.0
30	NaN	0.0	97978.5
31	NaN	0.0	187200.0

	AMT_CREDIT_SUM_DEBT	AMT_CREDIT_SUM_LIMIT	AMT_CREDIT_SUM_OVERDUE
\			
0	NaN	NaN	0.0
1	0.0	NaN	0.0
6	245781.0	0.0	0.0
12	0.0	0.0	0.0
14	NaN	NaN	NaN
16	0.0	0.0	0.0
18	0.0	0.0	0.0
19	NaN	NaN	0.0
20	0.0	0.0	0.0
21	0.0	0.0	0.0
22	0.0	0.0	0.0
23	0.0	0.0	0.0
24	0.0	0.0	0.0
25	0.0	0.0	0.0
26	326628.0	0.0	0.0
27	557959.5	0.0	0.0

```
28                  NaN                    NaN                     0.0

29                  0.0                    NaN                     0.0

30                  NaN                    NaN                     0.0

31                  0.0                    NaN                     0.0


        CREDIT_TYPE  DAYS_CREDIT_UPDATE  AMT_ANNUITY
0       Credit card             -1038.0          0.0
1       Credit card               -47.0          NaN
6   Consumer credit                -7.0          0.0
12  Consumer credit              -682.0          NaN
14              NaN                 NaN          NaN
16  Consumer credit             -1027.0          NaN
18  Consumer credit              -790.0          NaN
19  Consumer credit              -983.0          NaN
20  Consumer credit              -283.0          NaN
21  Consumer credit              -375.0          NaN
22  Consumer credit              -782.0          NaN
23  Consumer credit             -1055.0          NaN
24  Consumer credit              -591.0          NaN
25  Consumer credit             -1549.0          NaN
26      Credit card               -86.0          NaN
27  Consumer credit               -27.0          NaN
28  Consumer credit             -2152.0          NaN
29  Consumer credit             -1127.0          NaN
30  Consumer credit              -983.0          NaN
31      Credit card              -847.0          NaN
```

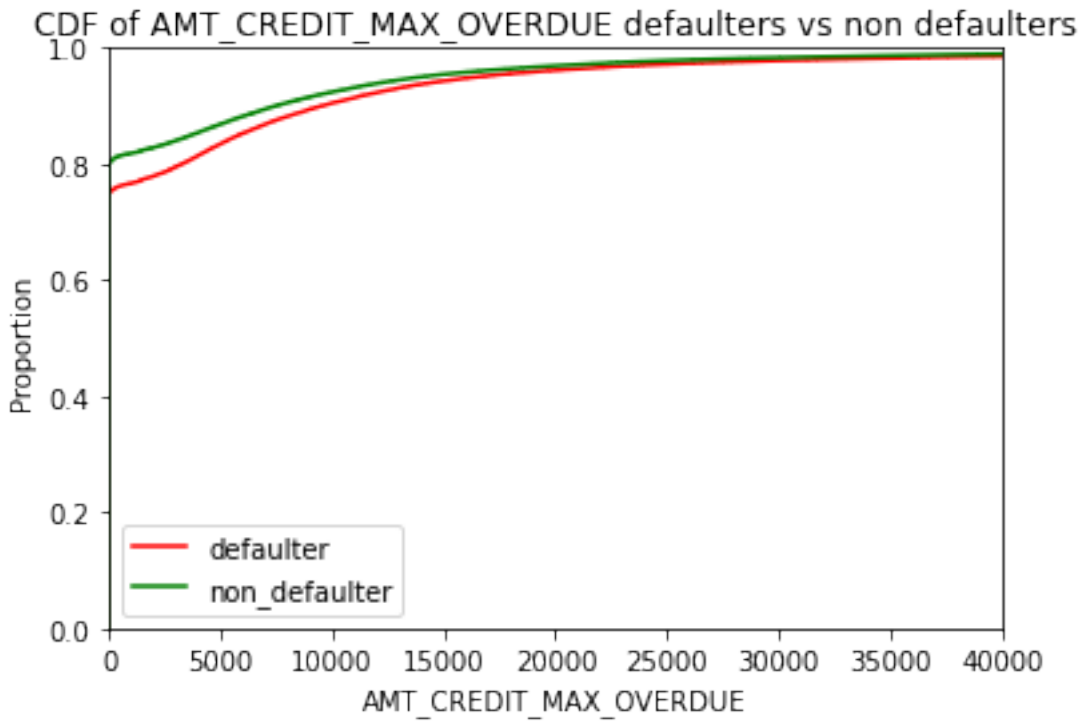
```
bureau_merged[bureau_merged['AMT_CREDIT_MAX_OVERDUE'].isna()]
['CREDIT_ACTIVE'].value_counts()
```

```
Closed      594933
Active      349325
Sold          4281
Bad debt         6
Name: CREDIT_ACTIVE, dtype: int64
```

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==1]
['AMT_CREDIT_MAX_OVERDUE'],label='defaulter',color='red')
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==0]
['AMT_CREDIT_MAX_OVERDUE'],label='non_defaulter',color='green')
plt.xlim([0,40000])
plt.title("CDF of AMT_CREDIT_MAX_OVERDUE defaulters vs non
defaulters")
```

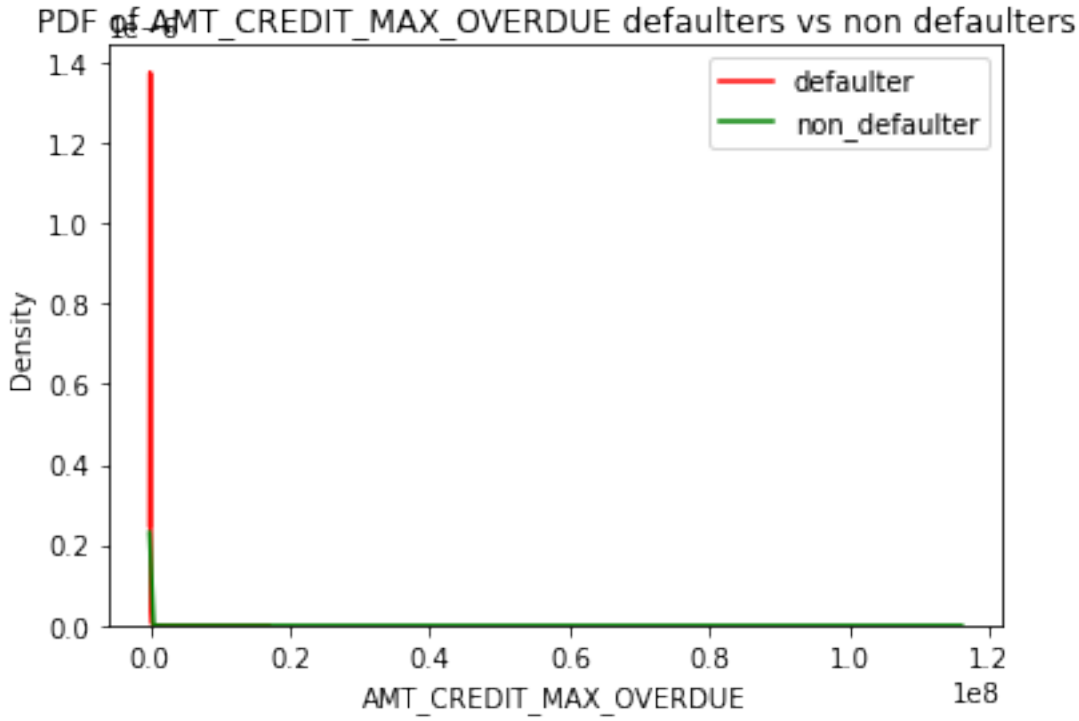
```
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d1b9285cd0>
```



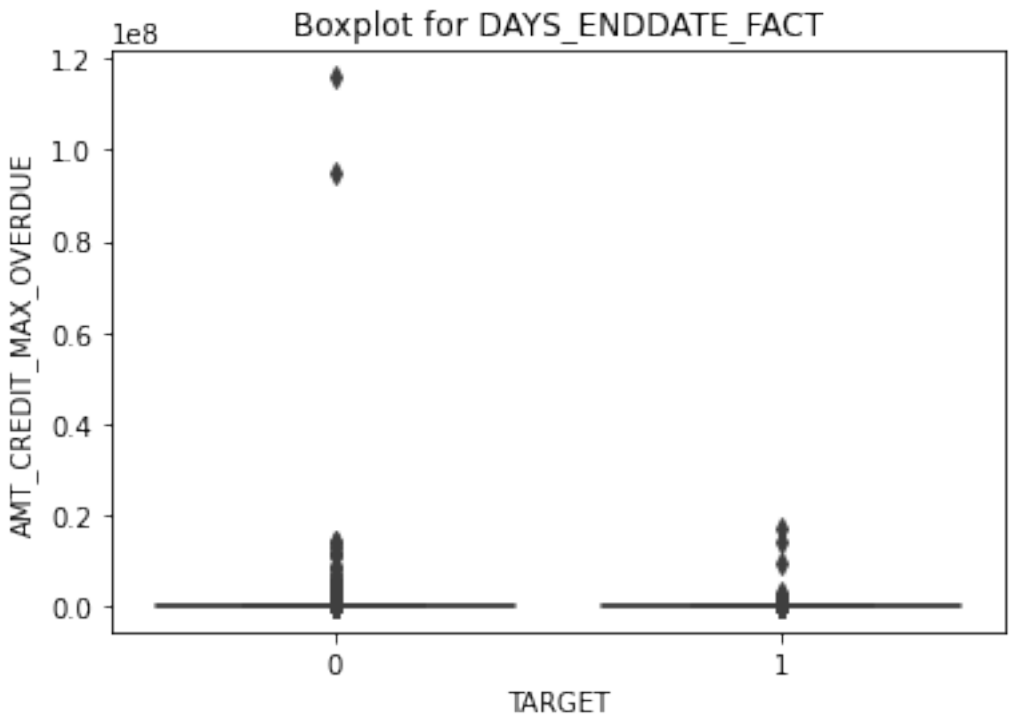

```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==1]
['AMT_CREDIT_MAX_OVERDUE'],label='defaulter',color='red')
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==0]
['AMT_CREDIT_MAX_OVERDUE'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of AMT_CREDIT_MAX_OVERDUE defaulters vs non
defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d19eeec040>
```




```
#we can see there aree most values are just around zero and very few
have more values ,that is also for non defaulters

sns.boxplot(x='TARGET',y='AMT_CREDIT_MAX_OVERDUE',data=bureau_merged)
#plt.ylim([0.0,50000])
plt.title("Boxplot for DAYS_ENDDATE_FACT")
```

```
Text(0.5, 1.0, 'Boxplot for DAYS_ENDDATE_FACT')
```



- AMT_CREDIT_MAX_OVERDUE looks similar distribution. For non defaulters some big outliers

AMT_CREDIT_SUM_OVERDUE

Current amount overdue on Credit Bureau credit

```
bureau_merged['AMT_CREDIT_SUM_OVERDUE'].describe()
```

```
count    1.465325e+06
mean     4.010547e+01
std      6.273639e+03
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      0.000000e+00
max      3.756681e+06
Name: AMT_CREDIT_SUM_OVERDUE, dtype: float64
```

```
#bureau_merged['AMT_CREDIT_SUM_OVERDUE'].value_counts()
```

```
bureau_merged['AMT_CREDIT_SUM_OVERDUE'].isna().sum()
```

```
44020
```

```
# total 44020 are null
```

```
bureau_merged['AMT_CREDIT_SUM_OVERDUE'].notnull().sum()
```

```
516780
```

```
bureau_merged[bureau_merged['AMT_CREDIT_SUM_OVERDUE'].isna()].isna().s
um()
```

```
SK_ID_CURR                   0
TARGET                       0
SK_ID_BUREAU             44020
CREDIT_ACTIVE            44020
CREDIT_CURRENCY          44020
DAYS_CREDIT              44020
CREDIT_DAY_OVERDUE       44020
DAYS_CREDIT_ENDDATE      44020
DAYS_ENDDATE_FACT        44020
AMT_CREDIT_MAX_OVERDUE   44020
CNT_CREDIT_PROLONG       44020
AMT_CREDIT_SUM           44020
AMT_CREDIT_SUM_DEBT      44020
AMT_CREDIT_SUM_LIMIT     44020
AMT_CREDIT_SUM_OVERDUE   44020
CREDIT_TYPE              44020
DAYS_CREDIT_UPDATE       44020
AMT_ANNUITY              44020
dtype: int64
```

```
bureau_merged[bureau_merged['AMT_CREDIT_SUM_OVERDUE'].isna()].head(20)
```

```
     SK_ID_CURR  TARGET  SK_ID_BUREAU CREDIT_ACTIVE CREDIT_CURRENCY  \
14       100006       0           NaN           NaN             NaN
43       100012       0           NaN           NaN             NaN
69       100018       0           NaN           NaN             NaN
76       100021       0           NaN           NaN             NaN
92       100024       0           NaN           NaN             NaN
122      100034       0           NaN           NaN             NaN
150      100040       0           NaN           NaN             NaN
200      100051       0           NaN           NaN             NaN
201      100052       0           NaN           NaN             NaN
219      100056       0           NaN           NaN             NaN
298      100081       0           NaN           NaN             NaN
346      100095       0           NaN           NaN             NaN
351      100098       0           NaN           NaN             NaN
367      100101       0           NaN           NaN             NaN
426      100118       0           NaN           NaN             NaN
456      100127       0           NaN           NaN             NaN
462      100133       0           NaN           NaN             NaN
470      100137       0           NaN           NaN             NaN
486      100140       0           NaN           NaN             NaN
511      100144       0           NaN           NaN             NaN

     DAYS_CREDIT  CREDIT_DAY_OVERDUE  DAYS_CREDIT_ENDDATE
DAYS_ENDDATE_FACT  \
```

```
14           NaN                  NaN                  NaN
NaN
43           NaN                  NaN                  NaN
NaN
69           NaN                  NaN                  NaN
NaN
76           NaN                  NaN                  NaN
NaN
92           NaN                  NaN                  NaN
NaN
122          NaN                  NaN                  NaN
NaN
150          NaN                  NaN                  NaN
NaN
200          NaN                  NaN                  NaN
NaN
201          NaN                  NaN                  NaN
NaN
219          NaN                  NaN                  NaN
NaN
298          NaN                  NaN                  NaN
NaN
346          NaN                  NaN                  NaN
NaN
351          NaN                  NaN                  NaN
NaN
367          NaN                  NaN                  NaN
NaN
426          NaN                  NaN                  NaN
NaN
456          NaN                  NaN                  NaN
NaN
462          NaN                  NaN                  NaN
NaN
470          NaN                  NaN                  NaN
NaN
486          NaN                  NaN                  NaN
NaN
511          NaN                  NaN                  NaN
NaN

     AMT_CREDIT_MAX_OVERDUE  CNT_CREDIT_PROLONG  AMT_CREDIT_SUM  \
14                      NaN                 NaN             NaN
43                      NaN                 NaN             NaN
69                      NaN                 NaN             NaN
76                      NaN                 NaN             NaN
92                      NaN                 NaN             NaN
122                     NaN                 NaN             NaN
150                     NaN                 NaN             NaN
200                     NaN                 NaN             NaN
```

201	NaN	NaN	NaN
219	NaN	NaN	NaN
298	NaN	NaN	NaN
346	NaN	NaN	NaN
351	NaN	NaN	NaN
367	NaN	NaN	NaN
426	NaN	NaN	NaN
456	NaN	NaN	NaN
462	NaN	NaN	NaN
470	NaN	NaN	NaN
486	NaN	NaN	NaN
511	NaN	NaN	NaN

	AMT_CREDIT_SUM_DEBT	AMT_CREDIT_SUM_LIMIT	AMT_CREDIT_SUM_OVERDUE
\			
14	NaN	NaN	NaN
43	NaN	NaN	NaN
69	NaN	NaN	NaN
76	NaN	NaN	NaN
92	NaN	NaN	NaN
122	NaN	NaN	NaN
150	NaN	NaN	NaN
200	NaN	NaN	NaN
201	NaN	NaN	NaN
219	NaN	NaN	NaN
298	NaN	NaN	NaN
346	NaN	NaN	NaN
351	NaN	NaN	NaN
367	NaN	NaN	NaN
426	NaN	NaN	NaN
456	NaN	NaN	NaN
462	NaN	NaN	NaN

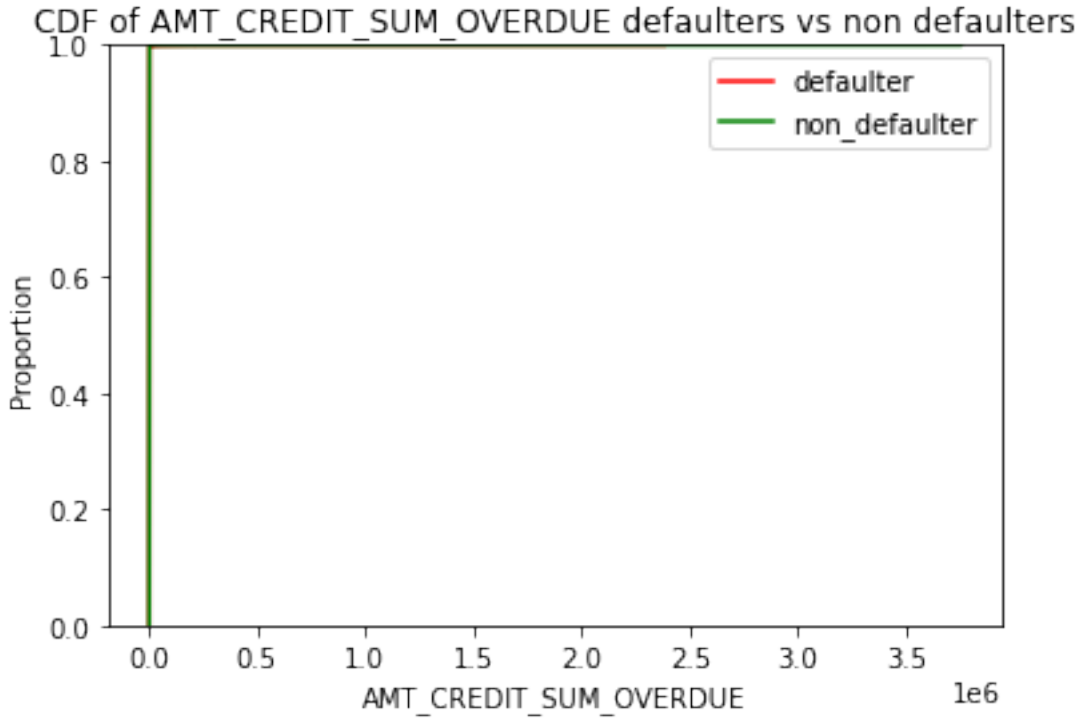
```
470                   NaN                     NaN                       NaN

486                   NaN                     NaN                       NaN

511                   NaN                     NaN                       NaN


    CREDIT_TYPE  DAYS_CREDIT_UPDATE  AMT_ANNUITY
14          NaN                 NaN          NaN
43          NaN                 NaN          NaN
69          NaN                 NaN          NaN
76          NaN                 NaN          NaN
92          NaN                 NaN          NaN
122         NaN                 NaN          NaN
150         NaN                 NaN          NaN
200         NaN                 NaN          NaN
201         NaN                 NaN          NaN
219         NaN                 NaN          NaN
298         NaN                 NaN          NaN
346         NaN                 NaN          NaN
351         NaN                 NaN          NaN
367         NaN                 NaN          NaN
426         NaN                 NaN          NaN
456         NaN                 NaN          NaN
462         NaN                 NaN          NaN
470         NaN                 NaN          NaN
486         NaN                 NaN          NaN
511         NaN                 NaN          NaN
```

```
bureau_merged[bureau_merged['AMT_CREDIT_SUM_OVERDUE'].isna()]
['CREDIT_ACTIVE'].value_counts()
```

```
Series([], Name: CREDIT_ACTIVE, dtype: int64)
```

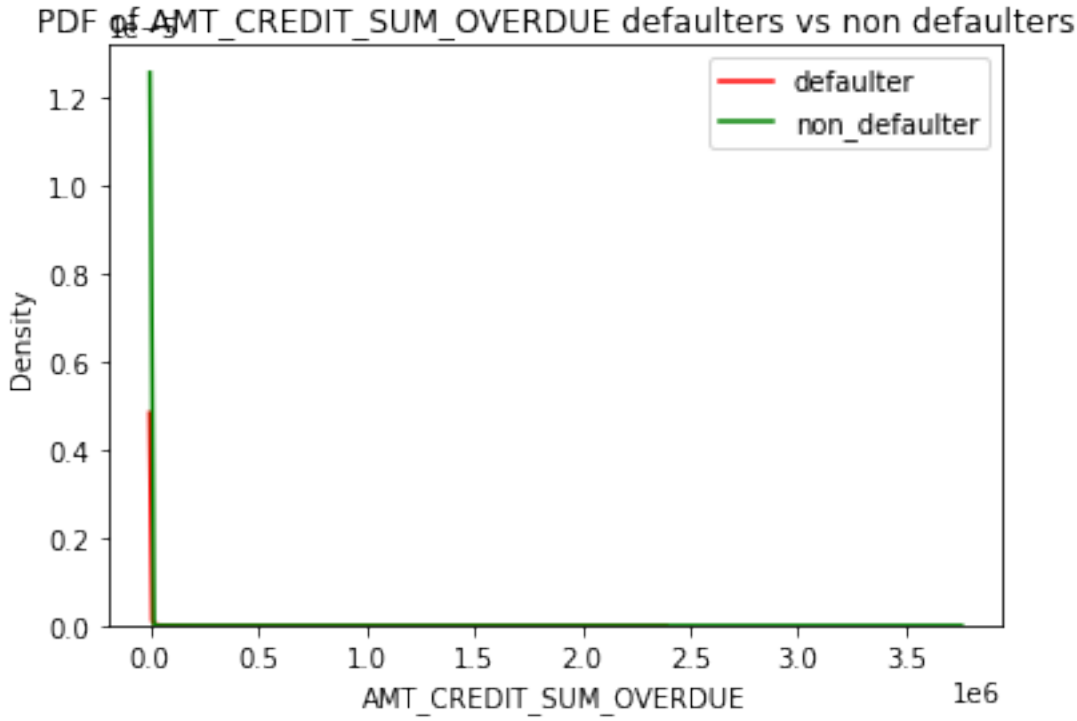
```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==1]
['AMT_CREDIT_SUM_OVERDUE'],label='defaulter',color='red')
sns.ecdfplot(bureau_merged[bureau_merged['TARGET']==0]
['AMT_CREDIT_SUM_OVERDUE'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of AMT_CREDIT_SUM_OVERDUE defaulters vs non
defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d1c50e4460>
```



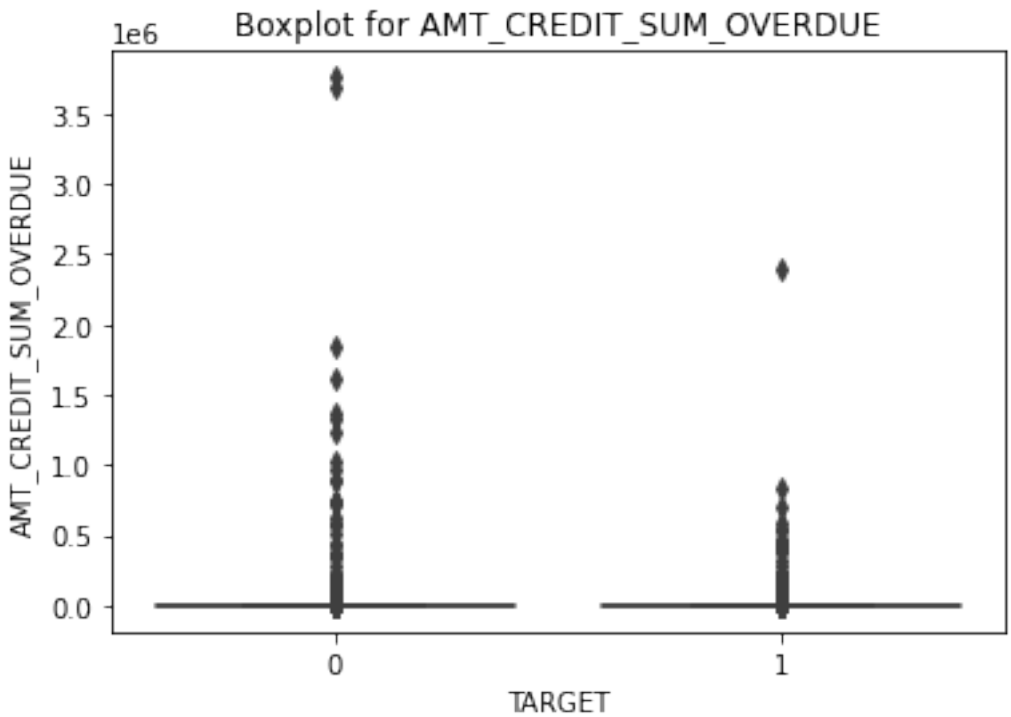

```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==1]
['AMT_CREDIT_SUM_OVERDUE'],label='defaulter',color='red')
sns.kdeplot(bureau_merged[bureau_merged['TARGET']==0]
['AMT_CREDIT_SUM_OVERDUE'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of AMT_CREDIT_SUM_OVERDUE defaulters vs non
defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d209e40520>
```




```
#we can see there aree most values are just around zero and very few
have more values ,that is also for non defaulters

sns.boxplot(x='TARGET',y='AMT_CREDIT_SUM_OVERDUE',data=bureau_merged)
#plt.ylim([0.0,50000])
plt.title("Boxplot for AMT_CREDIT_SUM_OVERDUE")
```

```
Text(0.5, 1.0, 'Boxplot for AMT_CREDIT_SUM_OVERDUE')
```



- AMT_CREDIT_SUM_OVERDUE looks similar distribution most values are zeros for both categories with more outliers for non defaulters

bureau_balance.csv

This table consists of Monthly balance of each credit for each of the previous credit that the client had with financial institutions other than Home Cre

```
bureau_balance.head()
```

```
   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
```

```
bureau_balance.shape
```

```
(27299925, 3)
```

```
#Checking NULL values
bureau_balance.isna().sum()
```

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

```
col_descriptions[col_descriptions['Table']=='bureau_balance.csv']
```

```
                   Table             Row  \
142  bureau_balance.csv    SK_BUREAU_ID
143  bureau_balance.csv  MONTHS_BALANCE
144  bureau_balance.csv          STATUS


Description  \
142
Recoded ID of Credit Bureau credit (unique coding for each
application) - use this to join to CREDIT_BUREAU table
143
Month of balance relative to application date (-1 means the freshest
balance date)
144  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,  5
means DPD 120+ or sold or written off ] )

                                  Special
142                                hashed
143  time only relative to the application
144                                    NaN
```

SK_ID_BUREAU

Recoded ID of Credit Bureau credit (unique coding for each application) - use this to join to CREDIT_BUREAU table

```
bureau_balance['SK_ID_BUREAU'].nunique()
```

```
817395
```

```
#So we have 27299925 total rows and 817395 unique SK_ID_BUREAU
```

```
['STATUS']
```

```
bureau_balance_merged=bureau_merged.iloc[:,:3].merge(bureau_balance,
on= 'SK_ID_BUREAU', how='left')
```

MONTHS_BALANCE

Month of balance relative to application date (-1 means the freshest balance date)

```
bureau_balance_merged['MONTHS_BALANCE'].describe()
```

```
count    1.470161e+07
mean    -2.863028e+01
std      2.206971e+01
min     -9.600000e+01
25%     -4.300000e+01
50%     -2.400000e+01
75%     -1.000000e+01
max      0.000000e+00
Name: MONTHS_BALANCE, dtype: float64
```

```
#bureau_balance_merged['MONTHS_BALANCE'].value_counts()
```

- here min shows the data is provided for 960 months ,Means 8 years

```
bureau_balance_merged['MONTHS_BALANCE'].isna().sum()
```

```
985830
```

```
# total 985830 are null
```

```
bureau_balance_merged['MONTHS_BALANCE'].notnull().sum()
```

```
14701612
```

```
bureau_balance_merged[bureau_balance_merged['MONTHS_BALANCE'].isna()].
isna().sum()
```

```
SK_ID_CURR             0
TARGET                 0
SK_ID_BUREAU       44020
MONTHS_BALANCE    985830
STATUS            985830
dtype: int64
```

```
bureau_balance_merged[bureau_balance_merged['MONTHS_BALANCE'].isna()].
head(20)
```

```
     SK_ID_CURR  TARGET  SK_ID_BUREAU  MONTHS_BALANCE STATUS
110      100003       0     5885877.0             NaN    NaN
111      100003       0     5885878.0             NaN    NaN
112      100003       0     5885879.0             NaN    NaN
113      100003       0     5885880.0             NaN    NaN
114      100004       0     6829133.0             NaN    NaN
115      100004       0     6829134.0             NaN    NaN
116      100006       0           NaN             NaN    NaN
117      100007       0     5987200.0             NaN    NaN
118      100008       0     6491433.0             NaN    NaN
119      100008       0     6491434.0             NaN    NaN
120      100008       0     6491432.0             NaN    NaN
121      100009       0     5651008.0             NaN    NaN
122      100009       0     5650996.0             NaN    NaN
123      100009       0     5650997.0             NaN    NaN
124      100009       0     5650998.0             NaN    NaN
```

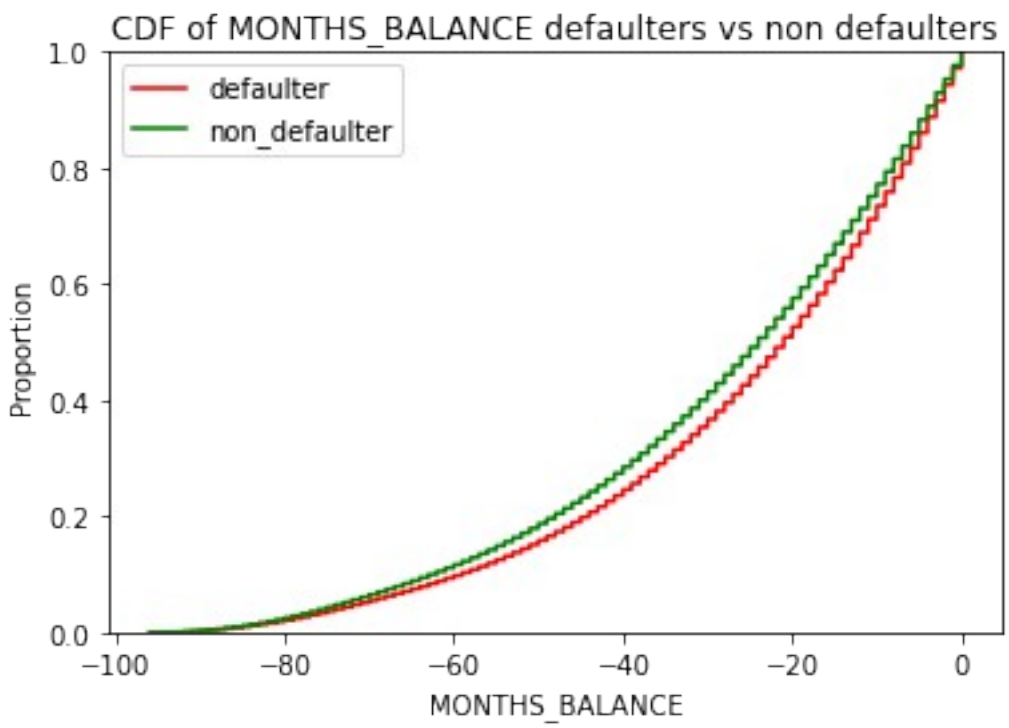
```
125      100009       0     5650999.0             NaN    NaN
126      100009       0     5651000.0             NaN    NaN
127      100009       0     5651002.0             NaN    NaN
128      100009       0     5651003.0             NaN    NaN
129      100009       0     5651004.0             NaN    NaN
```

```
bureau_balance_merged[bureau_balance_merged['MONTHS_BALANCE'].isna()]
['TARGET'].value_counts()
```

```
0    910164
1     75666
Name: TARGET, dtype: int64
```

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(bureau_balance_merged[bureau_balance_merged['TARGET']==1]
['MONTHS_BALANCE'],label='defaulter',color='red')
sns.ecdfplot(bureau_balance_merged[bureau_balance_merged['TARGET']==0]
['MONTHS_BALANCE'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of MONTHS_BALANCE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

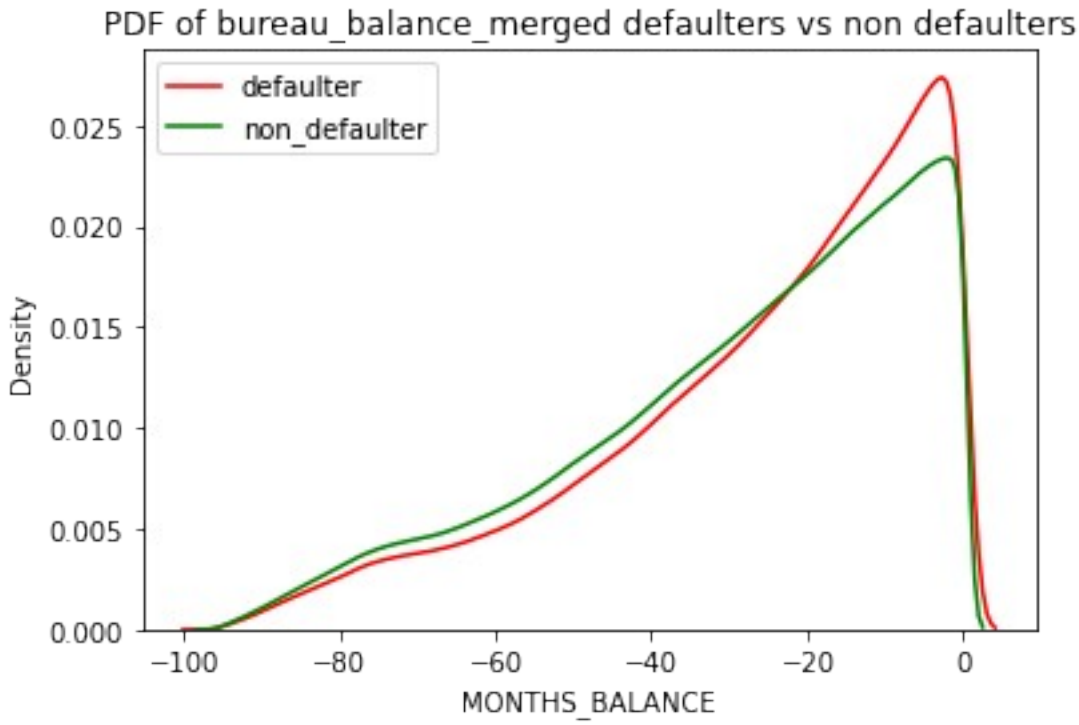
```
<matplotlib.legend.Legend at 0x1d0800ea340>
```




```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(bureau_balance_merged[bureau_balance_merged['TARGET']==1]
['MONTHS_BALANCE'],label='defaulter',color='red')
sns.kdeplot(bureau_balance_merged[bureau_balance_merged['TARGET']==0]
```

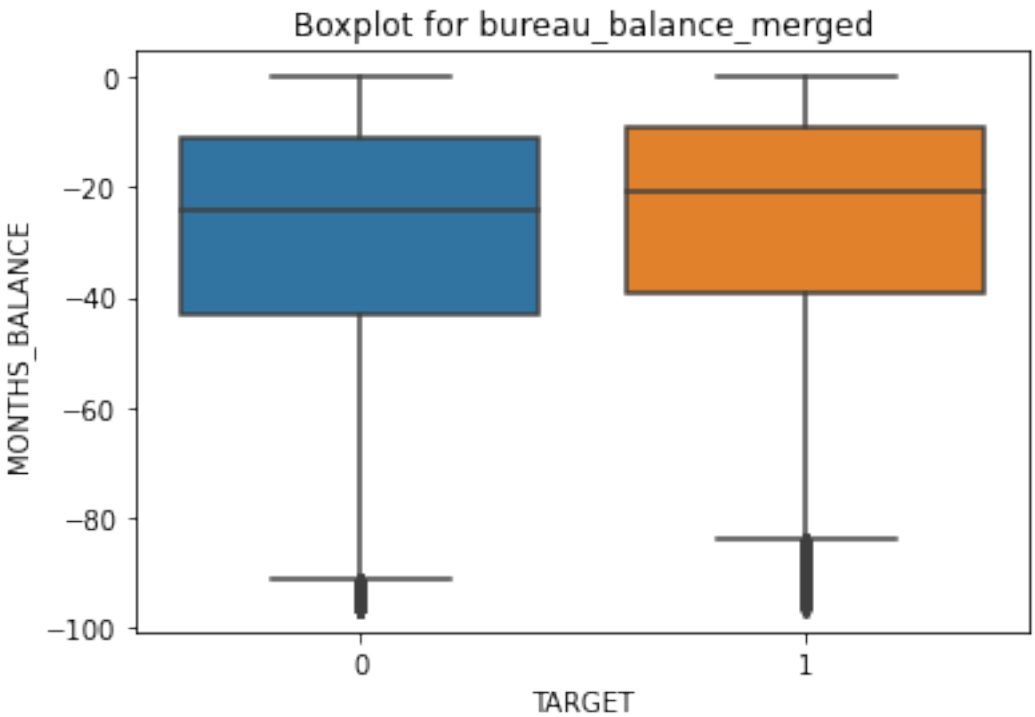
```
['MONTHS_BALANCE'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of bureau_balance_merged defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d1dc8f2940>
```




```
sns.boxplot(x='TARGET',y='MONTHS_BALANCE',data=bureau_balance_merged)
#plt.ylim([0.0,50000])
plt.title("Boxplot for bureau_balance_merged")
```

```
Text(0.5, 1.0, 'Boxplot for bureau_balance_merged')
```



- we can see there both TAREGT category have same distribution For MONTHLY BALANCE,with Defaulters values slight more tha non defaulters ones when the value is less than zero

STATUS

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,
- 5 means DPD 120+ or sold or written off]
- AMT_CREDIT_SUM_OVERDUE looks similar distribution most values are zeros for both categories with more outliers for non defaulters

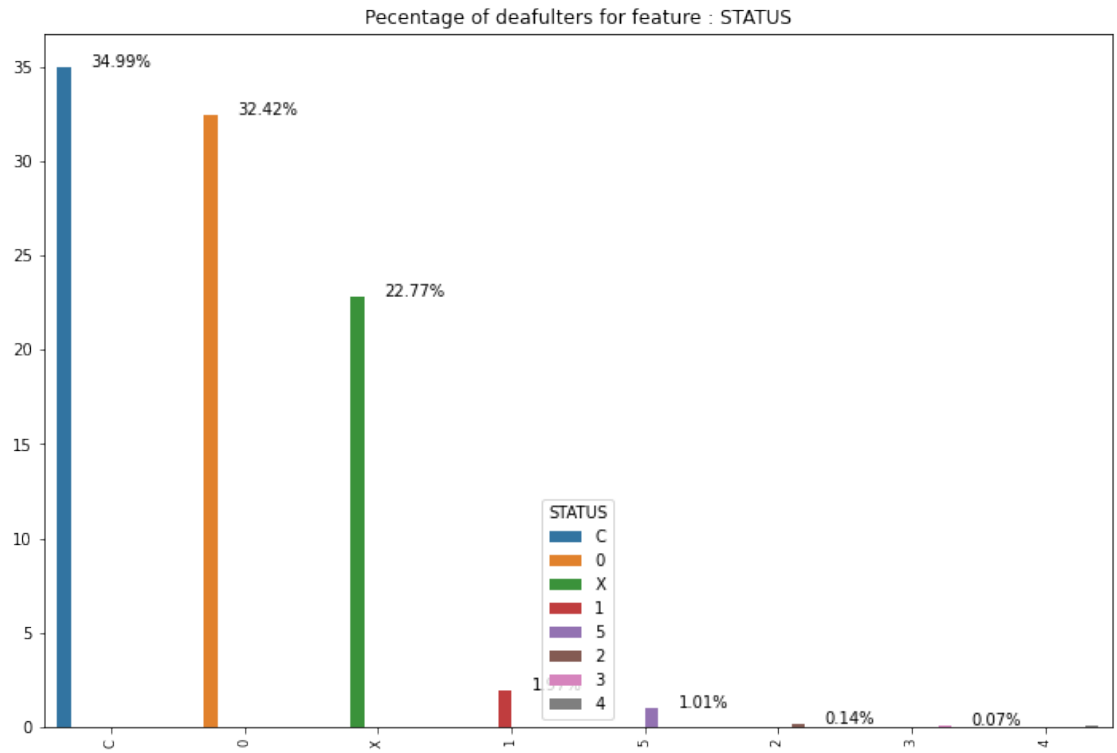
```
bureau_balance_merged['STATUS'].value_counts()
```

```
C    7027575
0    4615684
X    2837039
1     155330
5      40528
2      15583
```

```
3       5976
4       3897
Name: STATUS, dtype: int64
```

```
plot_for_categorical_defaulters(bureau_balance_merged,'STATUS')
```

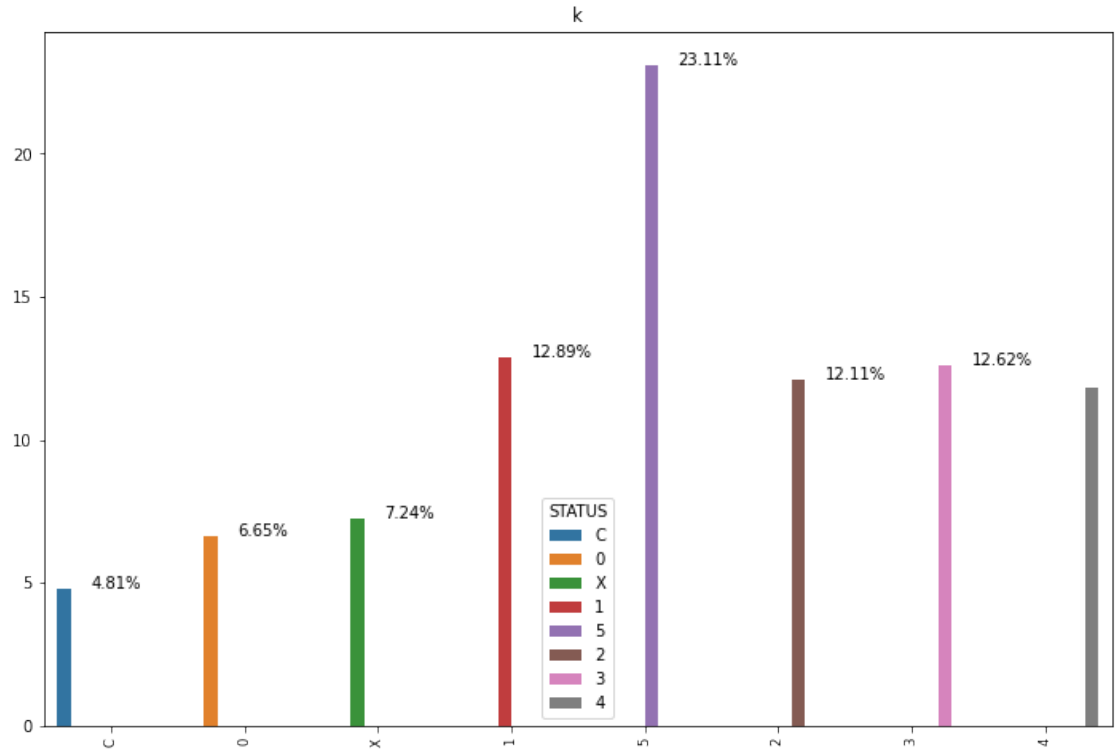
```
{'C': 34.99436181066423,
 '0': 32.41690382859905,
 'X': 22.767545508242495,
 '1': 1.9706814154540084,
 '5': 1.014874080438168,
 '2': 0.14498201149116682,
 '3': 0.06980615368093218,
 '4': 0.04832733716372228}
```




```
# we can see % of defaulters in each category is in same propotion of
the proption of category in the data tself
```

```
plot_for_deafulters_categorywise(bureau_balance_merged,'STATUS')
plt.title("% of defaulters in each category")
```

```
Text(0.5, 1.0, 'k')
```



- Here category 5 shows max % of deaulters in its own category, as 5 means DPD 120+ or sold or written off. means its been more than 4 month that overdue is there or loan is already marked as faulty loan

```
del bureau_balance_merged
```

previous_application.csv

- This table contains client applications data for previous loan in Home credit, contains info about previous loan parameters and client info at time of previous application
- One row per previous application

```
previous_application.head()
```

```
   SK_ID_PREV  SK_ID_CURR NAME_CONTRACT_TYPE  AMT_ANNUITY
AMT_APPLICATION  \
0     2030495      271877     Consumer loans     1730.430
17145.0
1     2802425      108129         Cash loans    25188.615
607500.0
2     2523466      122040         Cash loans    15060.735
112500.0
3     2819243      176158         Cash loans    47041.335
450000.0
4     1784265      202054         Cash loans    31924.395
337500.0
```

```
   AMT_CREDIT  AMT_DOWN_PAYMENT  AMT_GOODS_PRICE
WEEKDAY_APPR_PROCESS_START  \
0     17145.0               0.0          17145.0
SATURDAY
1    679671.0               NaN         607500.0
THURSDAY
2    136444.5               NaN         112500.0
TUESDAY
3    470790.0               NaN         450000.0
MONDAY
4    404055.0               NaN         337500.0
THURSDAY

   HOUR_APPR_PROCESS_START FLAG_LAST_APPL_PER_CONTRACT  \
0                       15                           Y
1                       11                           Y
2                       11                           Y
3                        7                           Y
4                        9                           Y

   NFLAG_LAST_APPL_IN_DAY  RATE_DOWN_PAYMENT  RATE_INTEREST_PRIMARY  \
0                       1                0.0               0.182832
1                       1                NaN                    NaN
2                       1                NaN                    NaN
3                       1                NaN                    NaN
4                       1                NaN                    NaN

   RATE_INTEREST_PRIVILEGED NAME_CASH_LOAN_PURPOSE
NAME_CONTRACT_STATUS  \
0                  0.867336                    XAP
Approved
1                       NaN                    XNA
Approved
2                       NaN                    XNA
Approved
3                       NaN                    XNA
Approved
4                       NaN                Repairs
Refused

   DAYS_DECISION      NAME_PAYMENT_TYPE CODE_REJECT_REASON
NAME_TYPE_SUITE  \
0            -73  Cash through the bank                XAP
NaN
1           -164                    XNA                XAP
Unaccompanied
2           -301  Cash through the bank                XAP  Spouse,
partner
3           -512  Cash through the bank                XAP
```

```
NaN
4           -781  Cash through the bank                 HC
NaN

  NAME_CLIENT_TYPE NAME_GOODS_CATEGORY NAME_PORTFOLIO
NAME_PRODUCT_TYPE  \
0         Repeater              Mobile            POS
XNA
1         Repeater                 XNA           Cash            x-
sell
2         Repeater                 XNA           Cash            x-
sell
3         Repeater                 XNA           Cash            x-
sell
4         Repeater                 XNA           Cash           walk-
in

              CHANNEL_TYPE  SELLERPLACE_AREA NAME_SELLER_INDUSTRY  \
0             Country-wide                35         Connectivity
1           Contact center                -1                  XNA
2  Credit and cash offices                -1                  XNA
3  Credit and cash offices                -1                  XNA
4  Credit and cash offices                -1                  XNA

   CNT_PAYMENT NAME_YIELD_GROUP       PRODUCT_COMBINATION
DAYS_FIRST_DRAWING  \
0         12.0           middle  POS mobile with interest
365243.0
1         36.0       low_action          Cash X-Sell: low
365243.0
2         12.0             high         Cash X-Sell: high
365243.0
3         12.0           middle       Cash X-Sell: middle
365243.0
4         24.0             high         Cash Street: high
NaN

   DAYS_FIRST_DUE  DAYS_LAST_DUE_1ST_VERSION  DAYS_LAST_DUE
DAYS_TERMINATION  \
0           -42.0                      300.0          -42.0
-37.0
1          -134.0                      916.0       365243.0
365243.0
2          -271.0                       59.0       365243.0
365243.0
3          -482.0                     -152.0         -182.0
-177.0
4             NaN                        NaN            NaN
NaN
```

```
   NFLAG_INSURED_ON_APPROVAL
0                        0.0
1                        1.0
2                        1.0
3                        1.0
4                        NaN
```

```
col_descriptions[col_descriptions['Table']=='previous_application.csv'
]
```

```
                        Table                          Row  \
176  previous_application.csv                 SK_ID_PREV
177  previous_application.csv                  SK_ID_CURR
178  previous_application.csv          NAME_CONTRACT_TYPE
179  previous_application.csv                 AMT_ANNUITY
180  previous_application.csv             AMT_APPLICATION
181  previous_application.csv                  AMT_CREDIT
182  previous_application.csv            AMT_DOWN_PAYMENT
183  previous_application.csv             AMT_GOODS_PRICE
184  previous_application.csv   WEEKDAY_APPR_PROCESS_START
185  previous_application.csv      HOUR_APPR_PROCESS_START
186  previous_application.csv  FLAG_LAST_APPL_PER_CONTRACT
187  previous_application.csv       NFLAG_LAST_APPL_IN_DAY
188  previous_application.csv            NFLAG_MICRO_CASH
189  previous_application.csv           RATE_DOWN_PAYMENT
190  previous_application.csv        RATE_INTEREST_PRIMARY
191  previous_application.csv     RATE_INTEREST_PRIVILEGED
192  previous_application.csv       NAME_CASH_LOAN_PURPOSE
193  previous_application.csv         NAME_CONTRACT_STATUS
194  previous_application.csv                DAYS_DECISION
195  previous_application.csv           NAME_PAYMENT_TYPE
196  previous_application.csv          CODE_REJECT_REASON
197  previous_application.csv             NAME_TYPE_SUITE
198  previous_application.csv            NAME_CLIENT_TYPE
199  previous_application.csv         NAME_GOODS_CATEGORY
200  previous_application.csv              NAME_PORTFOLIO
201  previous_application.csv           NAME_PRODUCT_TYPE
202  previous_application.csv                CHANNEL_TYPE
203  previous_application.csv            SELLERPLACE_AREA
204  previous_application.csv         NAME_SELLER_INDUSTRY
205  previous_application.csv                  CNT_PAYMENT
206  previous_application.csv            NAME_YIELD_GROUP
207  previous_application.csv         PRODUCT_COMBINATION
208  previous_application.csv          DAYS_FIRST_DRAWING
209  previous_application.csv              DAYS_FIRST_DUE
210  previous_application.csv    DAYS_LAST_DUE_1ST_VERSION
211  previous_application.csv               DAYS_LAST_DUE
212  previous_application.csv            DAYS_TERMINATION
213  previous_application.csv    NFLAG_INSURED_ON_APPROVAL
```

Description \
176 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)
177
ID of loan in our sample
178
Contract product type (Cash loan, consumer loan [POS] ,...) of the previous application
179
Annuity of previous application
180
For how much credit did client ask on the previous application
181 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
182
Down payment on the previous application
183
Goods price of good that client asked for (if applicable) on the previous application
184
On which day of the week did the client apply for previous application

185
Approximately at what day hour did the client apply for the previous application
186
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
187 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
188
Flag Micro finance loan
189
Down payment rate normalized on previous credit
190
Interest rate normalized on previous credit
191
Interest rate normalized on previous credit
192
Purpose of the cash loan
193
Contract status (approved, cancelled, ...) of previous application

194
Relative to current application when was the decision about previous application made
195
Payment method that client chose to pay for the previous application

196
Why was the previous application rejected
197
Who accompanied client when applying for the previous application
198
Was the client old or new client when applying for the previous application
199
What kind of goods did the client apply for in the previous application
200
Was the previous application for CASH, POS, CAR, ⍰
201
Was the previous application x-sell o walk-in
202
Through which channel we acquired the client on the previous application
203
Selling area of seller place of the previous application
204
The industry of the seller
205
Term of previous credit at application of the previous application
206
Grouped interest rate into small medium and high of the previous application
207
Detailed product combination of the previous application
208
Relative to application date of current application when was the first disbursement of the previous application
209
Relative to application date of current application when was the first due supposed to be of the previous application
210
Relative to application date of current application when was the first due of the previous application
211
Relative to application date of current application when was the last due date of the previous application
212
Relative to application date of current application when was the expected termination of the previous application
213

```
Did the client requested insurance during the previous application

                                    Special
176                                  hashed
177                                  hashed
178                                     NaN
179                                     NaN
180                                     NaN
181                                     NaN
182                                     NaN
183                                     NaN
184                                     NaN
185                                 rounded
186                                     NaN
187                                     NaN
188                                     NaN
189                              normalized
190                              normalized
191                              normalized
192                                     NaN
193                                     NaN
194  time only relative to the application
195                                     NaN
196                                     NaN
197                                     NaN
198                                     NaN
199                                     NaN
200                                     NaN
201                                     NaN
202                                     NaN
203                                     NaN
204                                     NaN
205                                     NaN
206                                 grouped
207                                     NaN
208  time only relative to the application
209  time only relative to the application
210  time only relative to the application
211  time only relative to the application
212  time only relative to the application
213                                     NaN
```

```
previous_application.shape
```

```
(1670214, 37)
```

```
#Total 1.6 M rows and total 37 columns

previous_application_cat_cols=previous_application.select_dtypes(inclu
de=[object]).columns.to_list()
```

```
len(previous_application_cat_cols)
```

```
16
```

- There are total 16 categorical columns out of 37 in previous_application table

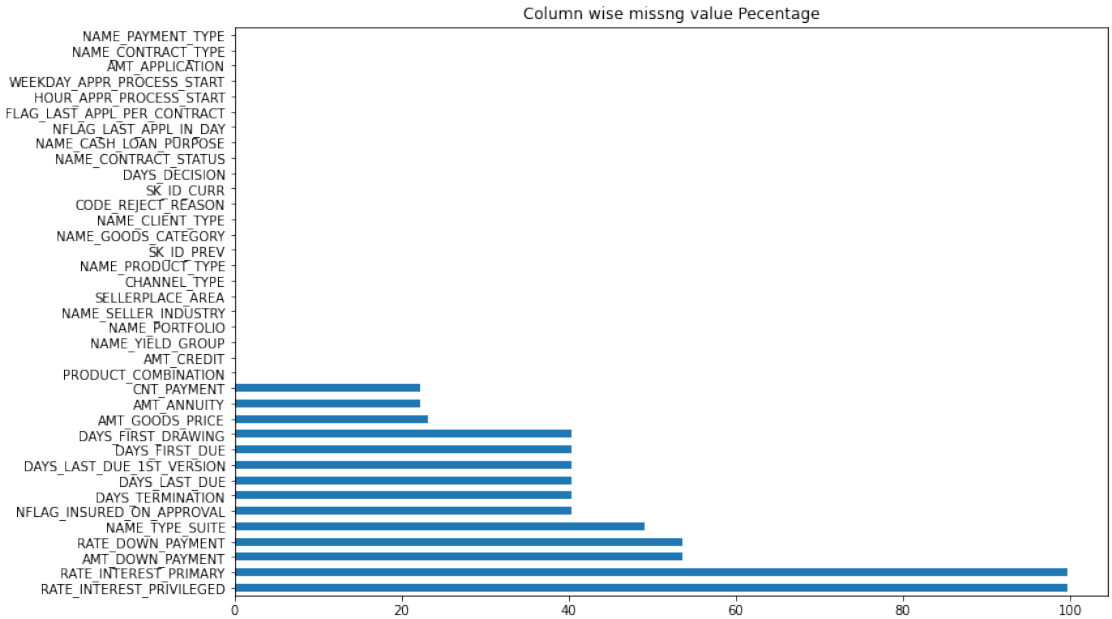
```
percent_missing = previous_application.isnull().sum() * 100 /
len(previous_application)
```

```
percent_missing.sort_values(ascending=False)[:30]
```

```
COMMONAREA_MEDI             69.872297
COMMONAREA_AVG              69.872297
COMMONAREA_MODE             69.872297
NONLIVINGAPARTMENTS_MODE    69.432963
NONLIVINGAPARTMENTS_AVG     69.432963
NONLIVINGAPARTMENTS_MEDI    69.432963
FONDKAPREMONT_MODE          68.386172
LIVINGAPARTMENTS_MODE       68.354953
LIVINGAPARTMENTS_AVG        68.354953
LIVINGAPARTMENTS_MEDI       68.354953
FLOORSMIN_AVG               67.848630
FLOORSMIN_MODE              67.848630
FLOORSMIN_MEDI              67.848630
YEARS_BUILD_MEDI            66.497784
YEARS_BUILD_MODE            66.497784
YEARS_BUILD_AVG             66.497784
OWN_CAR_AGE                 65.990810
LANDAREA_MEDI               59.376738
LANDAREA_MODE               59.376738
LANDAREA_AVG                59.376738
BASEMENTAREA_MEDI           58.515956
BASEMENTAREA_AVG            58.515956
BASEMENTAREA_MODE           58.515956
EXT_SOURCE_1                56.381073
NONLIVINGAREA_MODE          55.179164
NONLIVINGAREA_AVG           55.179164
NONLIVINGAREA_MEDI          55.179164
ELEVATORS_MEDI              53.295980
ELEVATORS_AVG               53.295980
ELEVATORS_MODE              53.295980
dtype: float64
```

```
plt.figure(figsize=(12,8))
percent_missing.sort_values(ascending=False).plot(kind='barh')
plt.title("Column wise missng value Pecentage")
```

```
Text(0.5, 1.0, 'Column wise missng value Pecentage')
```



These two column have 99.64% data as null

- RATE_INTEREST_PRIVILEGED
- RATE_INTEREST_PRIMARY

```
previous_application[previous_application['RATE_INTEREST_PRIVILEGED'].
notnull()].head(10)
```

```
      SK_ID_PREV  SK_ID_CURR NAME_CONTRACT_TYPE  AMT_ANNUITY
AMT_APPLICATION  \
0        2030495      271877     Consumer loans     1730.430
17145.000
30       2138466      170289     Consumer loans     3286.485
32562.000
506      1017424      250662     Consumer loans     6408.540
79372.215
598      2388655      414811     Consumer loans    14152.545
153387.000
659      1817104      332996     Consumer loans     1335.600
16279.290
962      1263918      346131     Consumer loans     3887.820
48153.960
1027     2813538      441346     Consumer loans     8021.115
79868.160
1028     1431119      205692     Consumer loans     3117.915
38403.270
1274     2229074      163626     Consumer loans     3031.695
30037.500
1360     1340211      420969     Consumer loans     8670.375
85905.000

      AMT_CREDIT  AMT_DOWN_PAYMENT  AMT_GOODS_PRICE  \
```

0	17145.00	0.000	17145.000
30	32562.00	0.000	32562.000
506	63495.00	15877.215	79372.215
598	138046.50	15340.500	153387.000
659	13027.50	3251.790	16279.290
962	38520.00	9633.960	48153.960
1027	79472.16	7987.500	79868.160
1028	30721.50	7681.770	38403.270
1274	30037.50	0.000	30037.500
1360	85905.00	0.000	85905.000

	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCESS_START	\
0	SATURDAY	15	
30	FRIDAY	12	
506	SUNDAY	16	
598	THURSDAY	9	
659	WEDNESDAY	13	
962	MONDAY	20	
1027	SUNDAY	11	
1028	WEDNESDAY	10	
1274	FRIDAY	18	
1360	SATURDAY	18	

	FLAG_LAST_APPL_PER_CONTRACT	NFLAG_LAST_APPL_IN_DAY	RATE_DOWN_PAYMENT	\
0	Y	1	0.000000	
30	Y	1	0.000000	
506	Y	1	0.217856	
598	Y	1	0.108922	
659	Y	1	0.217546	
962	Y	1	0.217890	
1027	Y	1	0.099464	
1028	Y	1	0.217850	
1274	Y	1	0.000000	
1360	Y	1	0.000000	

	RATE_INTEREST_PRIMARY	RATE_INTEREST_PRIVILEGED	NAME_CASH_LOAN_PURPOSE	\
0	0.182832	0.867336	XAP	

```
30                 0.196914                  0.867336
XAP
506                0.189136                  0.835095
XAP
598                0.695668                  0.568710
XAP
659                0.191757                  0.845137
XAP
962                0.189122                  0.835095
XAP
1027               0.193330                  0.852537
XAP
1028               0.160716                  0.715645
XAP
1274               0.196900                  0.867336
XAP
1360               0.196914                  0.867336
XAP

     NAME_CONTRACT_STATUS  DAYS_DECISION      NAME_PAYMENT_TYPE  \
0                Approved            -73  Cash through the bank
30               Approved           -307  Cash through the bank
506              Approved           -135  Cash through the bank
598              Approved           -690  Cash through the bank
659              Approved           -253  Cash through the bank
962              Approved           -267  Cash through the bank
1027             Approved           -301  Cash through the bank
1028             Approved           -336  Cash through the bank
1274             Approved           -182  Cash through the bank
1360             Approved           -427  Cash through the bank

     CODE_REJECT_REASON NAME_TYPE_SUITE NAME_CLIENT_TYPE  \
0                   XAP             NaN         Repeater
30                  XAP             NaN        Refreshed
506                 XAP             NaN         Repeater
598                 XAP          Family              New
659                 XAP             NaN         Repeater
962                 XAP             NaN              New
1027                XAP             NaN         Repeater
1028                XAP             NaN         Repeater
1274                XAP             NaN         Repeater
1360                XAP             NaN              New

           NAME_GOODS_CATEGORY NAME_PORTFOLIO NAME_PRODUCT_TYPE
CHANNEL_TYPE  \
0                       Mobile            POS               XNA
Country-wide
30    Photo / Cinema Equipment            POS               XNA
Country-wide
506   Photo / Cinema Equipment            POS               XNA
```

```
Country-wide
598     Construction Materials            POS               XNA
Stone
659       Consumer Electronics            POS               XNA
Country-wide
962                     Mobile            POS               XNA
Country-wide
1027                    Mobile            POS               XNA
Country-wide
1028                 Computers            POS               XNA
Country-wide
1274                    Mobile            POS               XNA
Country-wide
1360                 Computers            POS               XNA
Country-wide

      SELLERPLACE_AREA NAME_SELLER_INDUSTRY  CNT_PAYMENT
NAME_YIELD_GROUP  \
0                   35         Connectivity         12.0
middle
30                  20         Connectivity         12.0
middle
506                 51         Connectivity         12.0
middle
598                 30         Construction         12.0
middle
659                100         Connectivity         12.0
middle
962                 35         Connectivity         12.0
middle
1027                15         Connectivity         12.0
middle
1028                47         Connectivity         12.0
middle
1274                30         Connectivity         12.0
middle
1360                25         Connectivity         12.0
middle

             PRODUCT_COMBINATION  DAYS_FIRST_DRAWING
DAYS_FIRST_DUE  \
0       POS mobile with interest            365243.0           -42.0

30      POS mobile with interest            365243.0          -273.0

506     POS mobile with interest            365243.0          -100.0

598   POS industry with interest            365243.0          -659.0

659     POS mobile with interest            365243.0          -220.0
```

```
962     POS mobile with interest            365243.0          -233.0

1027    POS mobile with interest            365243.0          -261.0

1028    POS mobile with interest            365243.0          -297.0

1274    POS mobile with interest            365243.0          -131.0

1360    POS mobile with interest            365243.0          -391.0


      DAYS_LAST_DUE_1ST_VERSION  DAYS_LAST_DUE  DAYS_TERMINATION  \
0                         300.0          -42.0             -37.0
30                         57.0         -183.0            -179.0
506                       230.0          -40.0             -37.0
598                      -329.0         -329.0            -323.0
659                       110.0         -130.0            -123.0
962                        97.0         -143.0            -135.0
1027                       69.0         -201.0            -196.0
1028                       33.0         -207.0            -199.0
1274                      199.0          -71.0             -68.0
1360                      -61.0         -301.0            -293.0

      NFLAG_INSURED_ON_APPROVAL
0                           0.0
30                          0.0
506                         0.0
598                         0.0
659                         0.0
962                         0.0
1027                        0.0
1028                        0.0
1274                        0.0
1360                        0.0
```

```
#merge the previous application data with

previous_application_merged =
df_train.iloc[:,:2].merge(previous_application, on = 'SK_ID_CURR', how
= 'left')

previous_application_merged.corr()['TARGET'].sort_values()[:10]
```

```
DAYS_FIRST_DRAWING        -0.031154
HOUR_APPR_PROCESS_START   -0.027809
RATE_DOWN_PAYMENT         -0.026111
AMT_DOWN_PAYMENT          -0.016918
AMT_ANNUITY               -0.014922
DAYS_FIRST_DUE            -0.006651
```

```
AMT_APPLICATION           -0.005583
NFLAG_LAST_APPL_IN_DAY    -0.002887
SELLERPLACE_AREA          -0.002539
AMT_CREDIT                -0.002350
Name: TARGET, dtype: float64
```

Now we will check the categorical column first

```
for i in previous_application_cat_cols:
    print(f"For feature : {i } count for each category is is \n \n
{previous_application_merged[i].value_counts()}")
    print(f"For feature : {i } NULL values are :
{previous_application_merged[i].isna().sum()}")
    print("="*100)
```

```
For feature : NAME_CONTRACT_TYPE count for each category is is

  Cash loans         626764
Consumer loans     625256
Revolving loans    161368
XNA                   313
Name: NAME_CONTRACT_TYPE, dtype: int64
For feature : NAME_CONTRACT_TYPE NULL values are : 16454
==========================================================================
==============================
For feature : WEEKDAY_APPR_PROCESS_START count for each category is is


  WEDNESDAY    215519
TUESDAY      215058
MONDAY       214509
FRIDAY       213373
THURSDAY     211005
SATURDAY     204159
SUNDAY       140078
Name: WEEKDAY_APPR_PROCESS_START, dtype: int64
For feature : WEEKDAY_APPR_PROCESS_START NULL values are : 16454
==========================================================================
==============================
For feature : FLAG_LAST_APPL_PER_CONTRACT count for each category is
is

  Y    1406387
N       7314
Name: FLAG_LAST_APPL_PER_CONTRACT, dtype: int64
For feature : FLAG_LAST_APPL_PER_CONTRACT NULL values are : 16454
==========================================================================
==============================
For feature : NAME_CASH_LOAN_PURPOSE count for each category is is

  XAP                                   786937
```

```
XNA                                 567351
Repairs                              20117
Other                                13432
Urgent needs                          7236
Buying a used car                     2469
Building a house or an annex          2344
Everyday expenses                     2052
Medicine                              1871
Payments on other loans               1573
Education                             1334
Journey                               1042
Purchase of electronic equipment       934
Buying a new car                       886
Wedding / gift / holiday               801
Buying a home                          701
Car repairs                            691
Furniture                              660
Buying a holiday home / land           463
Business development                   359
Gasification / water supply            251
Buying a garage                        116
Hobby                                   45
Money for a third person                23
Refusal to name the goal                13
Name: NAME_CASH_LOAN_PURPOSE, dtype: int64
For feature : NAME_CASH_LOAN_PURPOSE NULL values are : 16454
========================================================================
==============================
For feature : NAME_CONTRACT_STATUS count for each category is is

  Approved        886099
Canceled        259441
Refused         245390
Unused offer     22771
Name: NAME_CONTRACT_STATUS, dtype: int64
For feature : NAME_CONTRACT_STATUS NULL values are : 16454
========================================================================
==============================
For feature : NAME_PAYMENT_TYPE count for each category is is

  Cash through the bank                        882700
XNA                                           522960
Non-cash from your account                      7100
Cashless from the account of the employer        941
Name: NAME_PAYMENT_TYPE, dtype: int64
For feature : NAME_PAYMENT_TYPE NULL values are : 16454
========================================================================
==============================
For feature : CODE_REJECT_REASON count for each category is is
```

```
  XAP       1145533
HC         145984
LIMIT       47773
SCO         32636
CLIENT      22771
SCOFR       10875
XNA          4378
VERIF        3079
SYSTEM        672
Name: CODE_REJECT_REASON, dtype: int64
For feature : CODE_REJECT_REASON NULL values are : 16454
==========================================================================
==============================
For feature : NAME_TYPE_SUITE count for each category is is

  Unaccompanied      427635
Family             181880
Spouse, partner     57586
Children            27106
Other_B             15073
Other_A              7820
Group of people      1929
Name: NAME_TYPE_SUITE, dtype: int64
For feature : NAME_TYPE_SUITE NULL values are : 711126
==========================================================================
==============================
For feature : NAME_CLIENT_TYPE count for each category is is

  Repeater     1037682
New           259540
Refreshed     114936
XNA             1543
Name: NAME_CLIENT_TYPE, dtype: int64
For feature : NAME_CLIENT_TYPE NULL values are : 16454
==========================================================================
==============================
For feature : NAME_GOODS_CATEGORY count for each category is is

  XNA                          797209
Mobile                       193743
Consumer Electronics         104785
Computers                     89923
Audio/Video                   85502
Furniture                     45439
Photo / Cinema Equipment      21707
Construction Materials        21356
Clothing and Accessories      19590
Auto Accessories               6228
Jewelry                        5188
Homewares                      4309
```

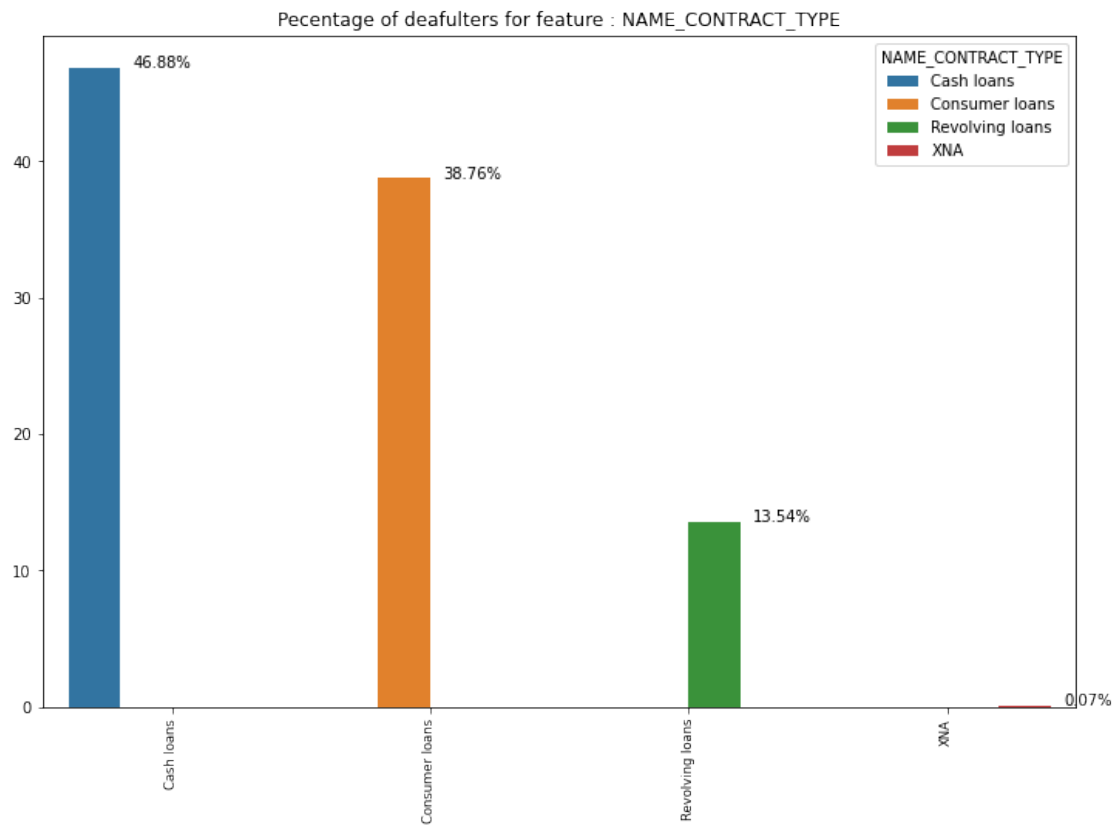
```
Medical Supplies              3240
Vehicles                      2854
Sport and Leisure             2540
Other                         2337
Gardening                     2322
Office Appliances             1985
Tourism                       1324
Medicine                      1302
Direct Sales                   330
Fitness                        166
Additional Service             109
Education                       89
Weapon                          65
Insurance                       58
Animals                          1
Name: NAME_GOODS_CATEGORY, dtype: int64
For feature : NAME_GOODS_CATEGORY NULL values are : 16454
========================================================================
==============================
For feature : NAME_PORTFOLIO count for each category is is

  POS      592460
Cash     391663
XNA      307213
Cards    121985
Cars        380
Name: NAME_PORTFOLIO, dtype: int64
For feature : NAME_PORTFOLIO NULL values are : 16454
========================================================================
==============================
For feature : NAME_PRODUCT_TYPE count for each category is is

  XNA        900053
x-sell     385394
walk-in    128254
Name: NAME_PRODUCT_TYPE, dtype: int64
For feature : NAME_PRODUCT_TYPE NULL values are : 16454
========================================================================
==============================
For feature : CHANNEL_TYPE count for each category is is

  Credit and cash offices       600342
Country-wide                  423062
Stone                         183508
Regional / Local               92472
Contact center                 58891
AP+ (Cash loan)                49624
Channel of corporate sales      5396
Car dealer                       406
Name: CHANNEL_TYPE, dtype: int64
```

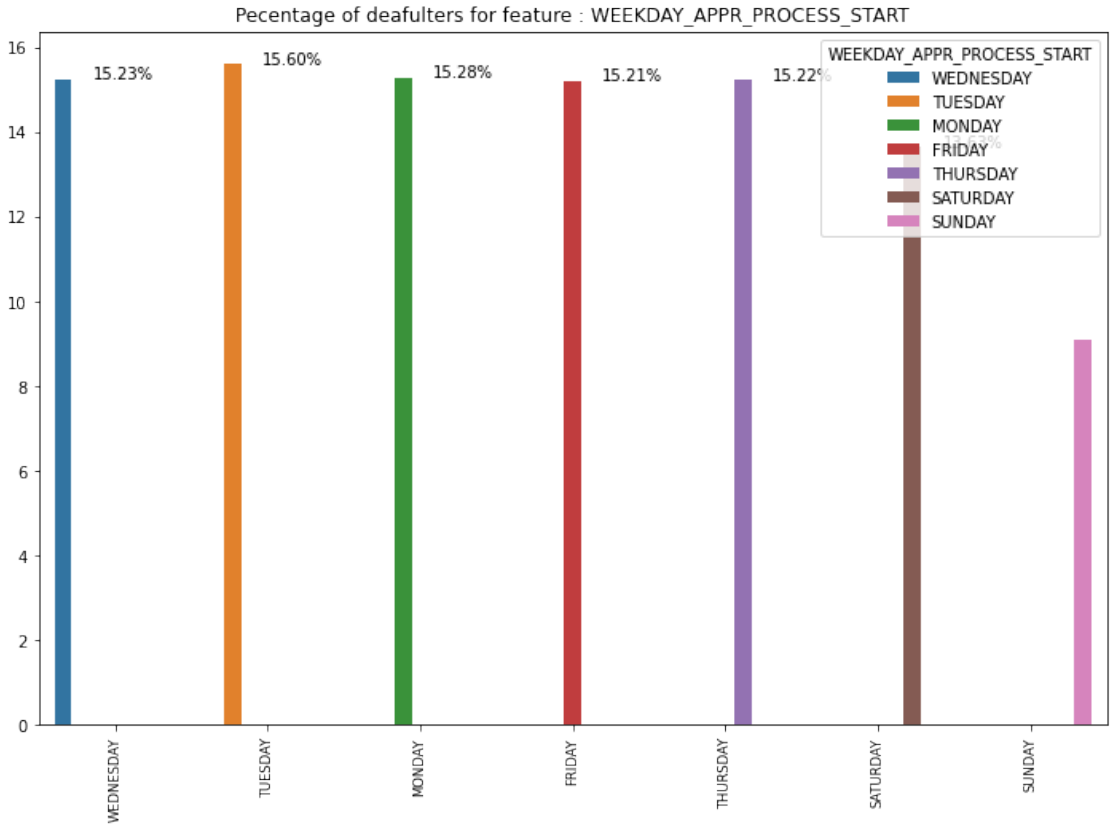
```
For feature : CHANNEL_TYPE NULL values are : 16454
==========================================================================
==============================
For feature : NAME_SELLER_INDUSTRY count for each category is is

  XNA                     715322
Consumer electronics    341701
Connectivity            238350
Furniture                48966
Construction             25417
Clothing                 19896
Industry                 16567
Auto technology           4080
Jewelry                   2107
MLM partners               921
Tourism                    374
Name: NAME_SELLER_INDUSTRY, dtype: int64
For feature : NAME_SELLER_INDUSTRY NULL values are : 16454
==========================================================================
==============================
For feature : NAME_YIELD_GROUP count for each category is is

  XNA           429198
middle        324971
high          306061
low_normal    274884
low_action     78587
Name: NAME_YIELD_GROUP, dtype: int64
For feature : NAME_YIELD_GROUP NULL values are : 16454
==========================================================================
==============================
For feature : PRODUCT_COMBINATION count for each category is is

  Cash                              235101
POS household with interest       226464
POS mobile with interest          190850
Cash X-Sell: middle               120036
Cash X-Sell: low                  110599
Card Street                        94594
POS industry with interest         83508
POS household without interest     71519
Card X-Sell                        66774
Cash Street: high                  51699
Cash X-Sell: high                  50802
Cash Street: middle                29934
Cash Street: low                   28593
POS other with interest            20309
POS mobile without interest        19890
POS industry without interest      10621
POS others without interest         2095
```

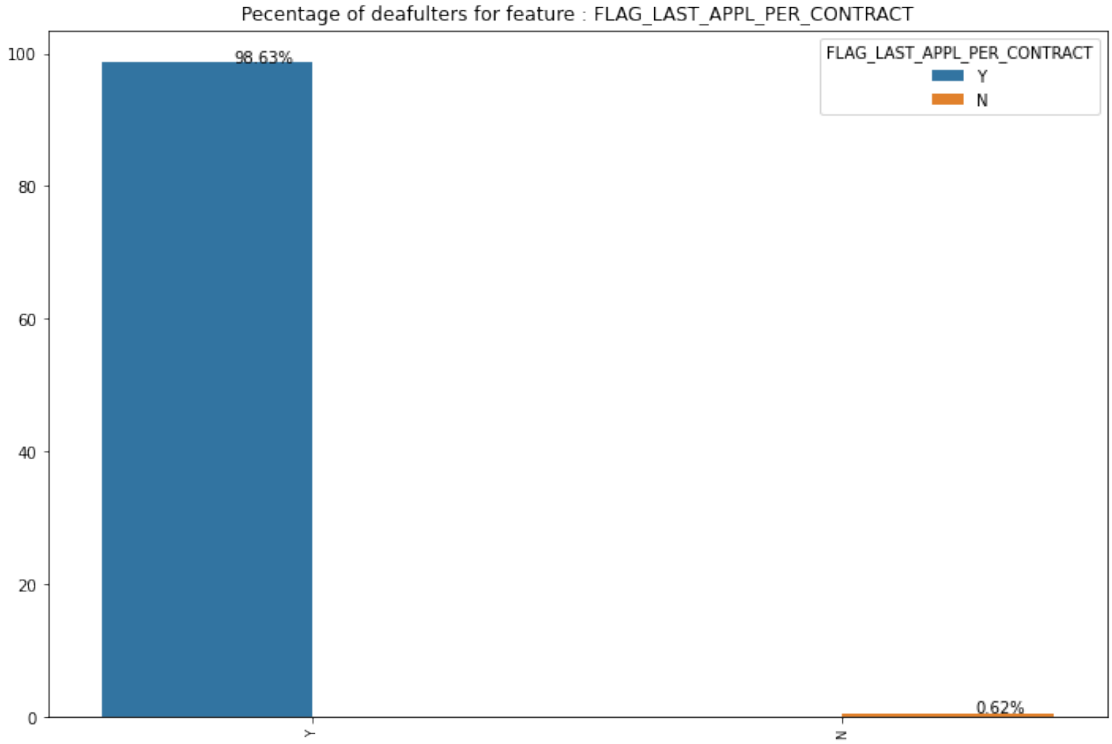
```
Name: PRODUCT_COMBINATION, dtype: int64
For feature : PRODUCT_COMBINATION NULL values are : 16767
==========================================================================
==============================
```

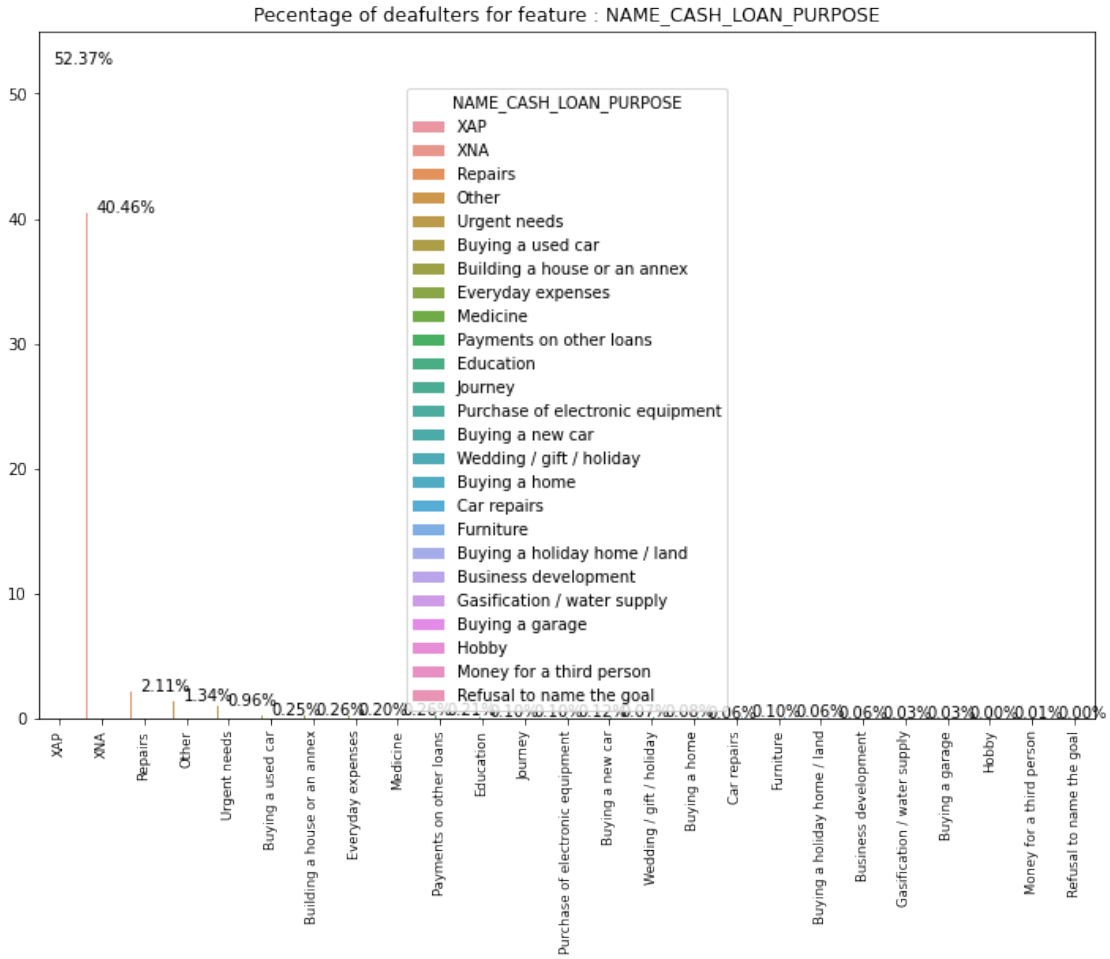
- We can see for most of categorical columns the count of missing values rows are 16454

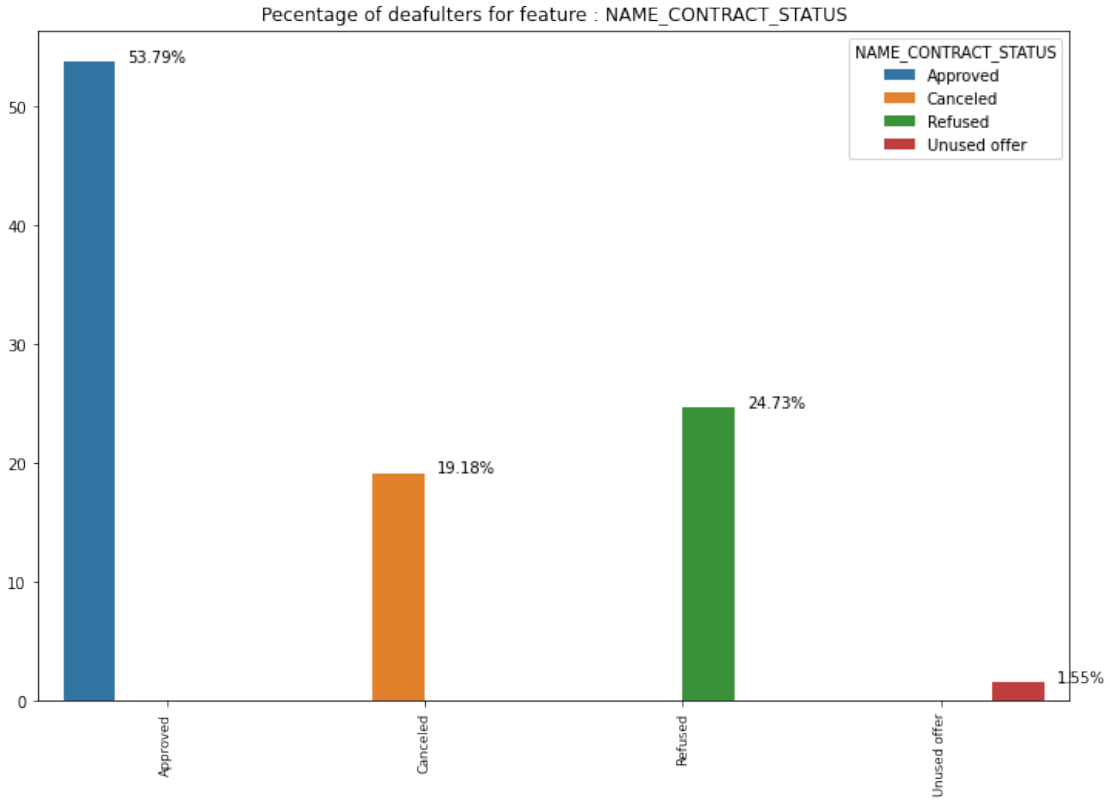
```
for i in previous_application_cat_cols:

    plot_for_categorical_defaulters(previous_application_merged,i)
```

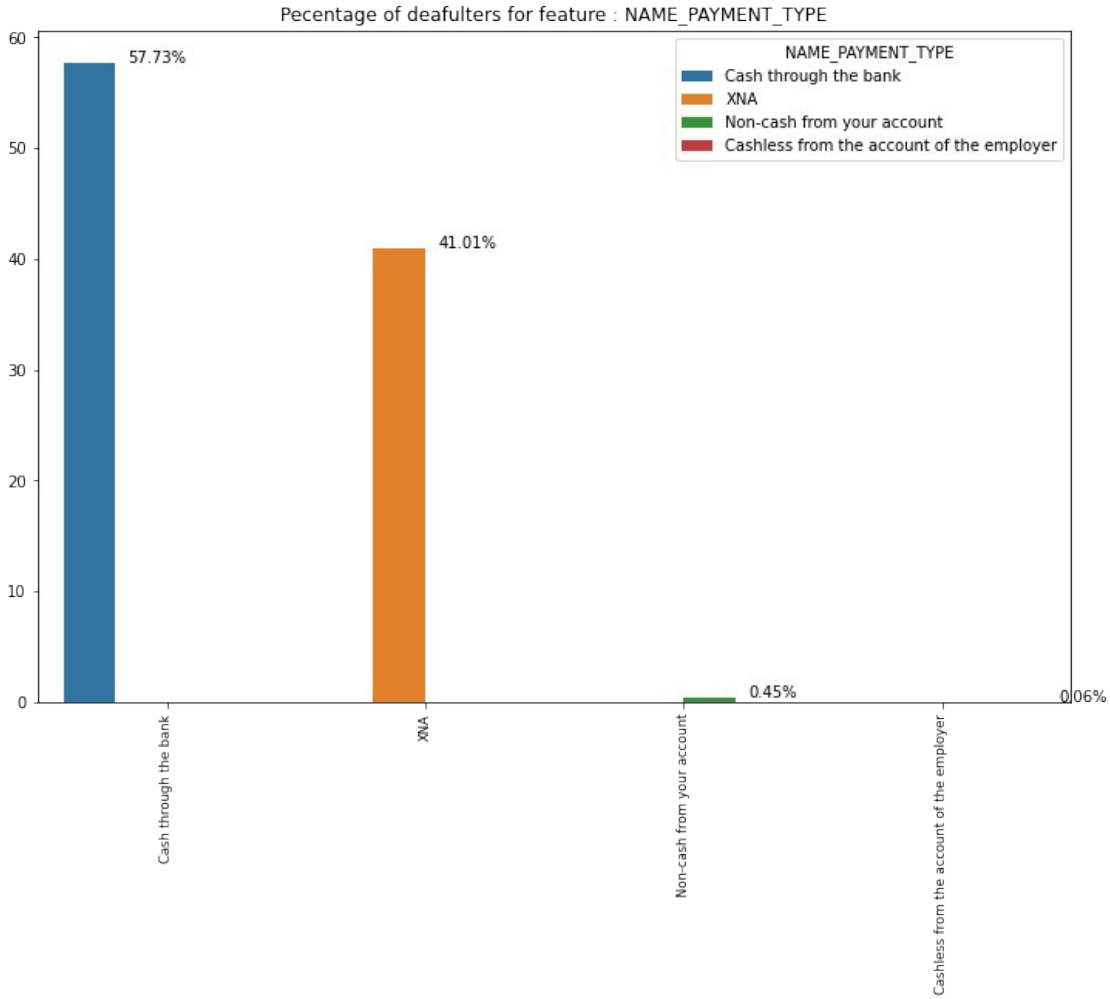


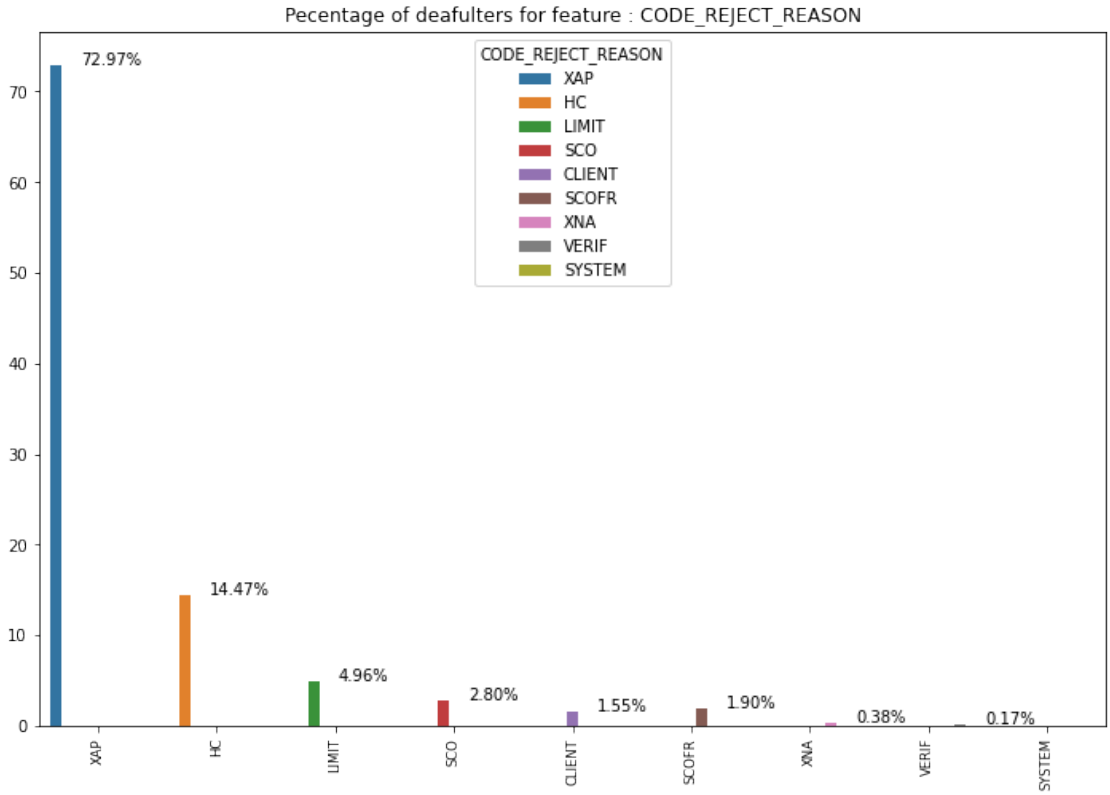

Pecentage of deafulters for feature : WEEKDAY_APPR_PROCESS_START
WEEKDAY_APPR_PROCESS_START
WEDNESDAY
TUESDAY
MONDAY
FRIDAY
THURSDAY
SATURDAY
SUNDAY
15.23%
15.60%
15.28%
15.21%
15.22%
0
2
4
6
8
10
12
14
16

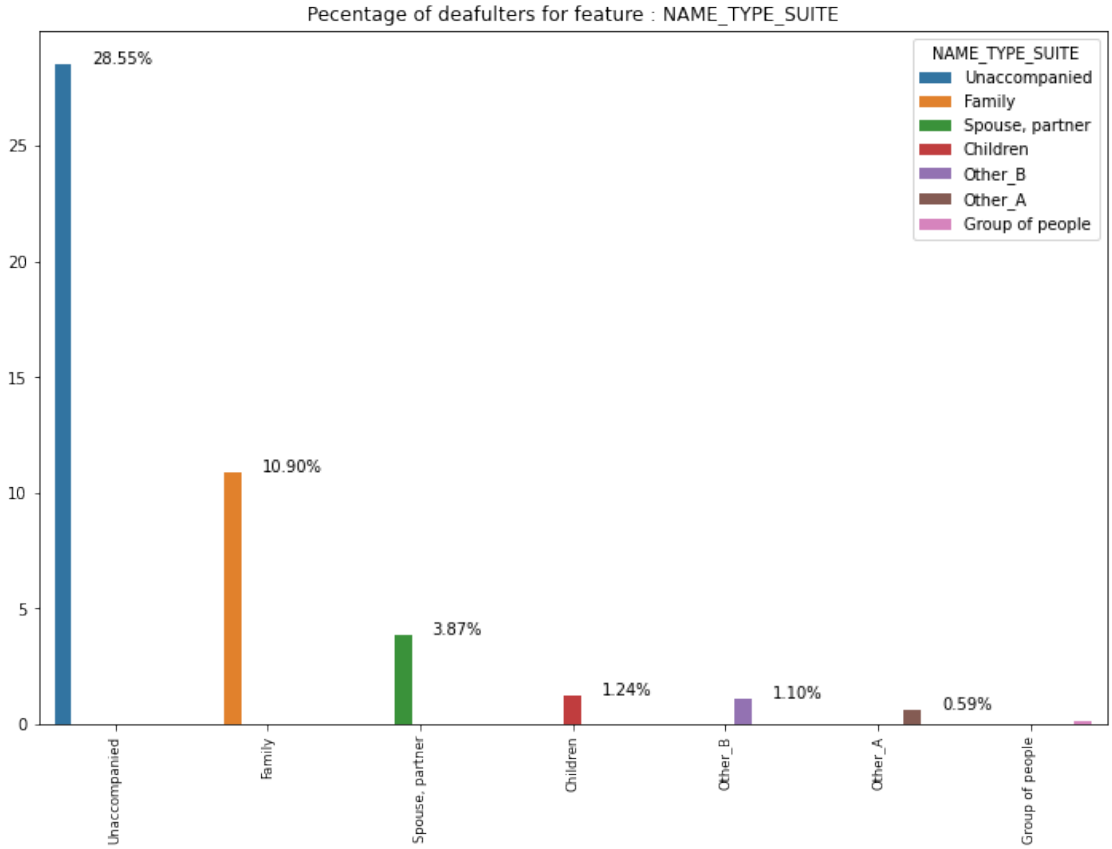

Pecentage of deafulters for feature : FLAG_LAST_APPL_PER_CONTRACT
FLAG_LAST_APPL_PER_CONTRACT
Y
N
98.63%
0.62%
0
20
40
60
80
100

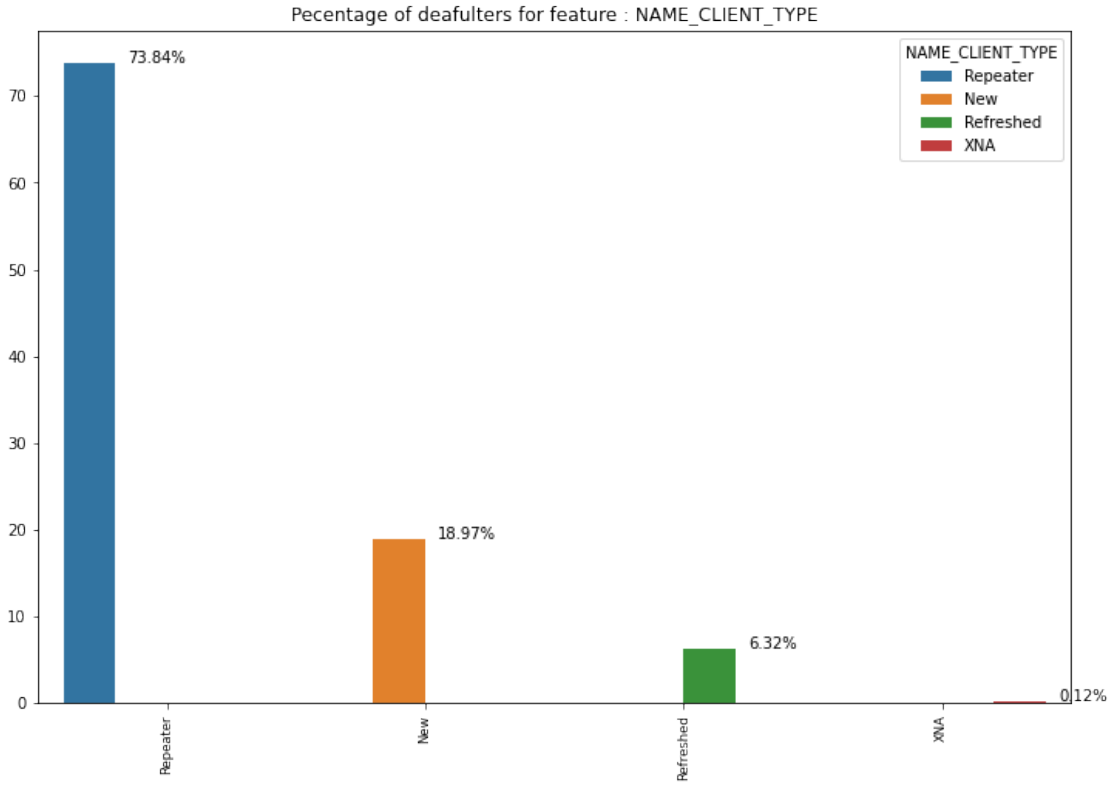

Pecentage of deafulters for feature : NAME_CASH_LOAN_PURPOSE
52.37%
40.46%
2.11%
1.34%
0.96%
0.25%
0.26%
0.20%
0.06%
0.10%
0.06%
0.06%
0.03%
0.03%
0.00%
0.01%
0.00%
0
10
20
30
40
50
NAME_CASH_LOAN_PURPOSE
XAP
XNA
Repairs
Other
Urgent needs
Buying a used car
Building a house or an annex
Everyday expenses
Medicine
Payments on other loans
Education
Journey
Purchase of electronic equipment
Buying a new car
Wedding / gift / holiday
Buying a home
Car repairs
Furniture
Buying a holiday home / land
Business development
Gasification / water supply
Buying a garage
Hobby
Money for a third person
Refusal to name the goal

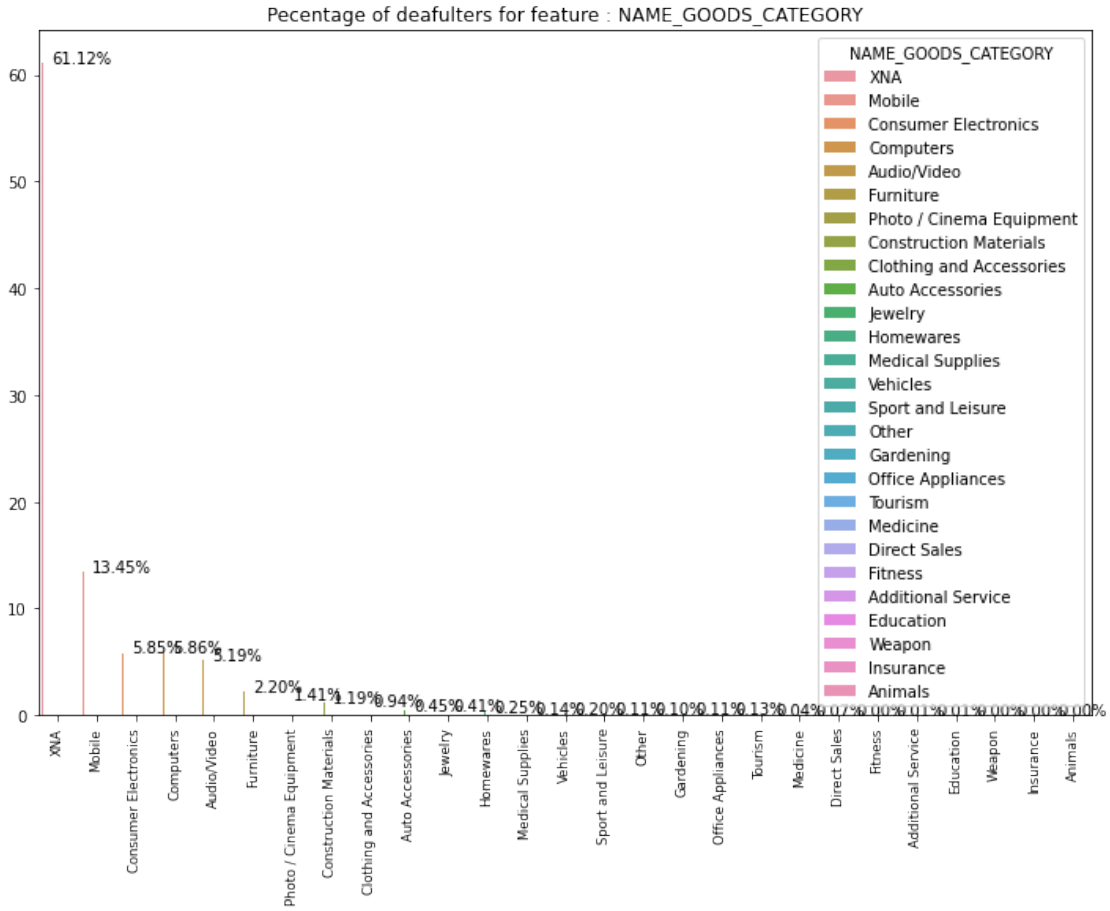

Pecentage of deafulters for feature : NAME_CONTRACT_STATUS
NAME_CONTRACT_STATUS
Approved
Canceled
Refused
Unused offer
53.79%
19.18%
24.73%
1.55%
0
10
20
30
40
50
Approved
Canceled
Refused
Unused offer

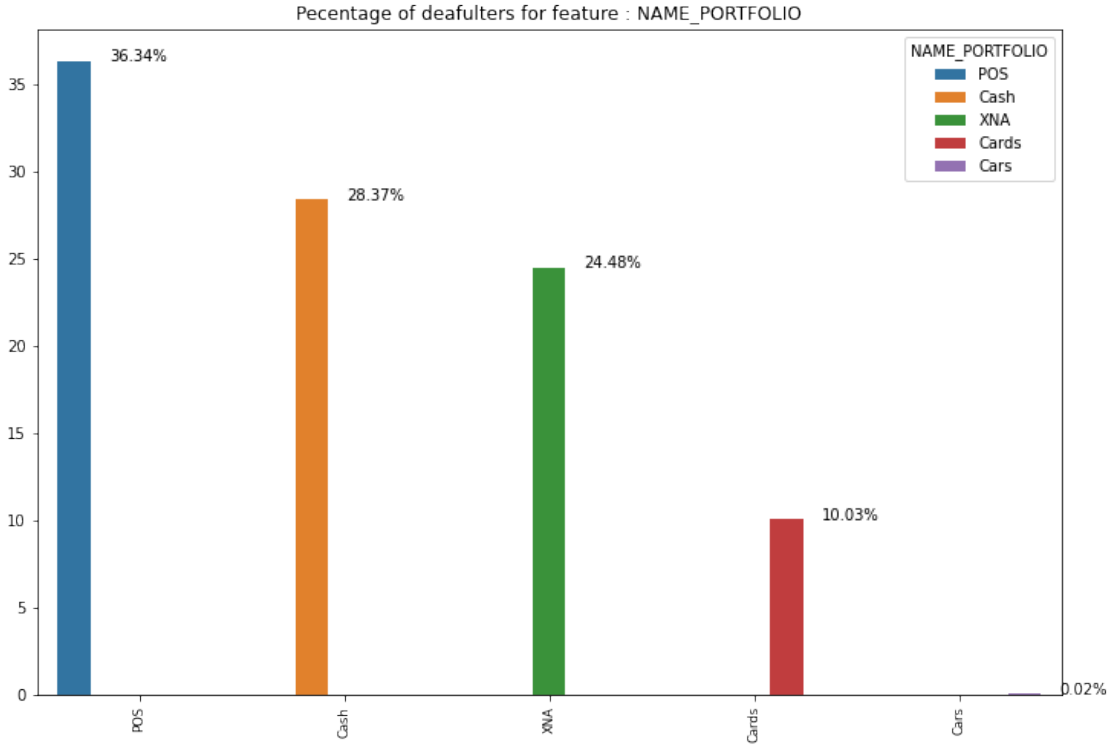

Pecentage of deafulters for feature : NAME_PAYMENT_TYPE
NAME_PAYMENT_TYPE
Cash through the bank
XNA
Non-cash from your account
Cashless from the account of the employer
57.73%
41.01%
0.45%
0.06%
0
10
20
30
40
50
60
Cash through the bank
XNA
Non-cash from your account
Cashless from the account of the employer

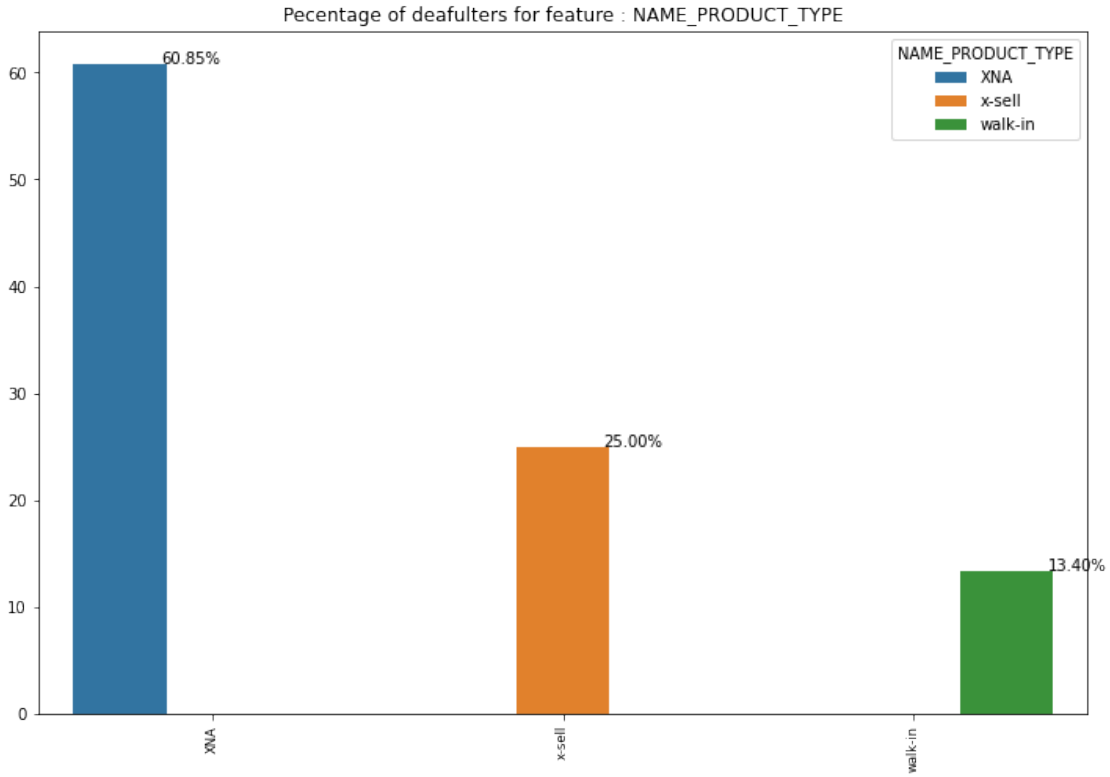

Pecentage of deafulters for feature : CODE_REJECT_REASON
CODE_REJECT_REASON
XAP
HC
LIMIT
SCO
CLIENT
SCOFR
XNA
VERIF
SYSTEM
72.97%
14.47%
4.96%
2.80%
1.55%
1.90%
0.38%
0.17%
0
10
20
30
40
50
60
70

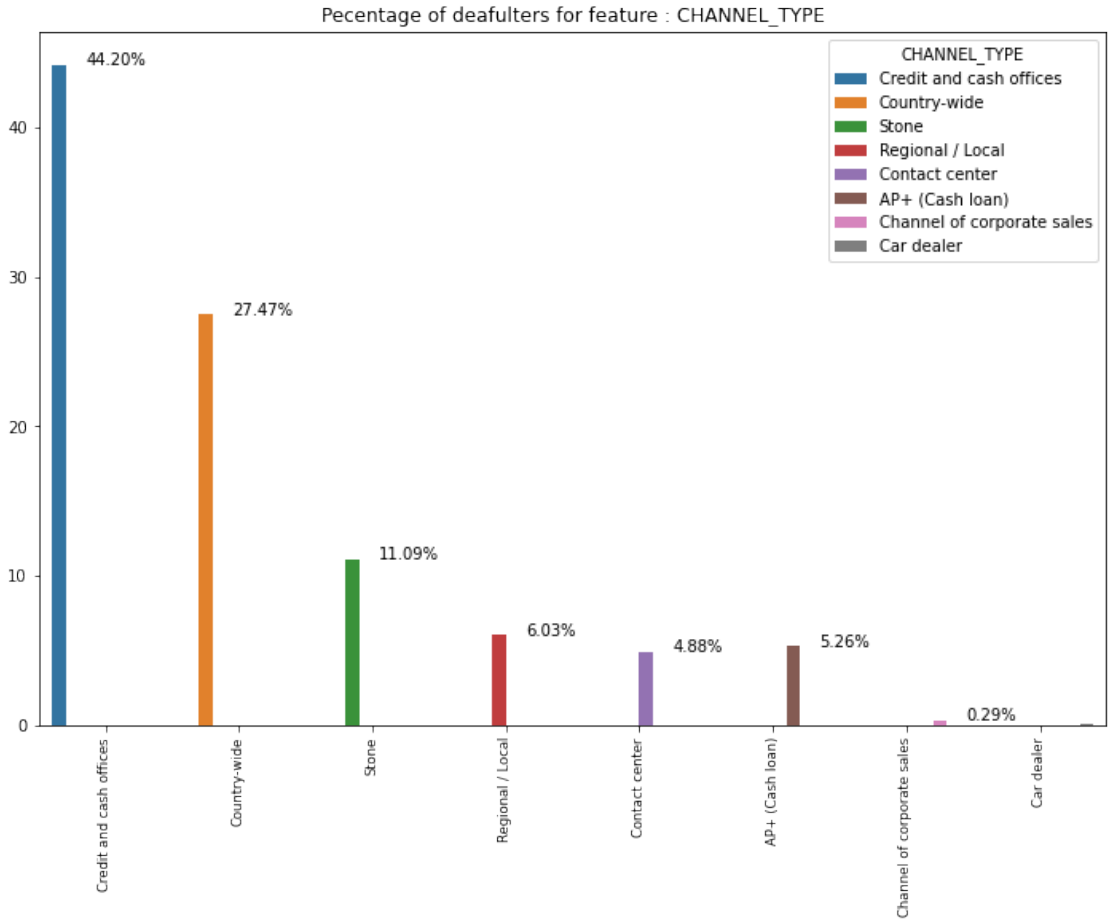

Pecentage of deafulters for feature : NAME_TYPE_SUITE
NAME_TYPE_SUITE
Unaccompanied
Family
Spouse, partner
Children
Other_B
Other_A
Group of people
28.55%
10.90%
3.87%
1.24%
1.10%
0.59%
0
5
10
15
20
25

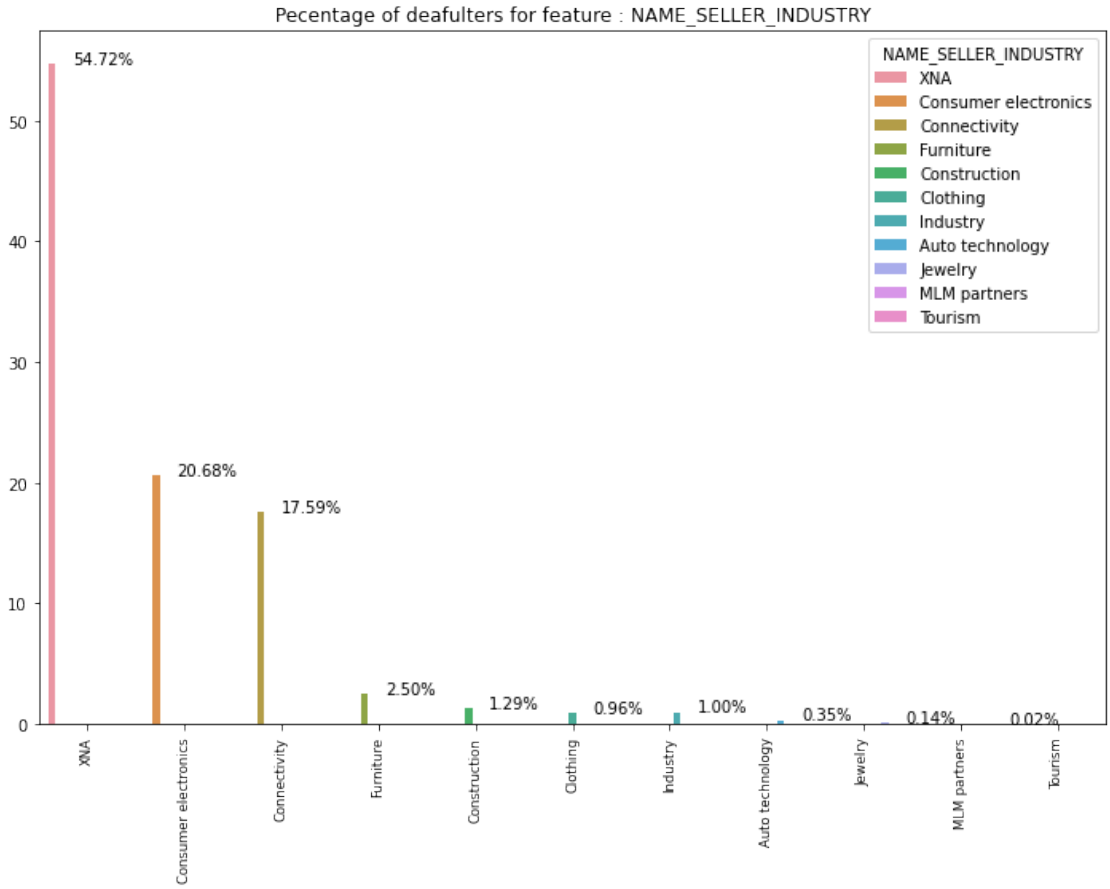

Pecentage of deafulters for feature : NAME_CLIENT_TYPE
NAME_CLIENT_TYPE
Repeater
New
Refreshed
XNA
73.84%
18.97%
6.32%
0.12%
0
10
20
30
40
50
60
70
Repeater
New
Refreshed
XNA

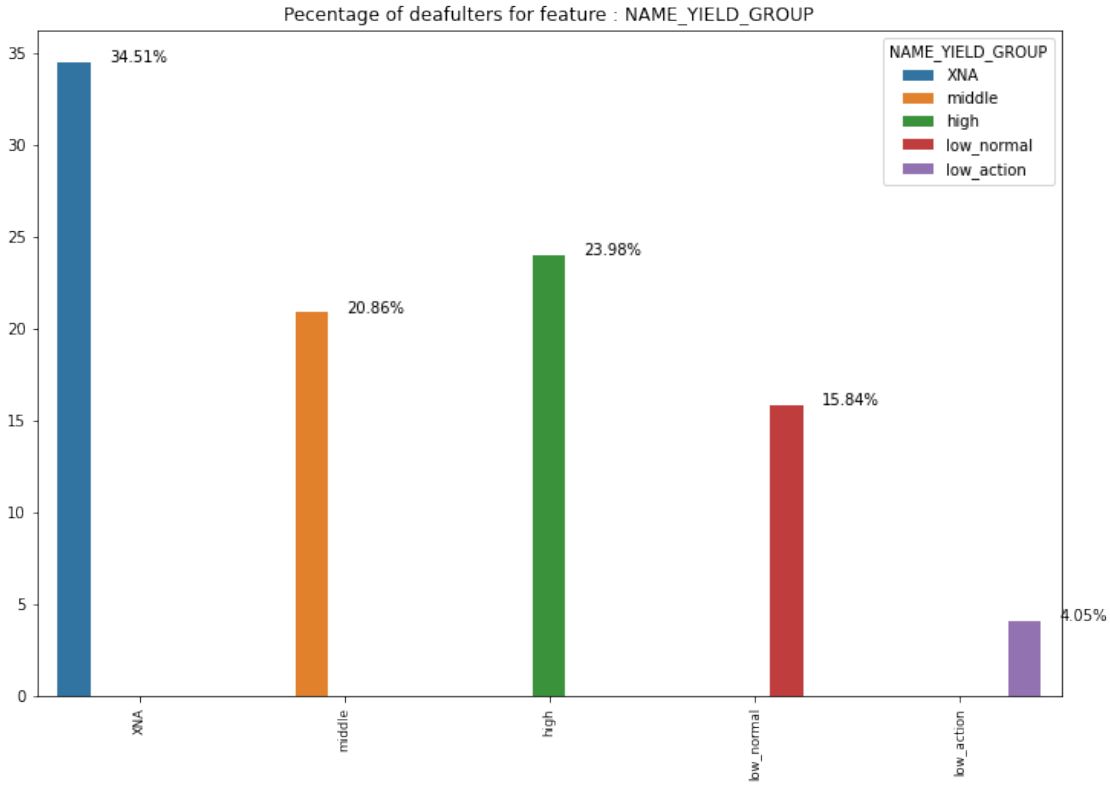

Pecentage of deafulters for feature : NAME_GOODS_CATEGORY
NAME_GOODS_CATEGORY
XNA
Mobile
Consumer Electronics
Computers
Audio/Video
Furniture
Photo / Cinema Equipment
Construction Materials
Clothing and Accessories
Auto Accessories
Jewelry
Homewares
Medical Supplies
Vehicles
Sport and Leisure
Other
Gardening
Office Appliances
Tourism
Medicine
Direct Sales
Fitness
Additional Service
Education
Weapon
Insurance
Animals
61.12%
13.45%
5.85%
5.86%
5.19%
2.20%
1.41%
1.19%
0.94%
0.45%
0.41%
0.25%
0.14%
0.20%
0.11%
0.10%
0.11%
0.13%
0.04%
0.07%
0.00%
0.01%
0.01%
0.00%
0.00%
0.00%

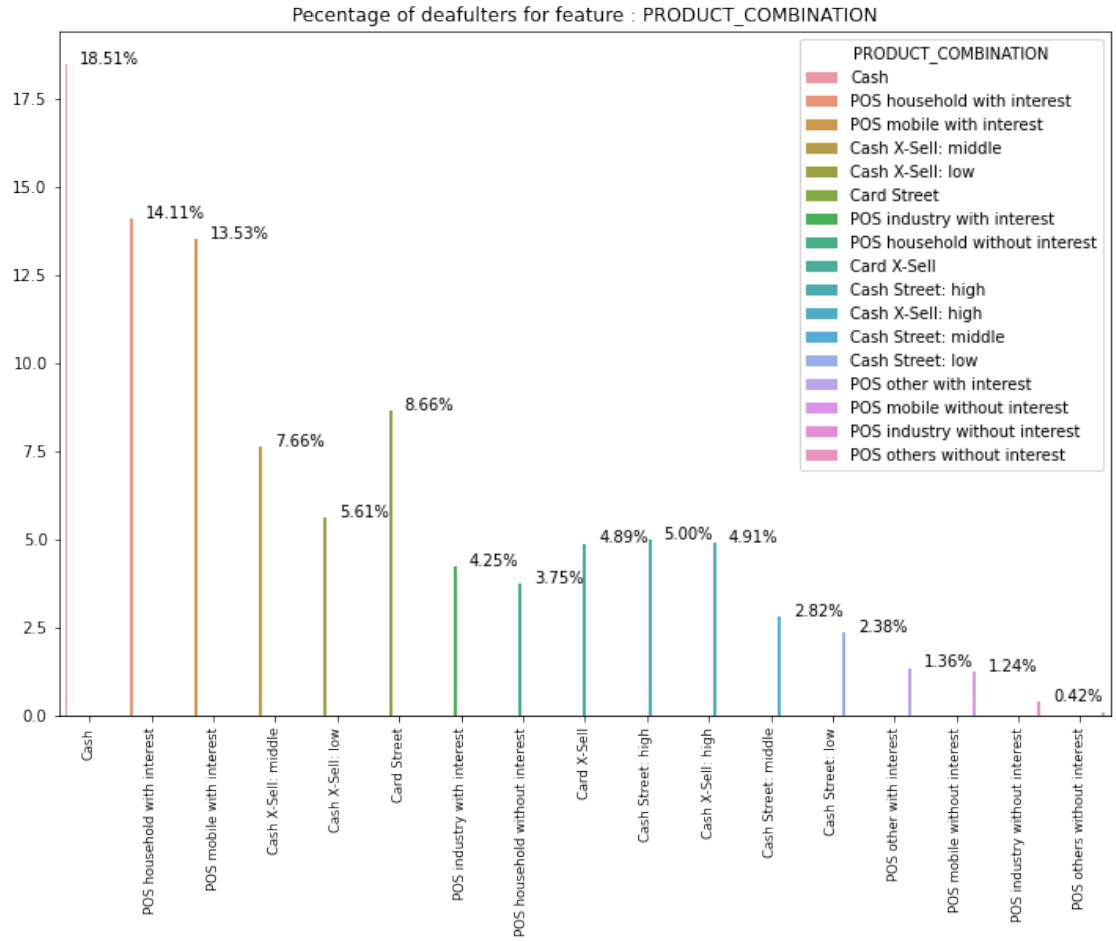

Pecentage of deafulters for feature : NAME_PORTFOLIO
NAME_PORTFOLIO
POS
Cash
XNA
Cards
Cars
36.34%
28.37%
24.48%
10.03%
0.02%


Pecentage of deafulters for feature : NAME_PRODUCT_TYPE
NAME_PRODUCT_TYPE
XNA
x-sell
walk-in
60.85%
25.00%
13.40%
0
10
20
30
40
50
60
XNA
x-sell
walk-in


Pecentage of deafulters for feature : CHANNEL_TYPE
CHANNEL_TYPE
Credit and cash offices
Country-wide
Stone
Regional / Local
Contact center
AP+ (Cash loan)
Channel of corporate sales
Car dealer
44.20%
27.47%
11.09%
6.03%
4.88%
5.26%
0.29%
0
10
20
30
40
Credit and cash offices
Country-wide
Stone
Regional / Local
Contact center
AP+ (Cash loan)
Channel of corporate sales
Car dealer


Pecentage of deafulters for feature : NAME_SELLER_INDUSTRY
NAME_SELLER_INDUSTRY
XNA
Consumer electronics
Connectivity
Furniture
Construction
Clothing
Industry
Auto technology
Jewelry
MLM partners
Tourism
54.72%
20.68%
17.59%
2.50%
1.29%
0.96%
1.00%
0.35%
0.14%
0.02%
0
10
20
30
40
50
XNA
Consumer electronics
Connectivity
Furniture
Construction
Clothing
Industry
Auto technology
Jewelry
MLM partners
Tourism


Pecentage of deafulters for feature : NAME_YIELD_GROUP
NAME_YIELD_GROUP
XNA
middle
high
low_normal
low_action
34.51%
20.86%
23.98%
15.84%
4.05%
0
5
10
15
20
25
30
35
XNA
middle
high
low_normal
low_action



lets see each of categorical column where we can see a better variation with respect to its distribution and its distribution with TARGET COLUMN

NAME_CONTRACT_TYPE

Contract product type (Cash loan, consumer loan [POS] ,...) of the previous application

```
previous_application_merged['NAME_CONTRACT_TYPE'].value_counts()
```

```
Cash loans         626764
Consumer loans     625256
Revolving loans    161368
XNA                   313
Name: NAME_CONTRACT_TYPE, dtype: int64
```

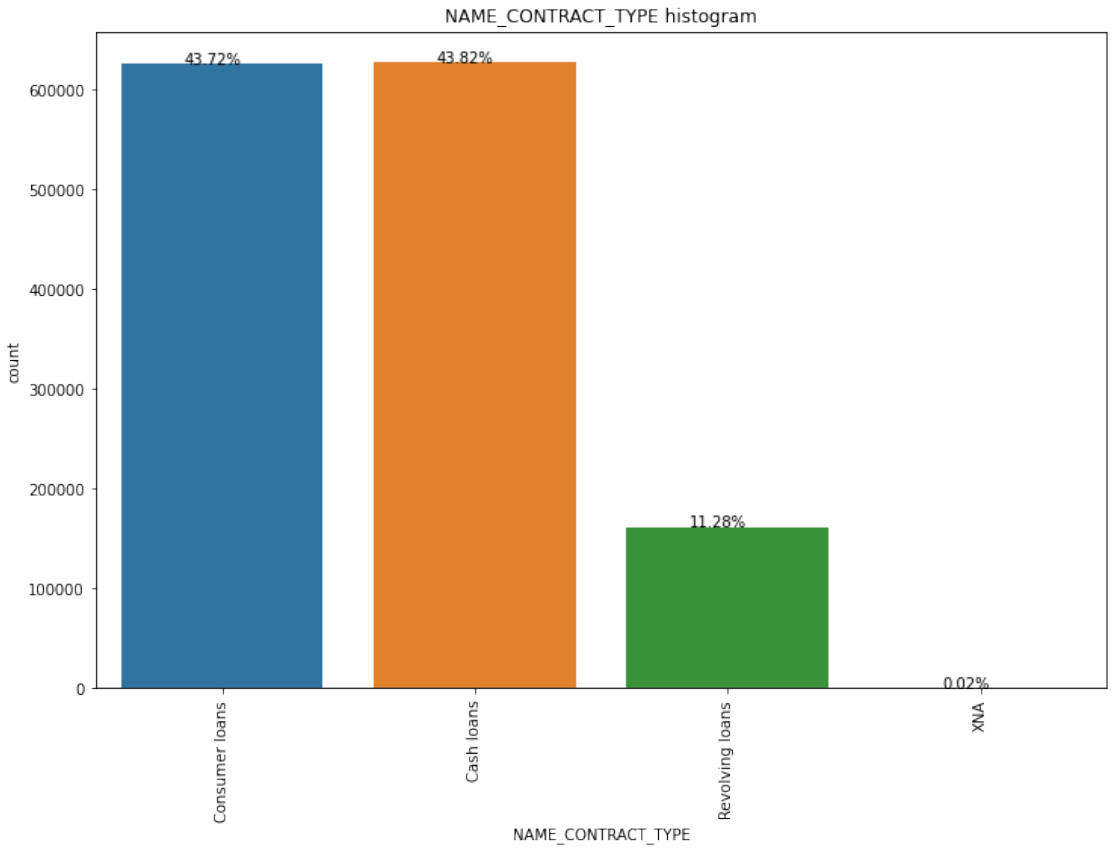
```
previous_application_merged['NAME_CONTRACT_TYPE'].isna().sum()
# NA values
```

```
16454
```

```
# 16454 NA VALUES
```

```
#lets see its distribution
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_CONTRACT_TYPE',data=previous_application_merg
ed)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}
%'.format(p.get_height()*100/len(previous_application_merged)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("NAME_CONTRACT_TYPE histogram")
```

```
Text(0.5, 1.0, 'NAME_CONTRACT_TYPE histogram')
```



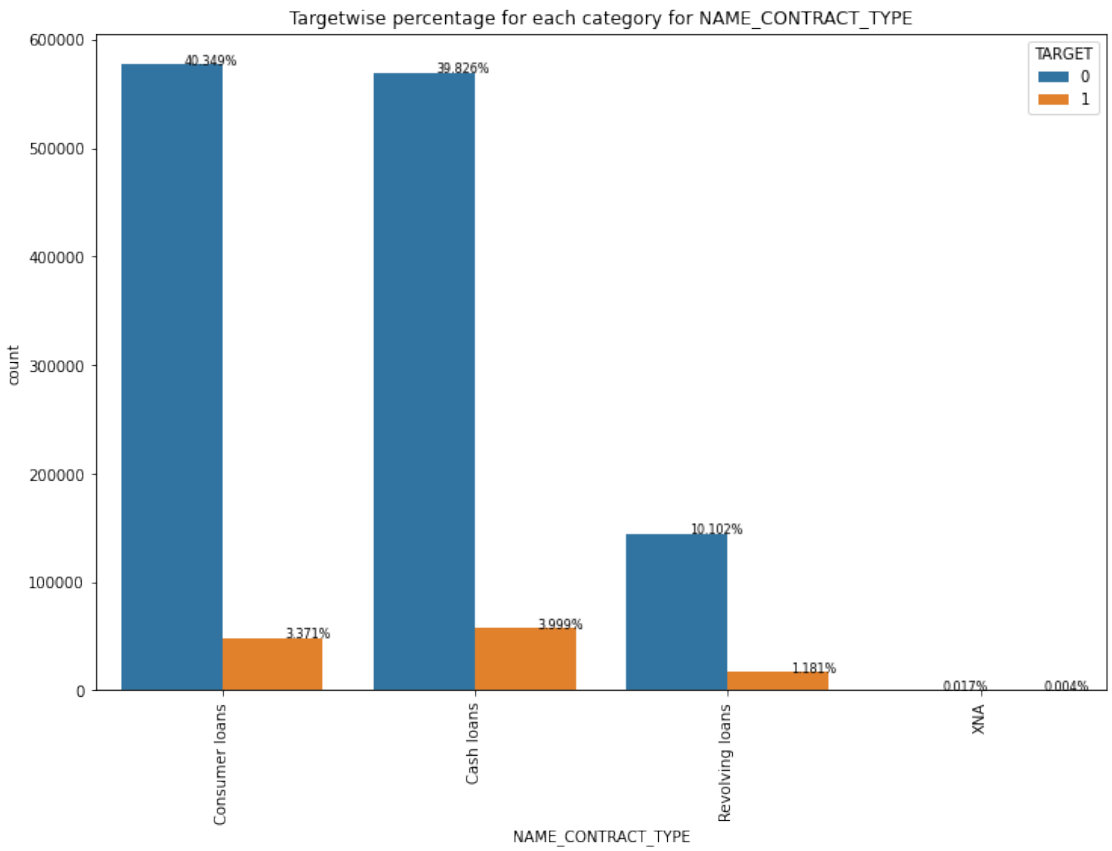
Most loansare in descending orders as

- Consumer Loans
- Cash Loans

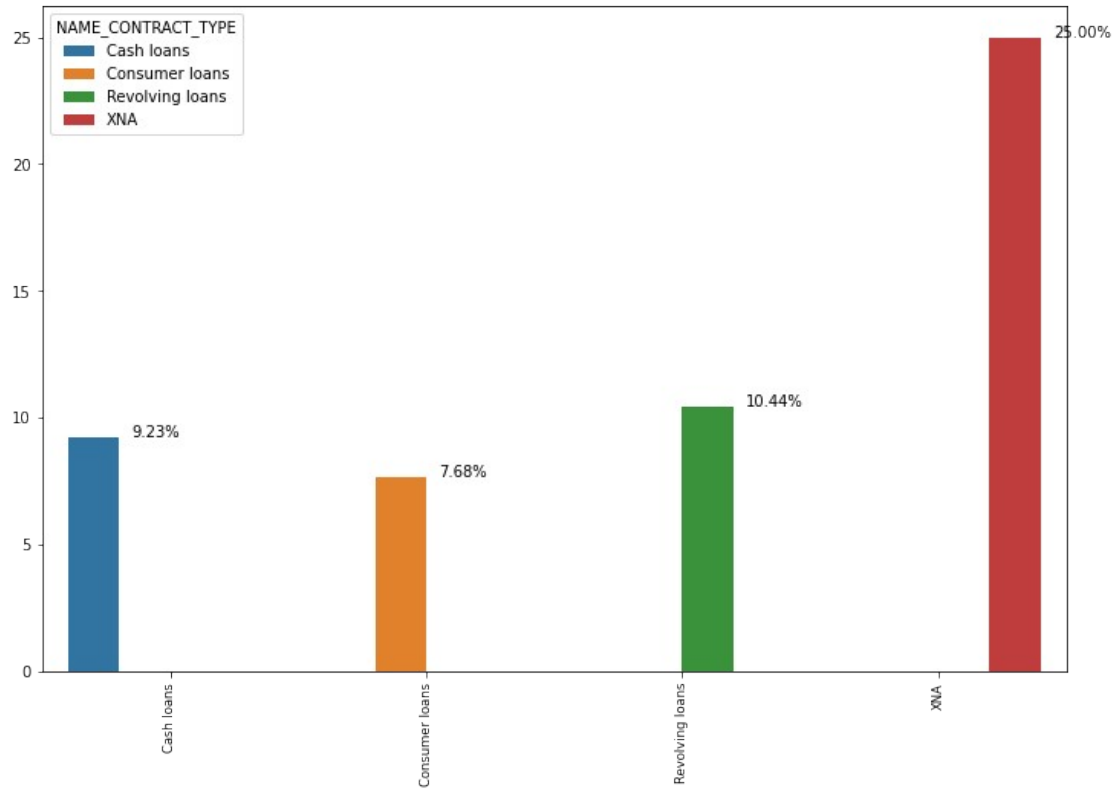
```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_CONTRACT_TYPE',hue='TARGET',data=previous_app
lication_merged)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}
%'.format(p.get_height()*100/len(previous_application_merged)),
```

```
(p.get_x()+0.25, p.get_height()+0.01),size=8)
ax.set_title("Targetwise percentage for each category for
NAME_CONTRACT_TYPE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NAME_CONTRACT_TYPE')
```




```
#lets see in each category what is % of defaulters of same category
plot_for_deafulters_categorywise(previous_application_merged,'NAME_CON
TRACT_TYPE')
```



- Here we can see XNA type loans have most default rate but the catch is ,This is very less occurance in this category
- Cash loan are the most distributed ones and have max defaulters and non defaulters rates

NAME_CASH_LOAN_PURPOSE

Purpose of the cash loan

```
previous_application_merged['NAME_CASH_LOAN_PURPOSE'].value_counts()
```

```
XAP                                   786937
XNA                                   567351
Repairs                                20117
Other                                  13432
Urgent needs                            7236
Buying a used car                       2469
Building a house or an annex            2344
Everyday expenses                       2052
Medicine                                1871
Payments on other loans                 1573
Education                               1334
Journey                                 1042
Purchase of electronic equipment         934
Buying a new car                         886
```

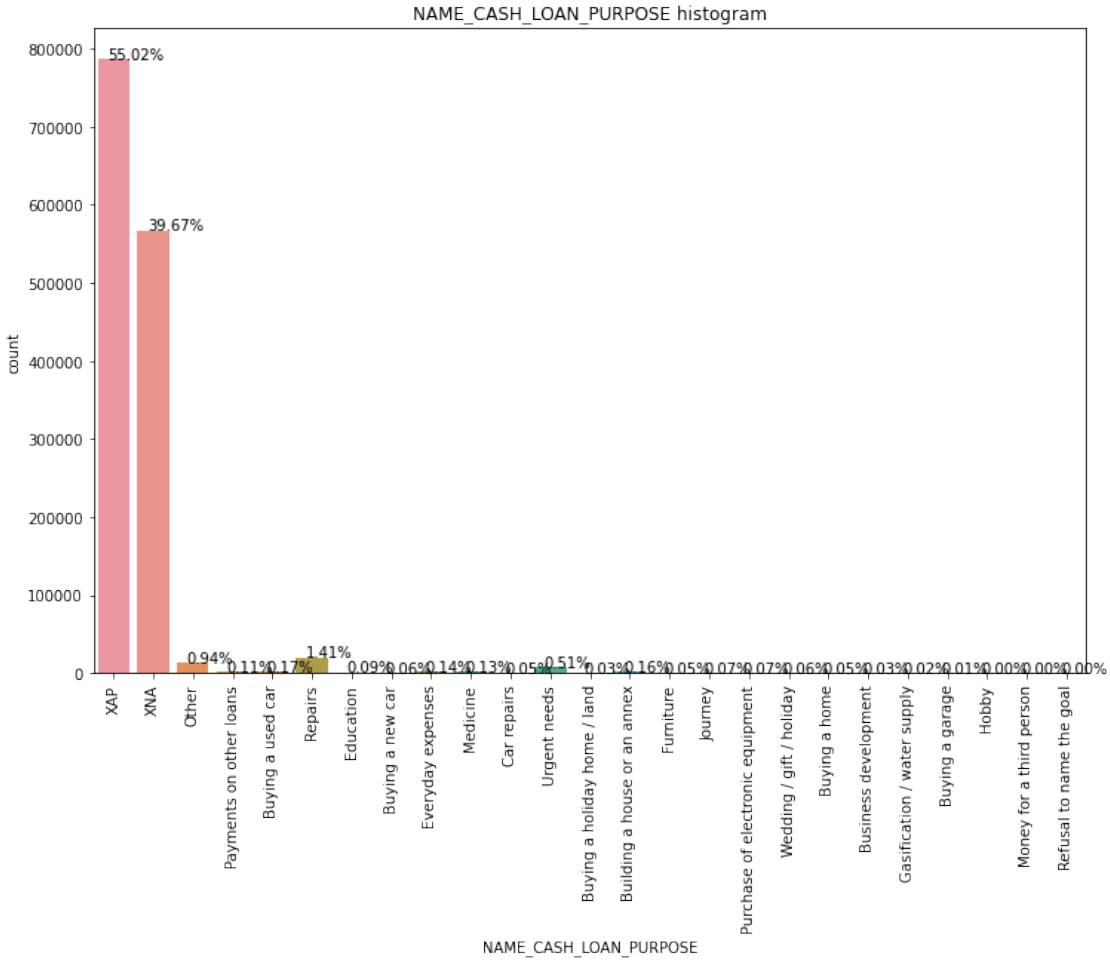
```
Wedding / gift / holiday               801
Buying a home                          701
Car repairs                            691
Furniture                              660
Buying a holiday home / land           463
Business development                   359
Gasification / water supply            251
Buying a garage                        116
Hobby                                   45
Money for a third person                23
Refusal to name the goal                13
Name: NAME_CASH_LOAN_PURPOSE, dtype: int64
```

```
previous_application_merged['NAME_CASH_LOAN_PURPOSE'].isna().sum()
# NA values
```

```
16454
```

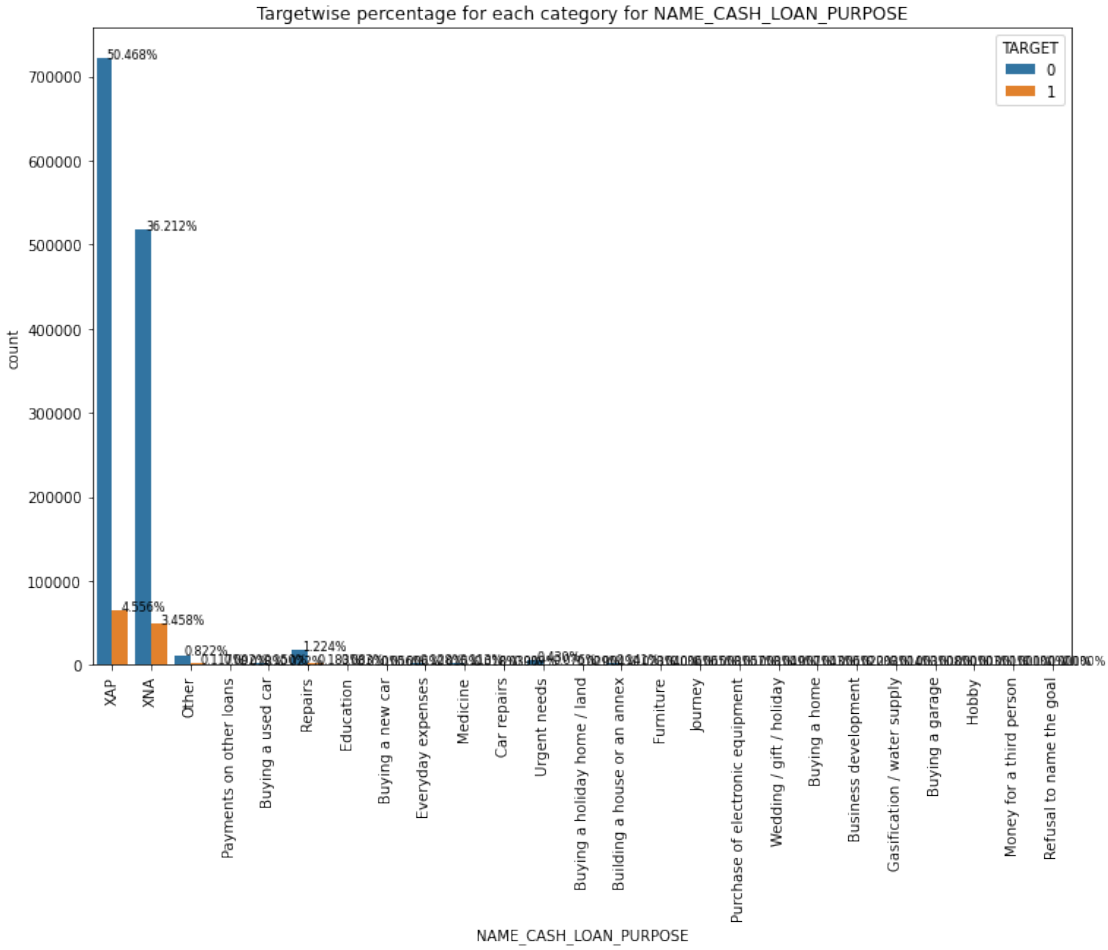
```
# 16454 NA VALUES

#lets see its distribution
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_CASH_LOAN_PURPOSE',data=previous_application_
merged)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}
%'.format(p.get_height()*100/len(previous_application_merged)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("NAME_CASH_LOAN_PURPOSE histogram")
```

```
Text(0.5, 1.0, 'NAME_CASH_LOAN_PURPOSE histogram')
```




```
# mOst NAME_CASH_LOAN_PURPOSE are actually Not defined

#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_CASH_LOAN_PURPOSE',hue='TARGET',data=previous
_application_merged)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}
%'.format(p.get_height()*100/len(previous_application_merged)),
(p.get_x()+0.25, p.get_height()+0.01),size=8)
ax.set_title("Targetwise percentage for each category for
NAME_CASH_LOAN_PURPOSE")
```

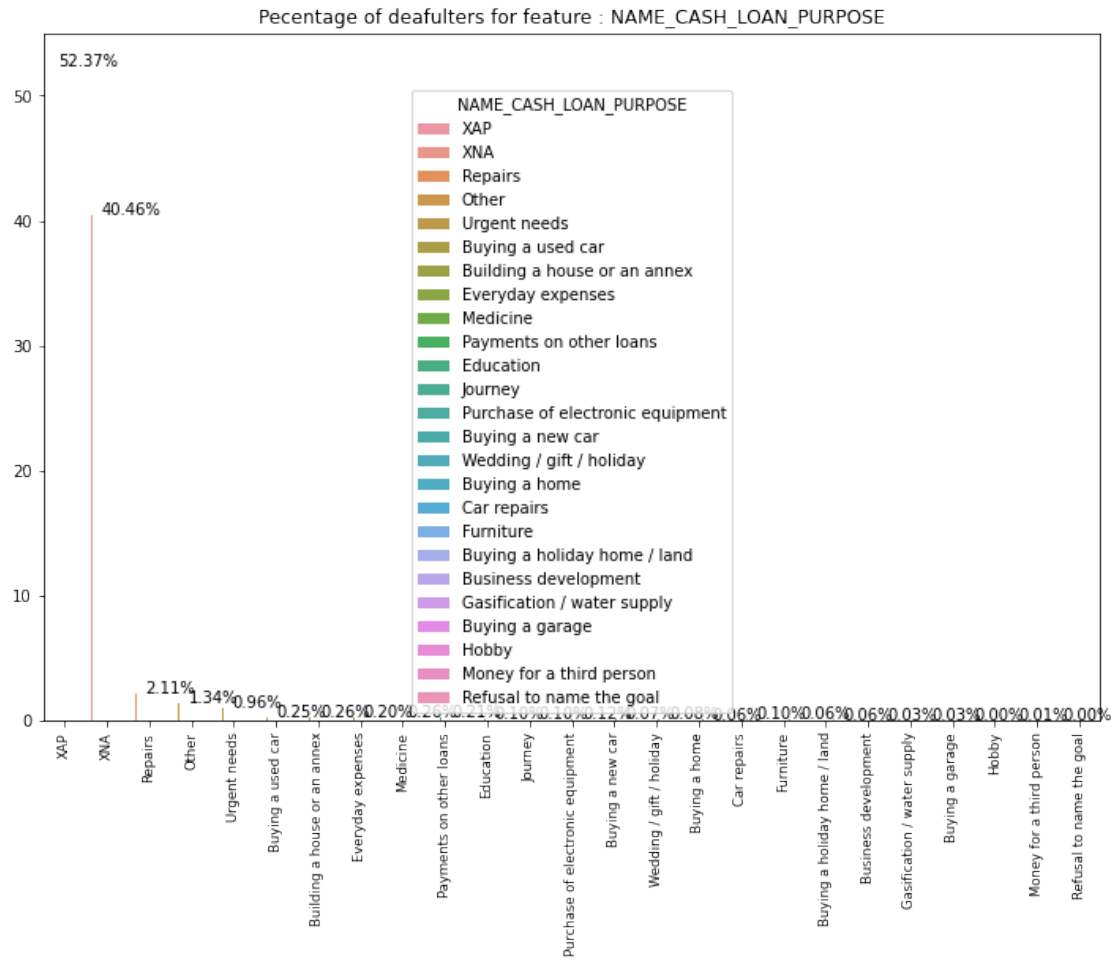
```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NAME_CASH_LOAN_PURPOSE')
```



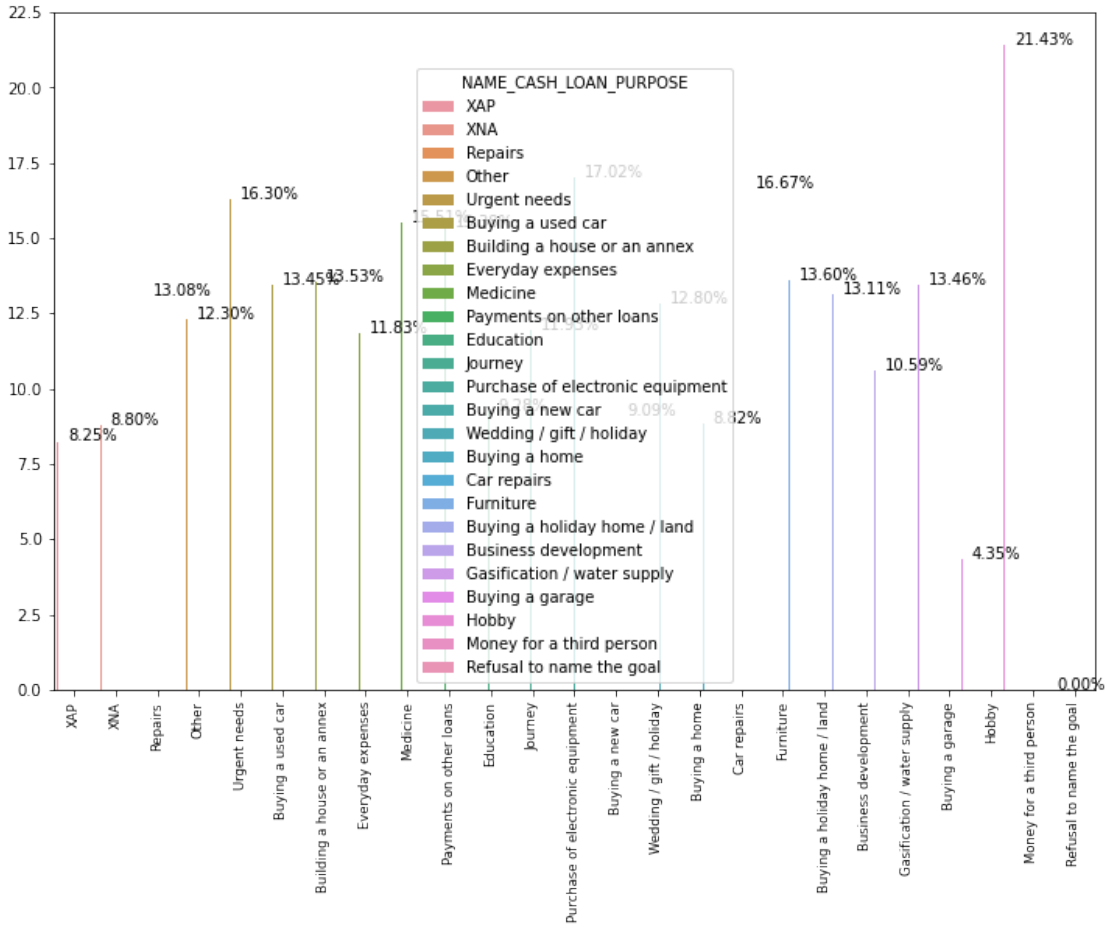

```
#lets see out of all deafulters what are the percentage of defaulters
in each category

plot_for_categorical_defaulters(previous_application_merged,'NAME_CASH
_LOAN_PURPOSE')
```

```
{'XAP': 52.36877725973831,
 'XNA': 40.45721161076854,
 'Repairs': 2.1130997142427432,
 'Other': 1.342307113851707,
 'Urgent needs': 0.9625507595127086,
 'Buying a used car': 0.25191758159121674,
 'Building a house or an annex': 0.2594375093999098,
 'Everyday expenses': 0.19927808693036547,
 'Medicine': 0.2594375093999098,
 'Payments on other loans': 0.21055797864340503,
 'Education': 0.101519025417356,
 'Journey': 0.09775906151300948,
 'Purchase of electronic equipment': 0.12031884493908858,
 'Buying a new car': 0.06767935027823734,
 'Wedding / gift / holiday': 0.07895924199127688,
 'Buying a home': 0.05639945856519778,
```

```
'Car repairs': 0.101519025417356,
'Furniture': 0.06391938637389082,
'Buying a holiday home / land': 0.06015942246954429,
'Business development': 0.03383967513911867,
'Gasification / water supply': 0.02631974733042563,
'Buying a garage': 0.003759963904346518,
'Hobby': 0.011279891713039555,
'Money for a third person': 0.0,
'Refusal to name the goal': 0.0}
```




```
#lets see in each category what is % of defaulters of same category
plot_for_deafulters_categorywise(previous_application_merged,'NAME_CASH_LOAN_PURPOSE')
```



- Here we can see XNA and XAP type loans have most defaulter rate but the catch is ,This is very less occurance in this category

CODE_REJECT_REASON

CODE_REJECT_REASON Why was the previous application rejected

```
previous_application_merged['CODE_REJECT_REASON'].value_counts()
```

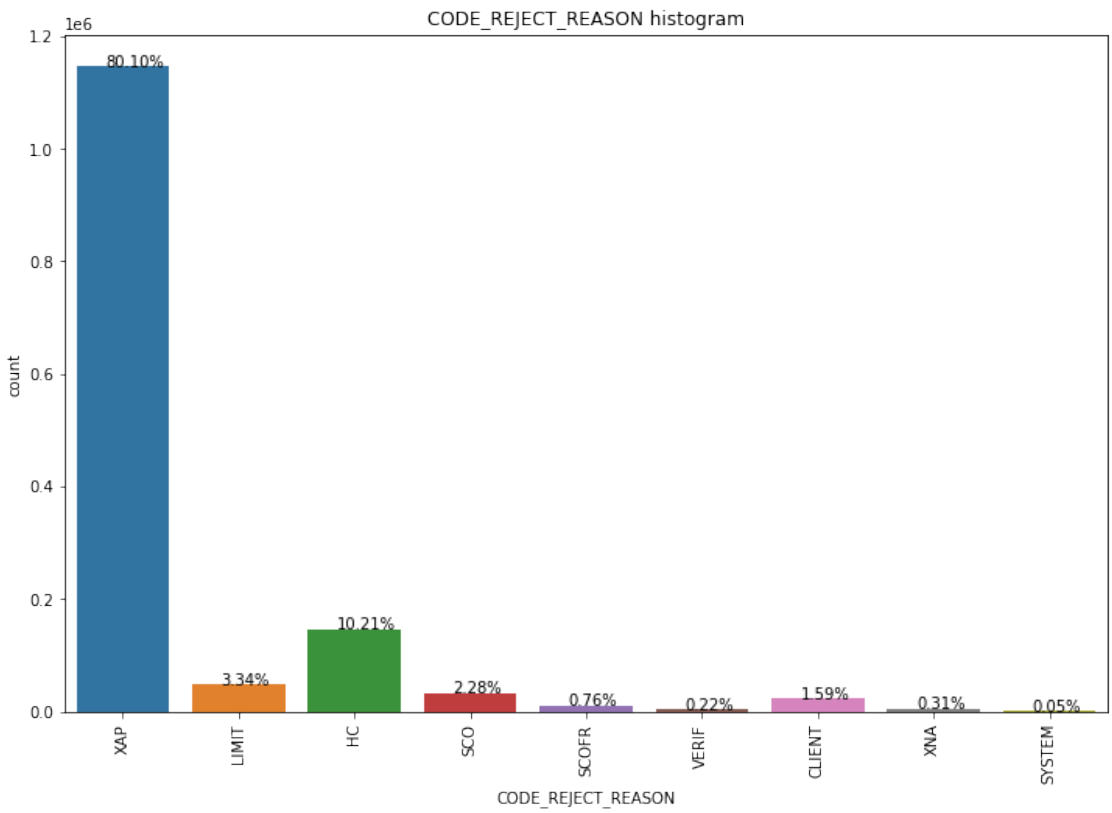
```
XAP        1145533
HC          145984
LIMIT        47773
SCO          32636
CLIENT       22771
SCOFR        10875
XNA           4378
VERIF         3079
SYSTEM         672
Name: CODE_REJECT_REASON, dtype: int64
```

```
previous_application_merged['CODE_REJECT_REASON'].isna().sum()
# NA values
```

16454

```
# 16454 NA VALUES

#lets see its distribution
plt.figure(figsize=(12,8))
ax=sns.countplot(x='CODE_REJECT_REASON',data=previous_application_merg
ed)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}
%'.format(p.get_height()*100/len(previous_application_merged)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("CODE_REJECT_REASON histogram")
```

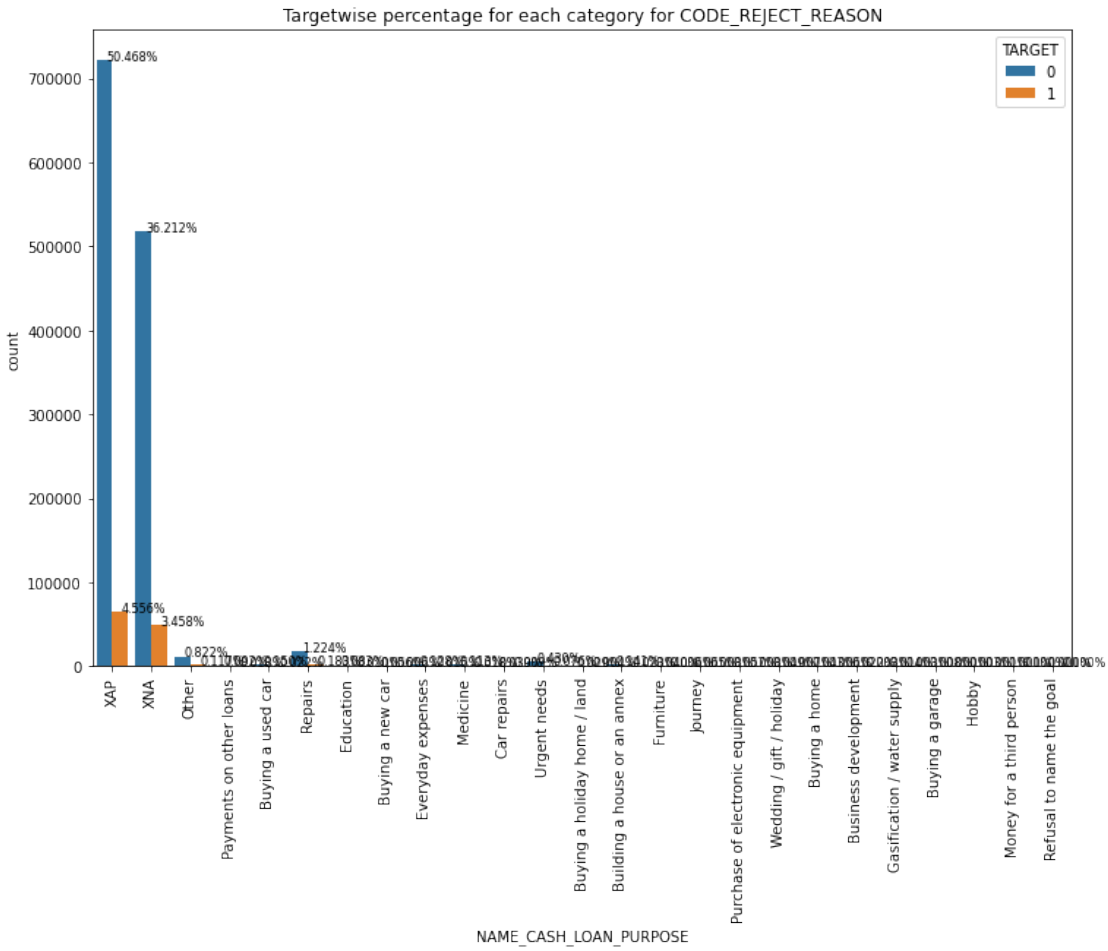
Text(0.5, 1.0, 'CODE_REJECT_REASON histogram')




```
# mOst CODE_REJECT_REASON are actually Not defined(XAP)

#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_CASH_LOAN_PURPOSE',hue='TARGET',data=previous
_application_merged)
plt.xticks(rotation=90)
```

```
for p in ax.patches:
    ax.annotate('{:.3f}
%'.format(p.get_height()*100/len(previous_application_merged)),
(p.get_x()+0.25, p.get_height()+0.01),size=8)
ax.set_title("Targetwise percentage for each category for
CODE_REJECT_REASON")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
CODE_REJECT_REASON')
```

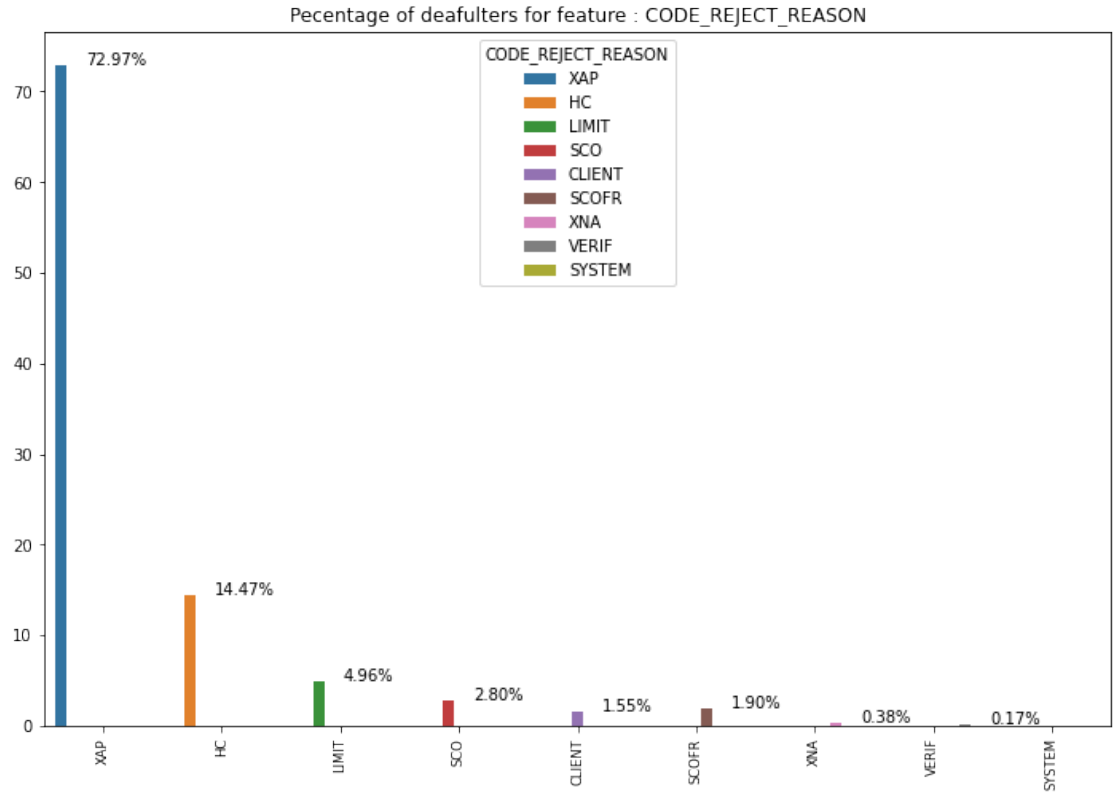



```
#lets see out of all deafulters what are the percentage of defaulters
in each category
```

```
plot_for_categorical_defaulters(previous_application_merged,'CODE_REJE
CT_REASON')
```

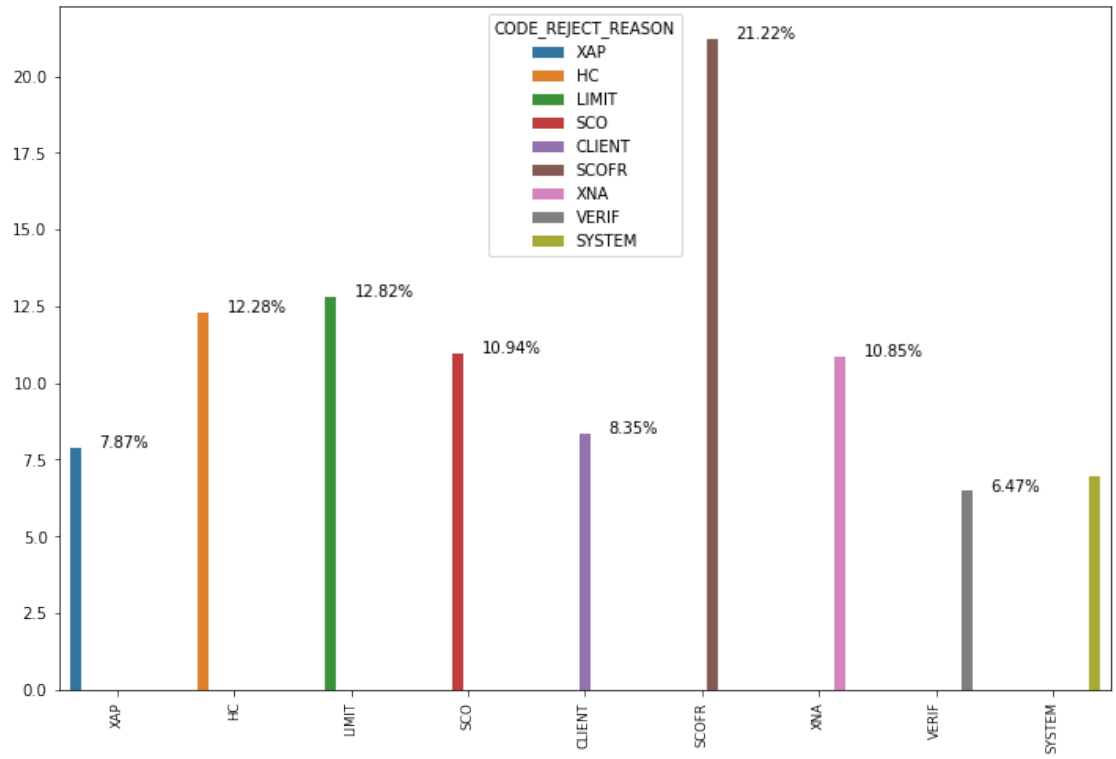
```
{'XAP': 72.96961949165288,
 'HC': 14.47210106782975,
 'LIMIT': 4.963152353737404,
 'SCO': 2.804933072642503,
 'CLIENT': 1.552865092495112,
 'SCOFR': 1.9025417355993381,
```

```
'XNA': 0.37599639043465183,
'VERIF': 0.16919837569559332,
'SYSTEM': 0.03759963904346518}
```



- 72% of defaulters belongs to XAP CODE_REJECT_REASON category

```
#lets see in each category what is % of defaulters of same category
plot_for_deafulters_categorywise(previous_application_merged,'CODE_REJ
ECT_REASON')
```



- we can see SCOFR have 22% defaulters in its own category

PRODUCT_COMBINATION

Detailed product combination of the previous application

```
previous_application_merged['PRODUCT_COMBINATION'].value_counts()
```

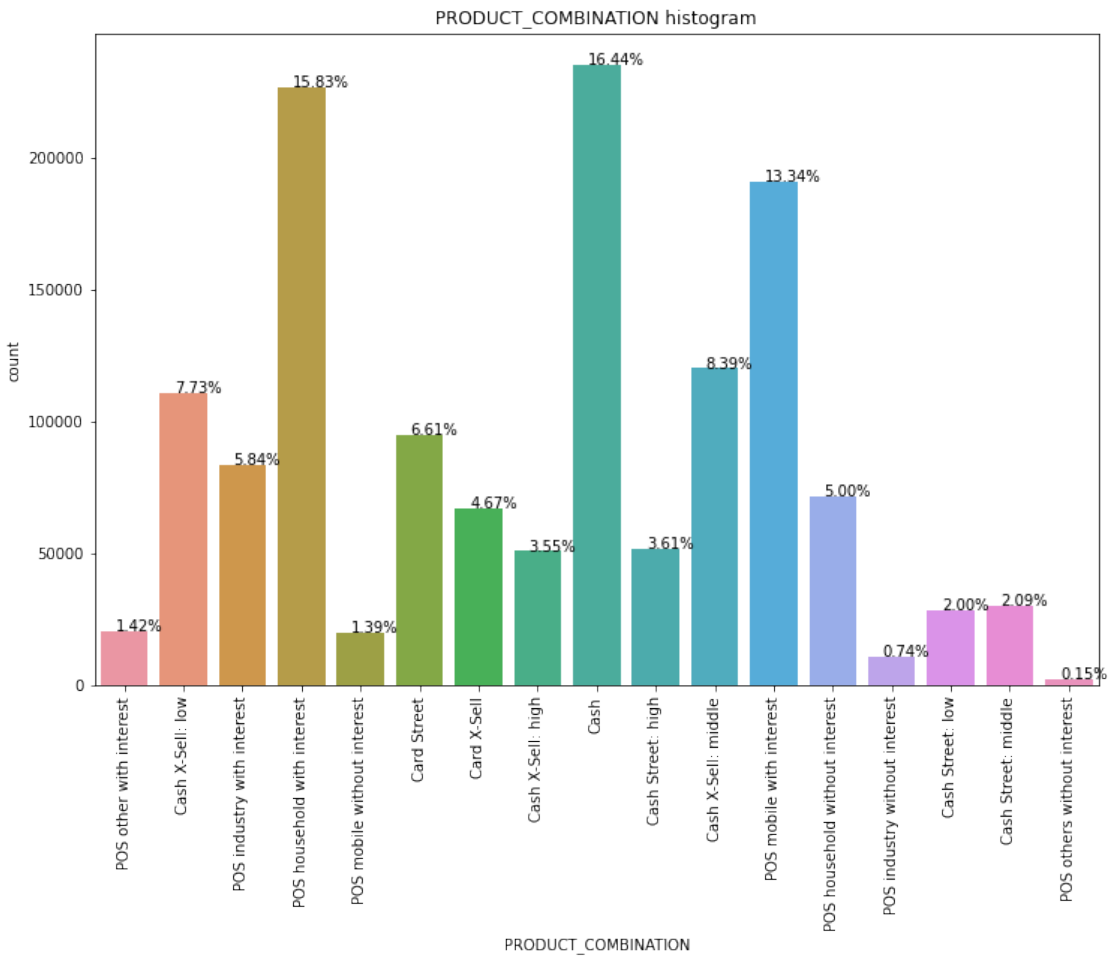
```
Cash                              235101
POS household with interest       226464
POS mobile with interest          190850
Cash X-Sell: middle               120036
Cash X-Sell: low                  110599
Card Street                        94594
POS industry with interest         83508
POS household without interest     71519
Card X-Sell                        66774
Cash Street: high                  51699
Cash X-Sell: high                  50802
Cash Street: middle                29934
Cash Street: low                   28593
POS other with interest            20309
POS mobile without interest        19890
POS industry without interest      10621
POS others without interest         2095
Name: PRODUCT_COMBINATION, dtype: int64
```

```
previous_application_merged['PRODUCT_COMBINATION'].isna().sum()
# NA values
```

16767

```
# 16767 NA VALUES

#lets see its distribution
plt.figure(figsize=(12,8))
ax=sns.countplot(x='PRODUCT_COMBINATION',data=previous_application_mer
ged)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}
%'.format(p.get_height()*100/len(previous_application_merged)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("PRODUCT_COMBINATION histogram")
```

Text(0.5, 1.0, 'PRODUCT_COMBINATION histogram')

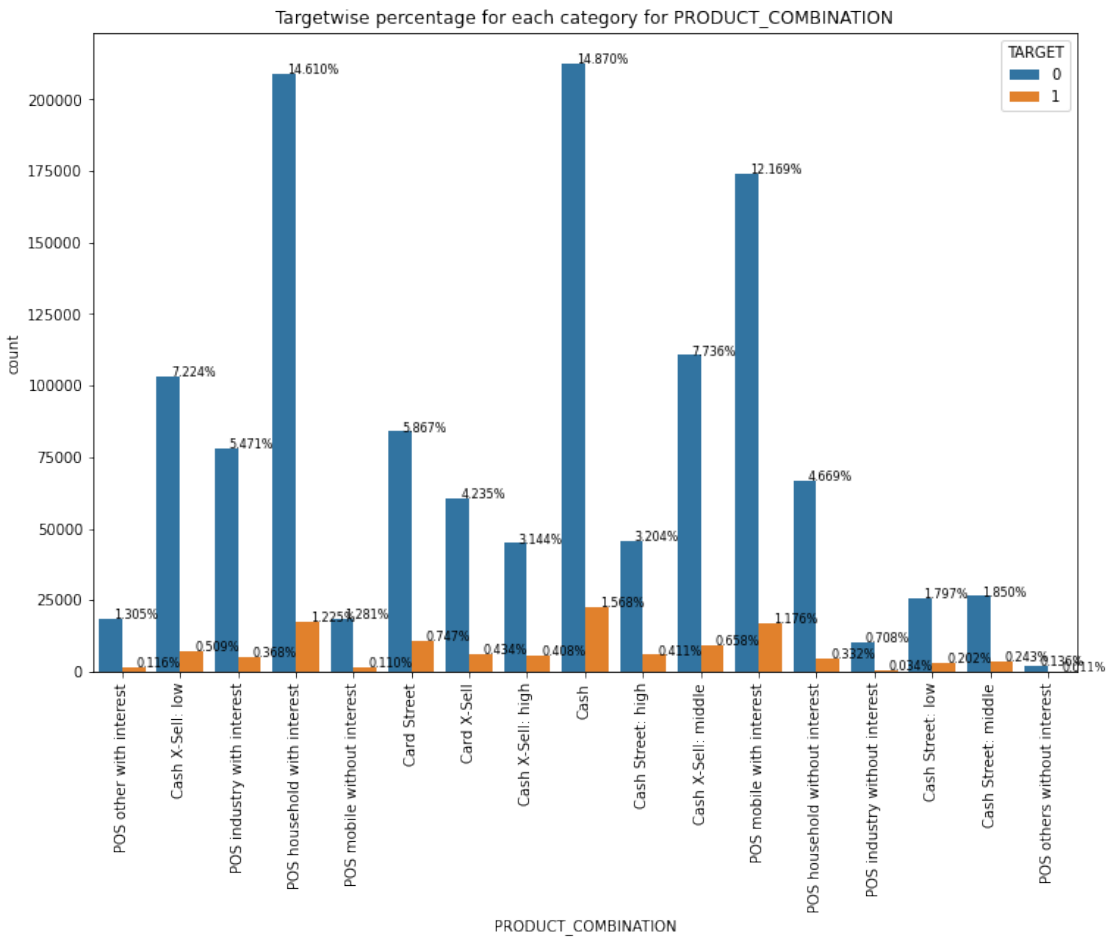


- Most PRODUCT_COMBINATION are actually cash ,POS industry with interest,POS mobile without interest

POS(Point OF Sale)

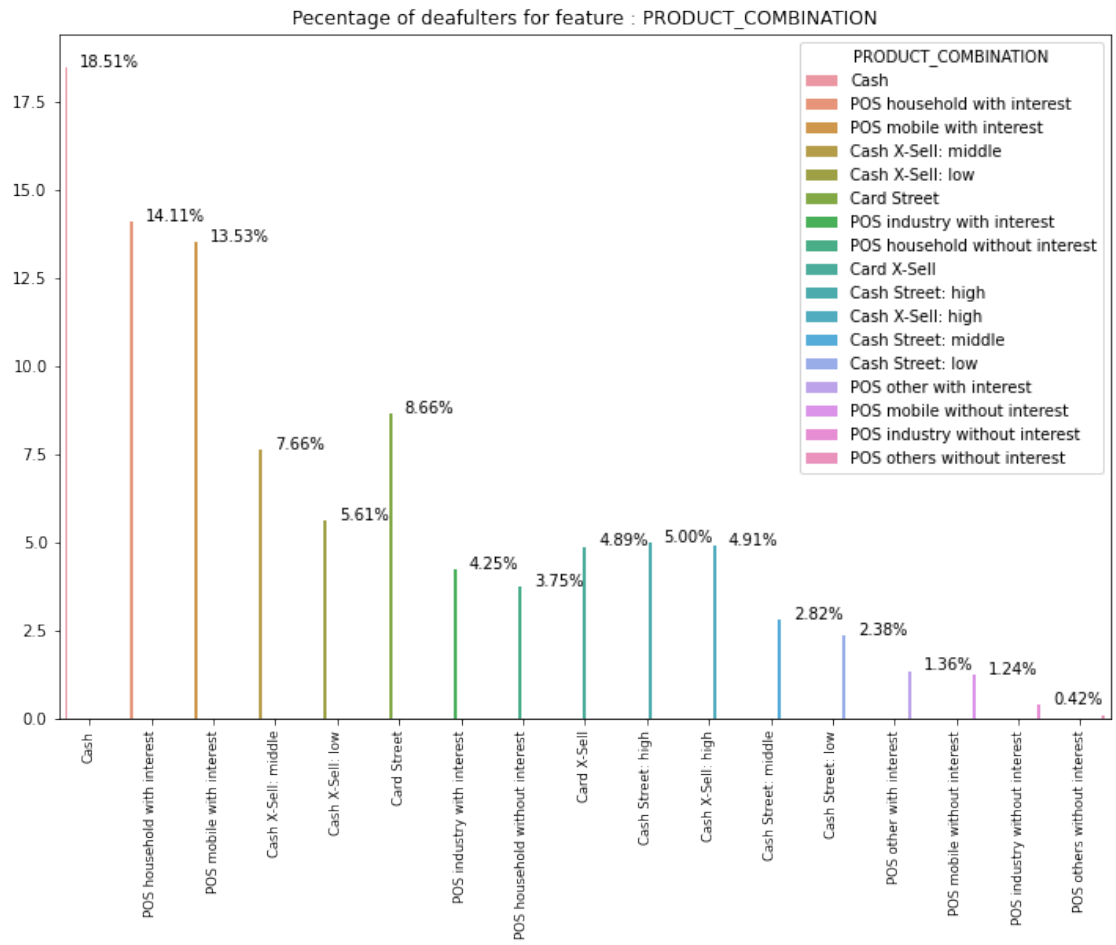
```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='PRODUCT_COMBINATION',hue='TARGET',data=previous_ap
plication_merged)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}
%'.format(p.get_height()*100/len(previous_application_merged)),
(p.get_x()+0.25, p.get_height()+0.01),size=8)
ax.set_title("Targetwise percentage for each category for
PRODUCT_COMBINATION")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
PRODUCT_COMBINATION')
```



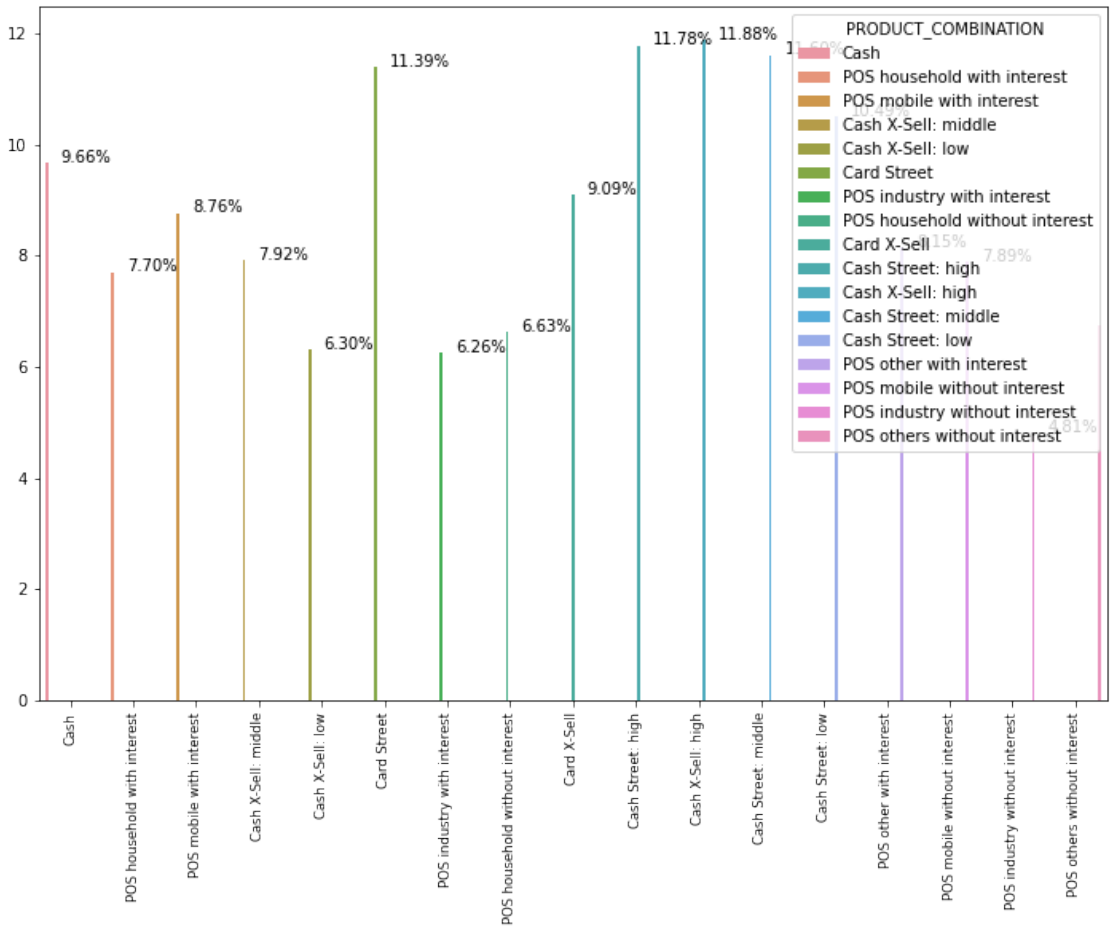

```
#lets see out of all deafulters what are the percentage of defaulters
in each category

plot_for_categorical_defaulters(previous_application_merged,'PRODUCT_C
OMBINATION')
```

```
{'Cash': 18.50654233719356,
 'POS household with interest': 14.107384569108136,
 'POS mobile with interest': 13.528350127838772,
 'Cash X-Sell: middle': 7.655286509249511,
 'Cash X-Sell: low': 5.613626109189352,
 'Card Street': 8.655436907805685,
 'POS industry with interest': 4.248759211911565,
 'POS household without interest': 3.7486840126334786,
 'Card X-Sell': 4.887953075650474,
 'Cash Street: high': 4.996992028876523,
 'Cash X-Sell: high': 4.906752895172207,
 'Cash Street: middle': 2.823732892164235,
 'Cash Street: low': 2.3762971875469994,
 'POS other with interest': 1.3573469694690932,
 'POS mobile without interest': 1.2370281245300045,
 'POS industry without interest': 0.41735599338246354,
 'POS others without interest': 0.11279891713039555}
```



- 18% of defaulters belongs to CASH loans for PRODUCT_COMBINATION category
- 14 and 11 percentage of defaulters belong to Household and Mobile with interest

```
#lets see in each category what is % of defaulters of same category
plot_for_deafulters_categorywise(previous_application_merged,'PRODUCT_
COMBINATION')
```



- we can see Cash,cash Street, Cash X sell -HIGH and ,cash Street-Low are the category for which there are more deaulters in their categiry

```
#Lets see how many columns are continous or numerical

numerics = ['int16', 'int32', 'int64', 'float16', 'float32',
'float64']
prev_app_continous_variables=previous_application_merged.select_dtypes
(include=numerics).columns.to_list()
len(prev_app_continous_variables)
```

22

```
for i in prev_app_continous_variables:
    print(f"Standard deviation for feature {i}:
{previous_application_merged[i].std()}")
    print(f"Non na values in feature {i} is :
{previous_application_merged[i].notnull().sum()} ")
    print("="*100)
```

```
Standard deviation for feature SK_ID_CURR: 102802.09533975559
Non na values in feature SK_ID_CURR is : 1430155
==========================================================================
==============================
Standard deviation for feature TARGET: 0.2807217551124661
Non na values in feature TARGET is : 1430155
==========================================================================
==============================
Standard deviation for feature SK_ID_PREV: 532715.3352547982
Non na values in feature SK_ID_PREV is : 1413701
==========================================================================
==============================
Standard deviation for feature AMT_ANNUITY: 14724.90689731526
Non na values in feature AMT_ANNUITY is : 1106483
==========================================================================
==============================
Standard deviation for feature AMT_APPLICATION: 293622.1931407793
Non na values in feature AMT_APPLICATION is : 1413701
==========================================================================
==============================
Standard deviation for feature AMT_CREDIT: 319481.2507159836
Non na values in feature AMT_CREDIT is : 1413700
==========================================================================
==============================
Standard deviation for feature AMT_DOWN_PAYMENT: 20620.300439618943
Non na values in feature AMT_DOWN_PAYMENT is : 664161
==========================================================================
==============================
Standard deviation for feature AMT_GOODS_PRICE: 315937.6041998381
Non na values in feature AMT_GOODS_PRICE is : 1094176
==========================================================================
==============================
Standard deviation for feature HOUR_APPR_PROCESS_START:
3.3314738452312067
Non na values in feature HOUR_APPR_PROCESS_START is : 1413701
==========================================================================
==============================
Standard deviation for feature NFLAG_LAST_APPL_IN_DAY:
0.05962559916667728
Non na values in feature NFLAG_LAST_APPL_IN_DAY is : 1413701
==========================================================================
==============================
Standard deviation for feature RATE_DOWN_PAYMENT: 0.10778366847594059
Non na values in feature RATE_DOWN_PAYMENT is : 664161
==========================================================================
==============================
Standard deviation for feature RATE_INTEREST_PRIMARY:
0.09084907783929053
Non na values in feature RATE_INTEREST_PRIMARY is : 4791
==========================================================================
```

```
==============================
Standard deviation for feature RATE_INTEREST_PRIVILEGED:
0.10064378932855686
Non na values in feature RATE_INTEREST_PRIVILEGED is : 4791
==========================================================================
==============================
Standard deviation for feature DAYS_DECISION: 783.540170754335
Non na values in feature DAYS_DECISION is : 1413701
==========================================================================
==============================
Standard deviation for feature SELLERPLACE_AREA: 7695.0816641697775
Non na values in feature SELLERPLACE_AREA is : 1413701
==========================================================================
==============================
Standard deviation for feature CNT_PAYMENT: 14.566745731143895
Non na values in feature CNT_PAYMENT is : 1106488
==========================================================================
==============================
Standard deviation for feature DAYS_FIRST_DRAWING: 88832.26659762385
Non na values in feature DAYS_FIRST_DRAWING is : 852595
==========================================================================
==============================
Standard deviation for feature DAYS_FIRST_DUE: 71650.14714587011
Non na values in feature DAYS_FIRST_DUE is : 852595
==========================================================================
==============================
Standard deviation for feature DAYS_LAST_DUE_1ST_VERSION:
106161.77593320515
Non na values in feature DAYS_LAST_DUE_1ST_VERSION is : 852595
==========================================================================
==============================
Standard deviation for feature DAYS_LAST_DUE: 149704.716371441
Non na values in feature DAYS_LAST_DUE is : 852595
==========================================================================
==============================
Standard deviation for feature DAYS_TERMINATION: 153537.06427434084
Non na values in feature DAYS_TERMINATION is : 852595
==========================================================================
==============================
Standard deviation for feature NFLAG_INSURED_ON_APPROVAL:
0.470419385998328
Non na values in feature NFLAG_INSURED_ON_APPROVAL is : 852595
==========================================================================
==============================
```

So we will take some columns which have significant values of NOT NULL and a good variance and analyze it

HOUR_APPR_PROCESS_START

Approximately at what day hour did the client apply for the previous application

```
previous_application_merged['HOUR_APPR_PROCESS_START'].describe()
```

```
count    1.413701e+06
mean     1.247893e+01
std      3.331474e+00
min      0.000000e+00
25%      1.000000e+01
50%      1.200000e+01
75%      1.500000e+01
max      2.300000e+01
Name: HOUR_APPR_PROCESS_START, dtype: float64
```

```
previous_application_merged['HOUR_APPR_PROCESS_START'].value_counts()
```

```
11.0    163264
12.0    157528
10.0    153800
13.0    145838
14.0    133621
15.0    120895
9.0     107696
16.0    102526
17.0     80267
8.0      62317
18.0     53948
7.0      38846
19.0     28726
6.0      21705
5.0      12996
20.0     12260
4.0       7861
3.0       4182
21.0      3422
2.0        950
22.0       618
1.0        180
23.0       163
0.0         92
Name: HOUR_APPR_PROCESS_START, dtype: int64
```

```
previous_application_merged['HOUR_APPR_PROCESS_START'].isna().sum()
```

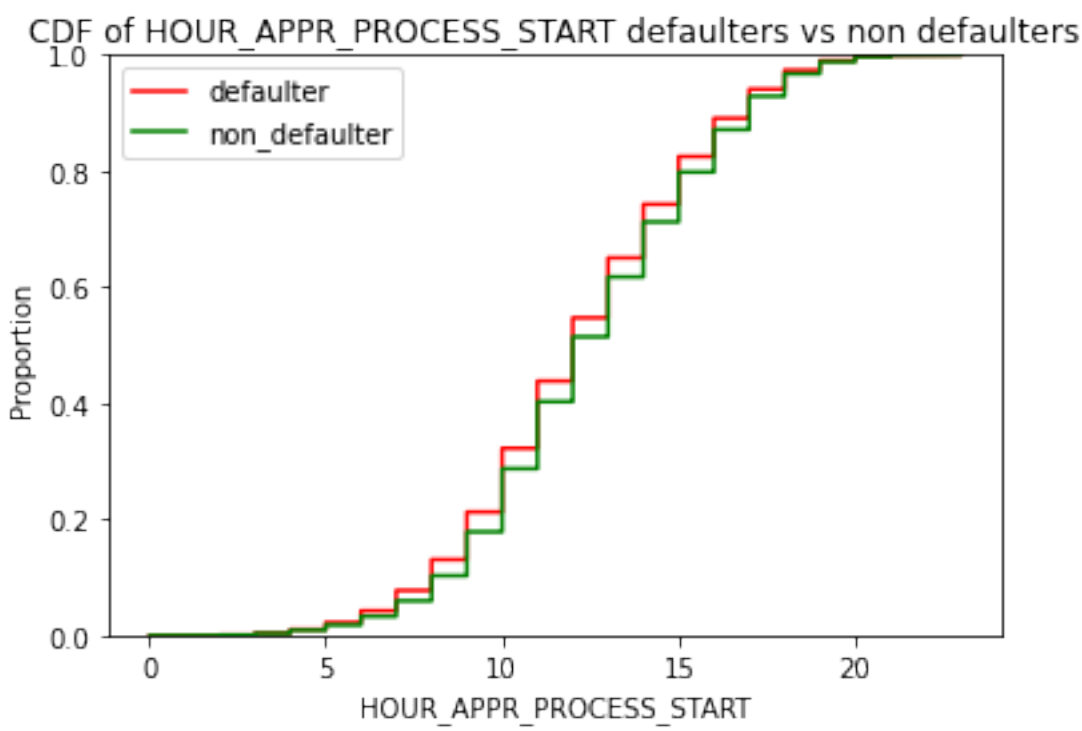
```
16454
```

```
# total 16454 are null

previous_application_merged['HOUR_APPR_PROCESS_START'].notnull().sum()
```

```
1413701
```

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(previous_application_merged[previous_application_merged['
TARGET']==1]['HOUR_APPR_PROCESS_START'],label='defaulter',color='red')
sns.ecdfplot(previous_application_merged[previous_application_merged['
TARGET']==0]
['HOUR_APPR_PROCESS_START'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of HOUR_APPR_PROCESS_START defaulters vs non
defaulters")
plt.legend()
#plt.axis('off')
```

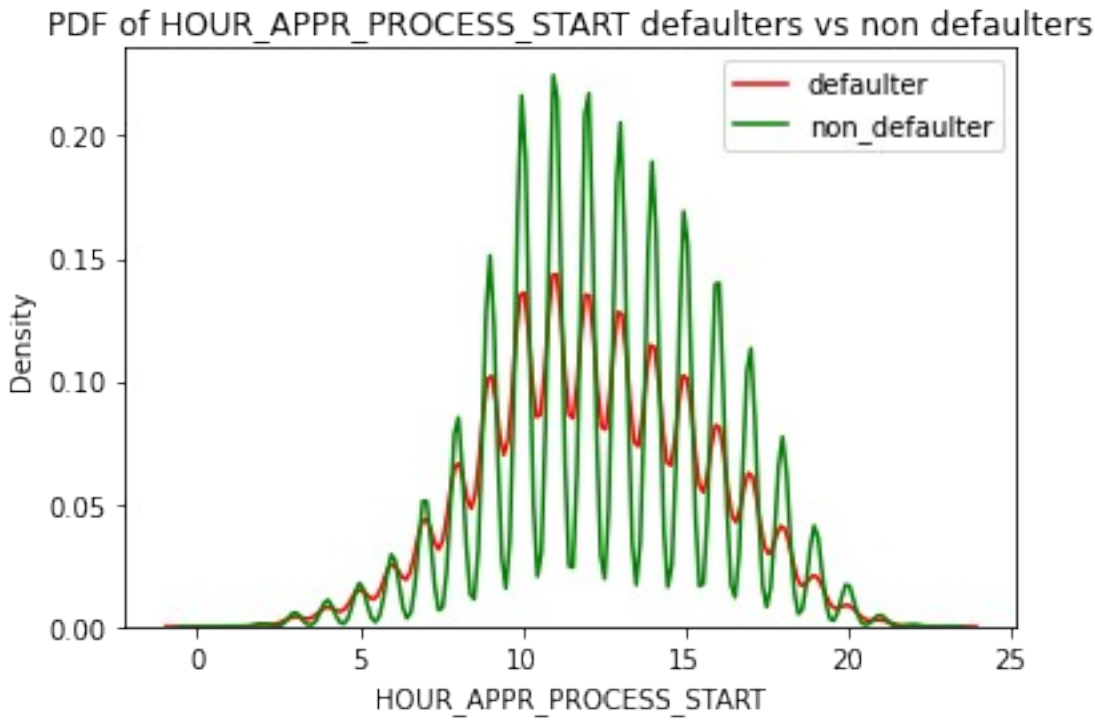
```
<matplotlib.legend.Legend at 0x1d126d1bcd0>
```




```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(previous_application_merged[previous_application_merged['T
ARGET']==1]['HOUR_APPR_PROCESS_START'],label='defaulter',color='red')
sns.kdeplot(previous_application_merged[previous_application_merged['T
ARGET']==0]
['HOUR_APPR_PROCESS_START'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of HOUR_APPR_PROCESS_START defaulters vs non
defaulters")
```

```
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d12ad94970>
```



- I think this just time and non significant actucally

DAYS_DECISION

Relative to current application when was the decision about previous application made

```
previous_application_merged['DAYS_DECISION'].describe()
```

```
count    1.413701e+06
mean    -8.803670e+02
std      7.835402e+02
min     -2.922000e+03
25%     -1.313000e+03
50%     -5.820000e+02
75%     -2.710000e+02
max     -1.000000e+00
Name: DAYS_DECISION, dtype: float64
```

```
previous_application_merged['DAYS_DECISION'].isna().sum()
```

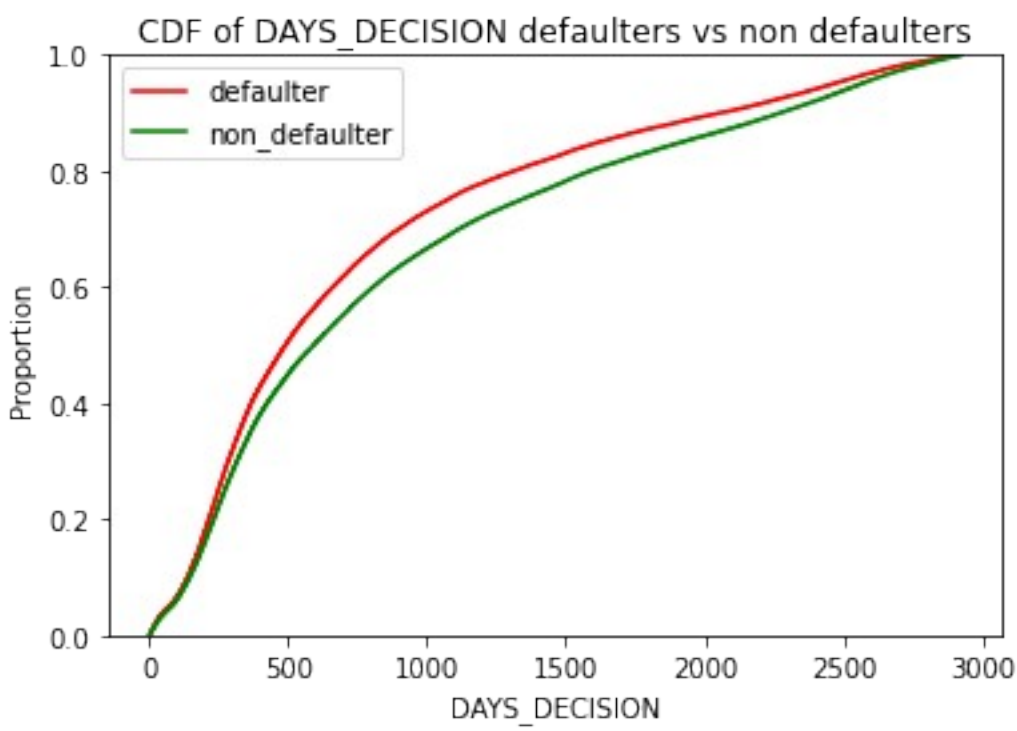
```
16454
```

```
# total 16454 are null

previous_application_merged['DAYS_DECISION'].notnull().sum()
```

```
1413701
```

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(-
1*previous_application_merged[previous_application_merged['TARGET']==1
]['DAYS_DECISION'],label='defaulter',color='red')
sns.ecdfplot(-
1*previous_application_merged[previous_application_merged['TARGET']==0
]['DAYS_DECISION'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of DAYS_DECISION defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d080670d90>
```

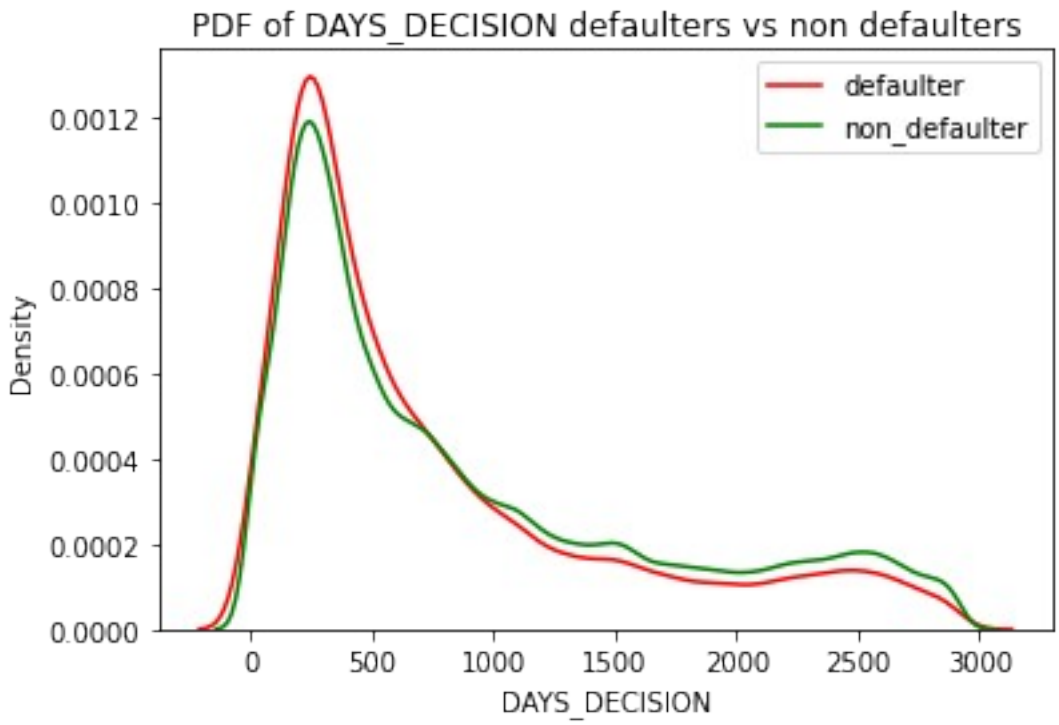


- DAYS_DECISION seems to be more for defaulters compare to non defaulters

```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(-
1*previous_application_merged[previous_application_merged['TARGET']==1
]['DAYS_DECISION'],label='defaulter',color='red')
sns.kdeplot(-
1*previous_application_merged[previous_application_merged['TARGET']==0
]['DAYS_DECISION'],label='non_defaulter',color='green')
```

```
#plt.ylim([0,1000])
plt.title("PDF of DAYS_DECISION defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

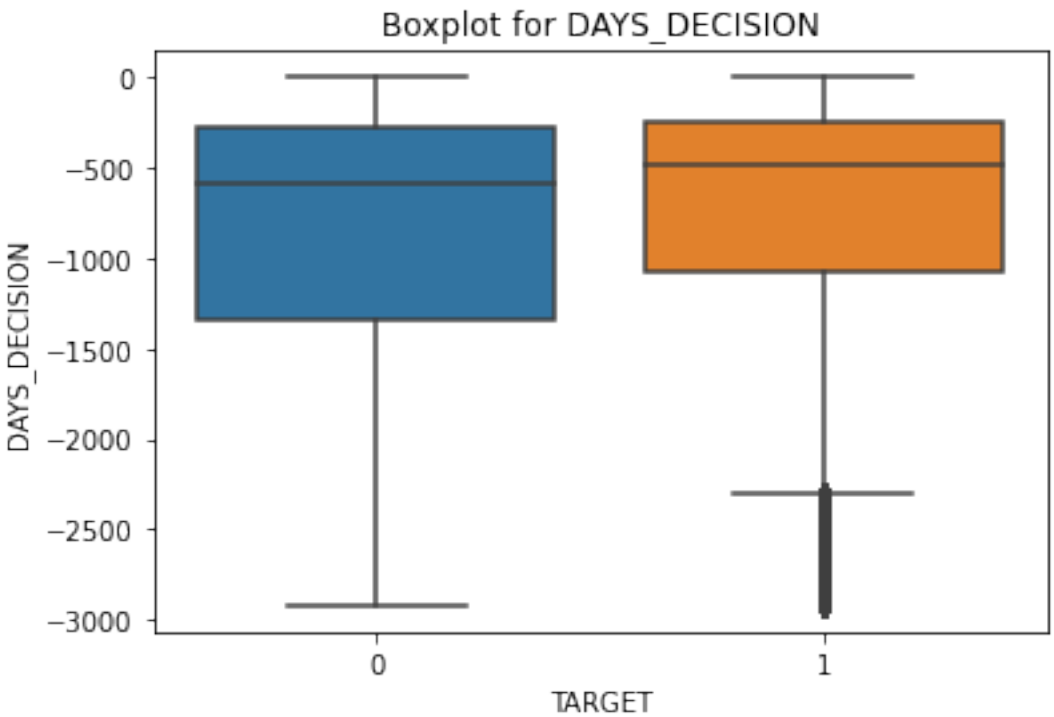
```
<matplotlib.legend.Legend at 0x1d298d74880>
```



- For DAYS_DECISION values above -1000 density of defaulters is slightly more and below -1000 non defaulters density is more

```
sns.boxplot(x='TARGET',y='DAYS_DECISION',data=previous_application_mer
ged)
#plt.ylim([0.0,50000])
plt.title("Boxplot for DAYS_DECISION")
```

```
Text(0.5, 1.0, 'Boxplot for DAYS_DECISION')
```



- There are some outliers for deafulters for DAYS_DECISION where normally non defaulters should lie but some defaulters are also there

CNT_PAYMENT

Term of previous credit at application of the previous application

```
previous_application_merged['CNT_PAYMENT'].describe()
```

```
count    1.106488e+06
mean     1.605194e+01
std      1.456675e+01
min      0.000000e+00
25%      6.000000e+00
50%      1.200000e+01
75%      2.400000e+01
max      8.400000e+01
Name: CNT_PAYMENT, dtype: float64
```

```
previous_application_merged['CNT_PAYMENT'].isna().sum()
```

```
323667
```

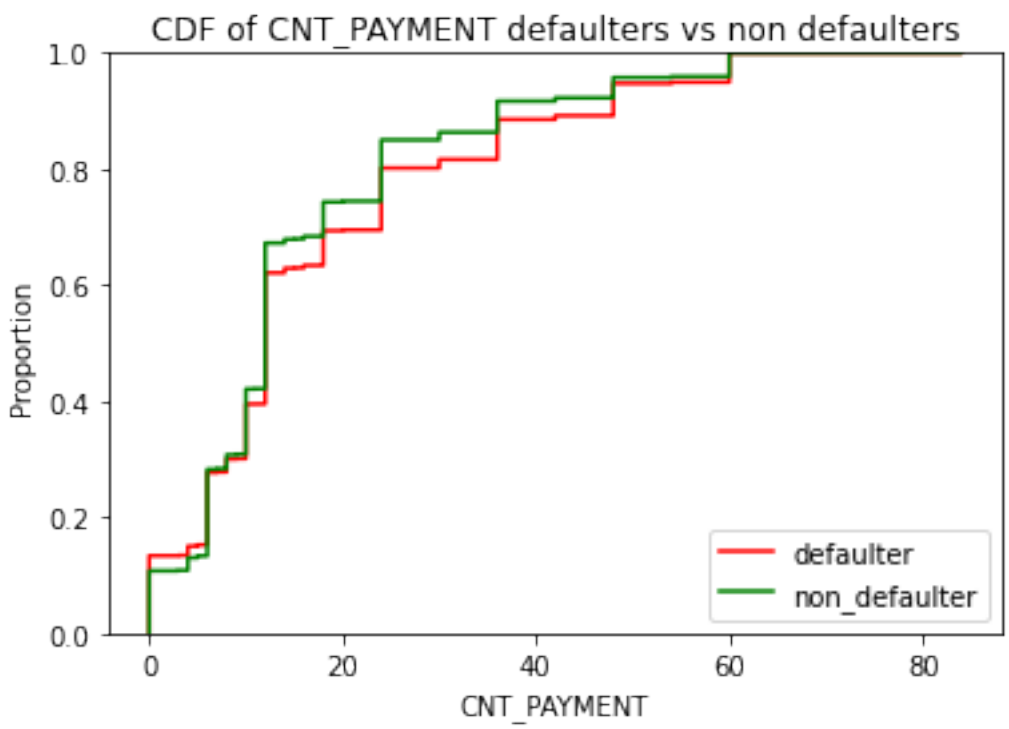
```
# total 323667 are null
```

```
previous_application_merged['CNT_PAYMENT'].notnull().sum()
```

```
1106488
```

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(previous_application_merged[previous_application_merged['
TARGET']==1]['CNT_PAYMENT'],label='defaulter',color='red')
sns.ecdfplot(previous_application_merged[previous_application_merged['
TARGET']==0]['CNT_PAYMENT'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of CNT_PAYMENT defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

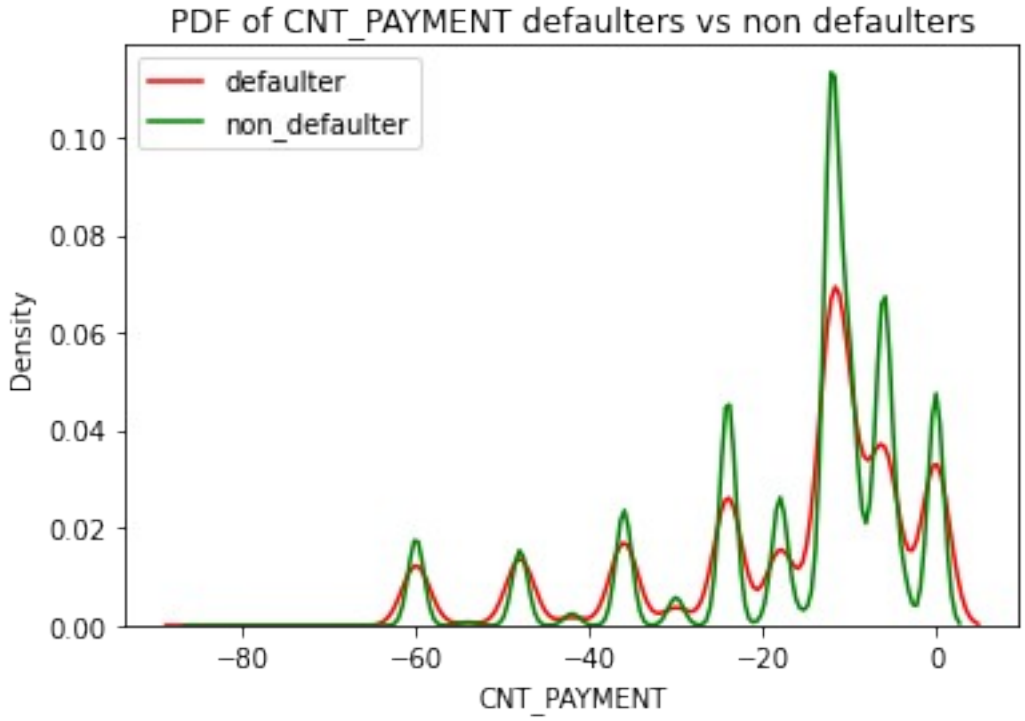
```
<matplotlib.legend.Legend at 0x1d080691bb0>
```



- CNT_PAYMENT seems to be little lesser for defaulters compare to non defaulters

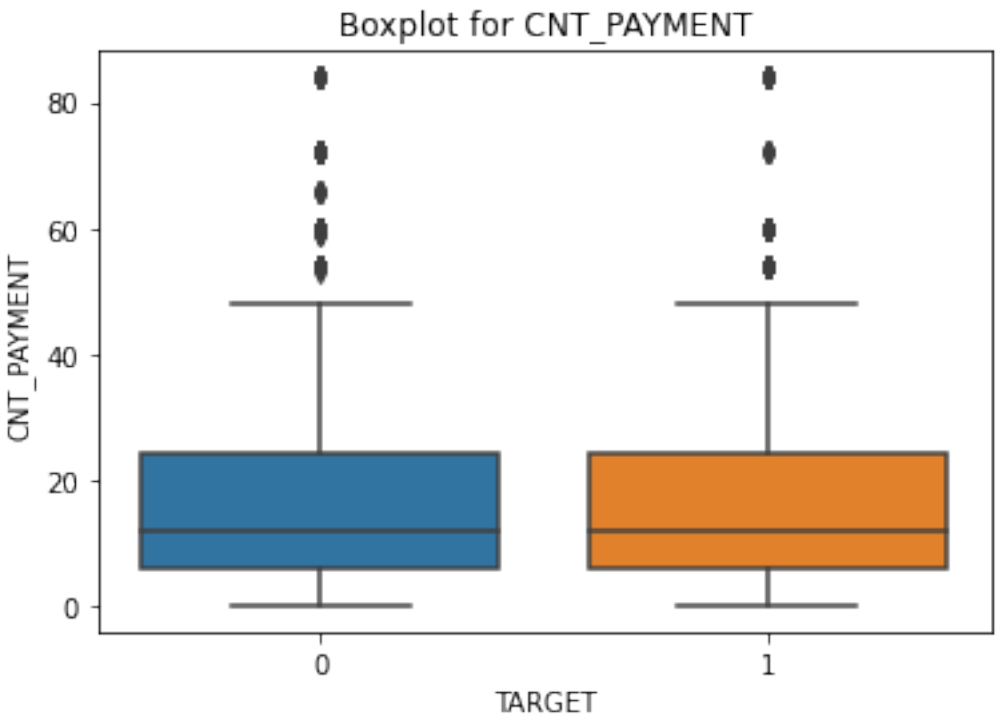
```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(-
1*previous_application_merged[previous_application_merged['TARGET']==1
]['CNT_PAYMENT'],label='defaulter',color='red')
sns.kdeplot(-
1*previous_application_merged[previous_application_merged['TARGET']==0
]['CNT_PAYMENT'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of CNT_PAYMENT defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d2a22c2070>
```




```
sns.boxplot(x='TARGET',y='CNT_PAYMENT',data=previous_application_merge
d)
#plt.ylim([0.0,50000])
plt.title("Boxplot for CNT_PAYMENT")
```

```
Text(0.5, 1.0, 'Boxplot for CNT_PAYMENT')
```

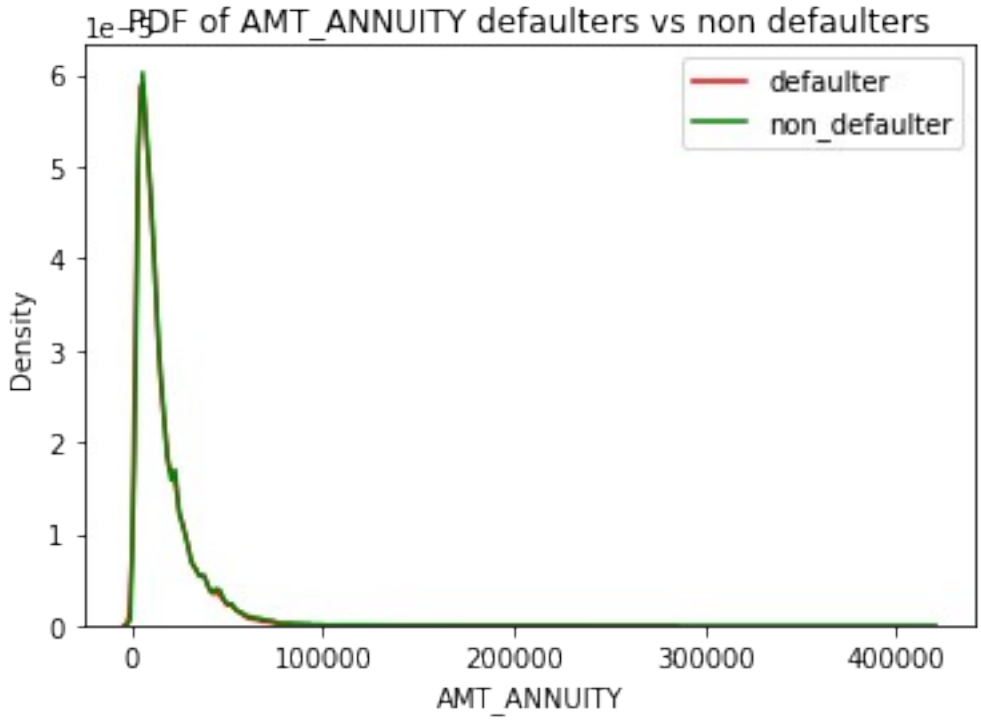


This looks almost same,we can draw any conclusion

Lets check a column which have very high standard deviation :AMT_ANNUITY

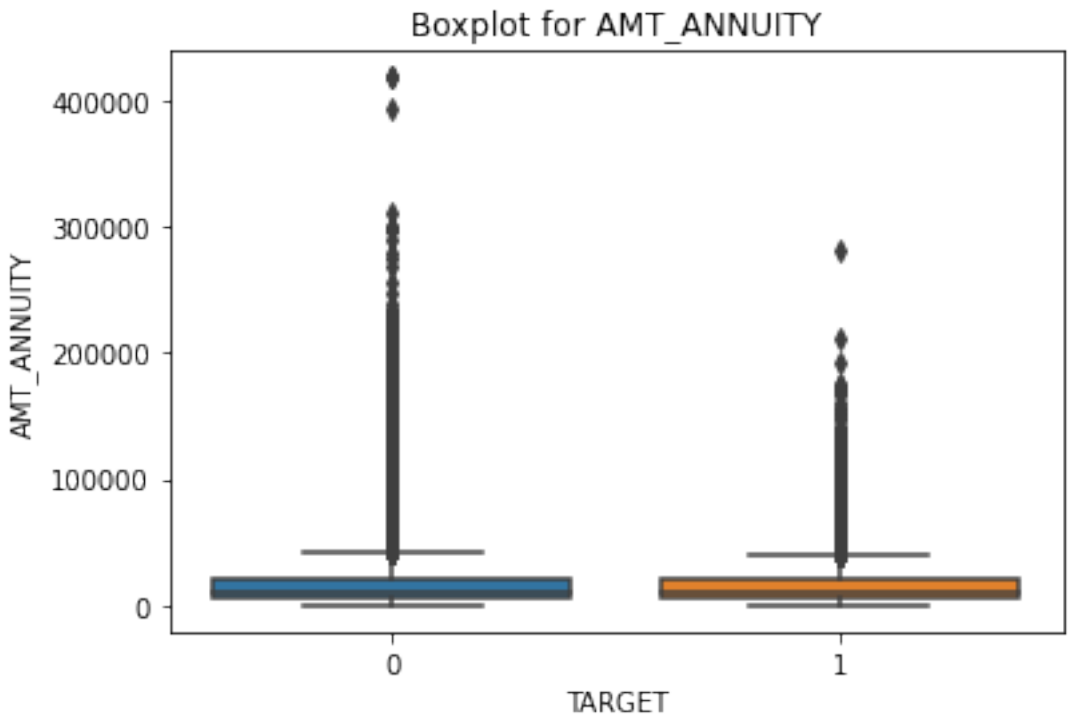
```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(previous_application_merged[previous_application_merged['T
ARGET']==1]['AMT_ANNUITY'],label='defaulter',color='red')
sns.kdeplot(previous_application_merged[previous_application_merged['T
ARGET']==0]['AMT_ANNUITY'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of AMT_ANNUITY defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d080a59bb0>
```




```
sns.boxplot(x='TARGET',y='AMT_ANNUITY',data=previous_application_merge
d)
#plt.ylim([0.0,50000])
plt.title("Boxplot for AMT_ANNUITY")
```

```
Text(0.5, 1.0, 'Boxplot for AMT_ANNUITY')
```




```
del previous_application_merged
```

we can see here ,there are very high number of outliers which dont follow their distribution ,due to which the variance is too high

This is same with other columns which have HIGH Standard deviations.

POS_CASH_balance.csv

- This table contains Monthly balance of client's previous credit loans in Home Credit along with behavioral data

```
col_descriptions[col_descriptions['Table']=='POS_CASH_balance.csv']
```

```
                    Table                     Row  \
145  POS_CASH_balance.csv            SK_ID_PREV
146  POS_CASH_balance.csv             SK_ID_CURR
147  POS_CASH_balance.csv         MONTHS_BALANCE
148  POS_CASH_balance.csv         CNT_INSTALMENT
149  POS_CASH_balance.csv  CNT_INSTALMENT_FUTURE
150  POS_CASH_balance.csv   NAME_CONTRACT_STATUS
151  POS_CASH_balance.csv                 SK_DPD
152  POS_CASH_balance.csv             SK_DPD_DEF


Description  \
145
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)
146
ID of loan in our sample
147  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 )
148
Term of previous credit (can change over time)
149
Installments left to pay on the previous credit
150
Contract status during the month
151
DPD (days past due) during the month of previous credit
152
DPD during the month with tolerance (debts with low loan amounts are
ignored) of the previous credit

                                   Special
145                                    NaN
146                                    NaN
147  time only relative to the application
148                                    NaN
149                                    NaN
150                                    NaN
151                                    NaN
152                                    NaN
```

```
POS_CASH_balance.head()
```

```
   SK_ID_PREV  SK_ID_CURR  MONTHS_BALANCE  CNT_INSTALMENT  \
0     1803195      182943             -31            48.0
1     1715348      367990             -33            36.0
2     1784872      397406             -32            12.0
3     1903291      269225             -35            48.0
4     2341044      334279             -35            36.0

   CNT_INSTALMENT_FUTURE NAME_CONTRACT_STATUS  SK_DPD  SK_DPD_DEF
0                   45.0               Active       0           0
1                   35.0               Active       0           0
2                    9.0               Active       0           0
3                   42.0               Active       0           0
4                   35.0               Active       0           0
```

```
POS_CASH_balance.shape
```

```
(10001358, 8)
```

- 10.01 M rows and 8 columns

```python
for i in POS_CASH_balance.columns:
    print(f"For feature {i}")
    print(f"Total Uniuqe values are: {POS_CASH_balance[i].nunique()}")
    print(f"Total NA values are: {POS_CASH_balance[i].isna().sum()}")
    print("="*100)
    print("\n \n")
```

```
For feature SK_ID_PREV
Total Uniuqe values are: 936325
Total NA values are: 0
==========================================================================
==============================



For feature SK_ID_CURR
Total Uniuqe values are: 337252
Total NA values are: 0
==========================================================================
==============================



For feature MONTHS_BALANCE
Total Uniuqe values are: 96
Total NA values are: 0
==========================================================================
==============================



For feature CNT_INSTALMENT
Total Uniuqe values are: 73
Total NA values are: 26071
==========================================================================
==============================



For feature CNT_INSTALMENT_FUTURE
Total Uniuqe values are: 79
Total NA values are: 26087
==========================================================================
==============================



For feature NAME_CONTRACT_STATUS
Total Uniuqe values are: 9
```

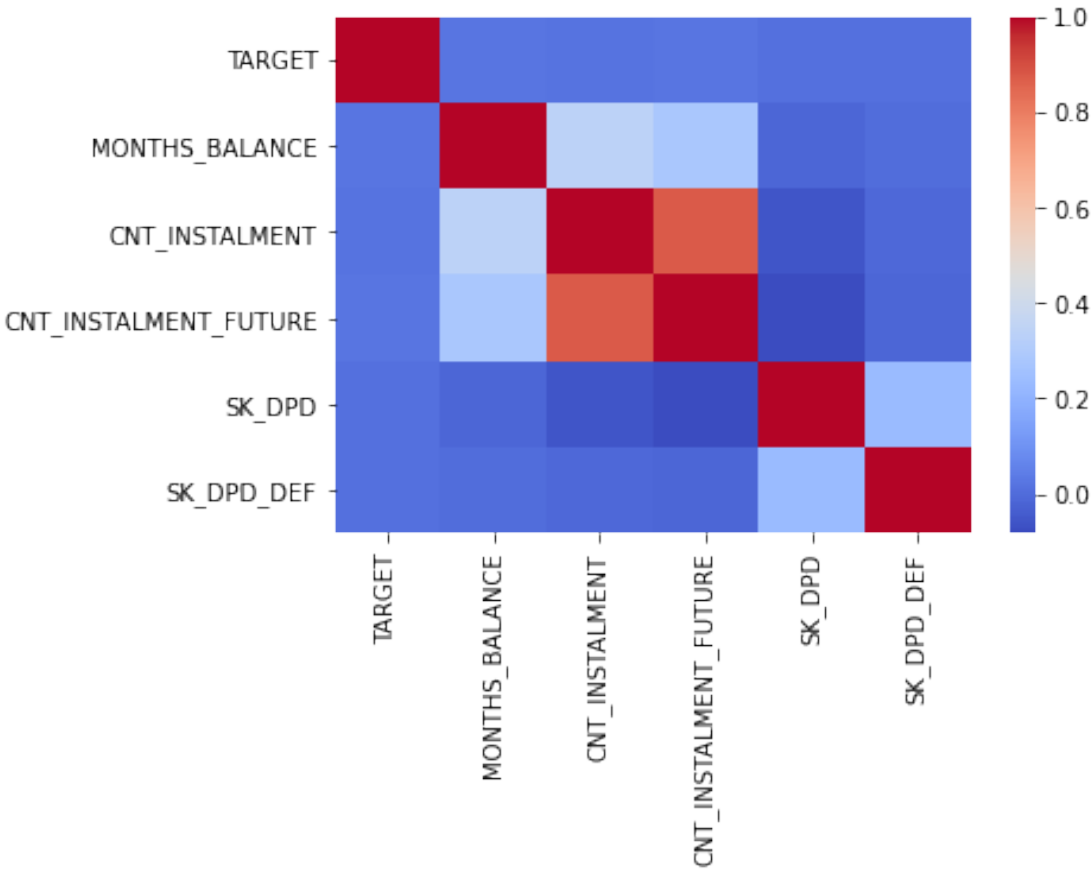
```
Total NA values are: 0
==========================================================================
==============================


For feature SK_DPD
Total Uniuqe values are: 3400
Total NA values are: 0
==========================================================================
==============================


For feature SK_DPD_DEF
Total Uniuqe values are: 2307
Total NA values are: 0
==========================================================================
==============================
```

```
#Merging with training data

pos_cash_bal_merged = df_train.iloc[:,:2].merge(POS_CASH_balance, on =
'SK_ID_CURR', how = 'left')
```

- **SK_ID_PREV** is 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) and **SK_ID_CURR** are customer is for ID of loan in our sample

```
# Lets see the correlation between features

sns.heatmap(pos_cash_bal_merged.drop(['SK_ID_CURR','SK_ID_PREV'],axis=
1).corr(),cmap='coolwarm')
```

```
<AxesSubplot:>
```



None has correlation with TAREGT

MONTHS_BALANCE

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)

```
pos_cash_bal_merged['MONTHS_BALANCE'].describe()
```

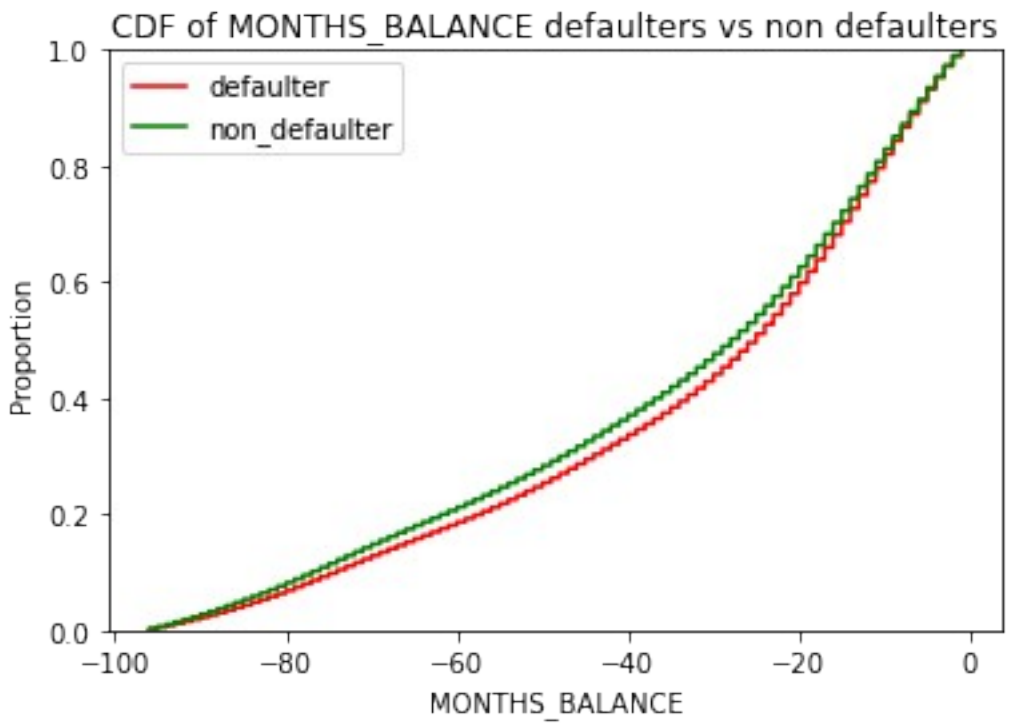
```
count    8.543375e+06
mean    -3.494812e+01
std      2.605372e+01
min     -9.600000e+01
25%     -5.400000e+01
50%     -2.800000e+01
75%     -1.300000e+01
max     -1.000000e+00
Name: MONTHS_BALANCE, dtype: float64
```

```
pos_cash_bal_merged['MONTHS_BALANCE'].isna().sum()
# NA values
```

18067

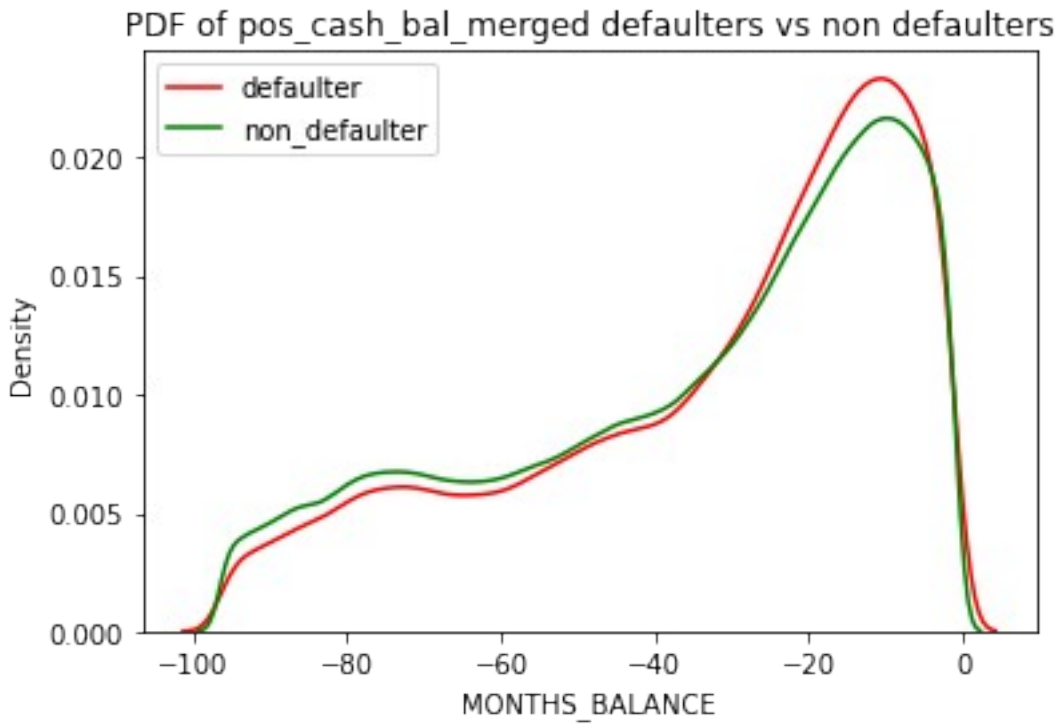
```
# 18067 NA VALUES

#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(pos_cash_bal_merged[pos_cash_bal_merged['TARGET']==1]
['MONTHS_BALANCE'],label='defaulter',color='red')
sns.ecdfplot(pos_cash_bal_merged[pos_cash_bal_merged['TARGET']==0]
['MONTHS_BALANCE'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of MONTHS_BALANCE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d3e039b9a0>
```



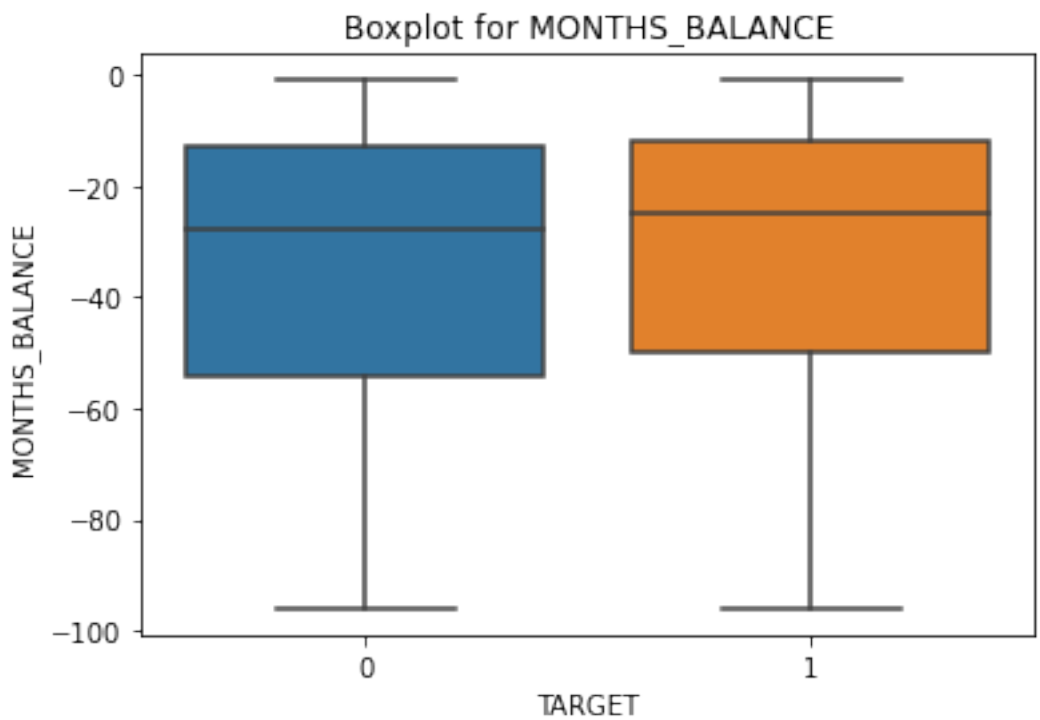

```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(pos_cash_bal_merged[pos_cash_bal_merged['TARGET']==1]
['MONTHS_BALANCE'],label='defaulter',color='red')
sns.kdeplot(pos_cash_bal_merged[pos_cash_bal_merged['TARGET']==0]
['MONTHS_BALANCE'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of MONTHS_BALANCE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d084b7cf40>
```




```
sns.boxplot(x='TARGET',y='MONTHS_BALANCE',data=pos_cash_bal_merged)
#plt.ylim([0.0,50000])
plt.title("Boxplot for MONTHS_BALANCE")
```

```
Text(0.5, 1.0, 'Boxplot for MONTHS_BALANCE')
```



```
#we dont see any pickable  difference here

###### referebce https://www.kaggle.com/tunguz/xgb-simple-features

#####With this source we get some approach as

### a.     Group /Aggregating columns and taking mean of more than one
related column
```

```
pos_cash_bal_merged.columns
```

```
Index(['SK_ID_CURR', 'TARGET', 'SK_ID_PREV', 'MONTHS_BALANCE',
       'CNT_INSTALMENT', 'CNT_INSTALMENT_FUTURE',
'NAME_CONTRACT_STATUS',
       'SK_DPD', 'SK_DPD_DEF'],
      dtype='object')
```

```
del pos_cash_bal_merged

#lets try aggregating
pos_cash_bal_merged_grouped=
pos_cash_bal_merged.groupby(['SK_ID_PREV']).mean()

pos_cash_bal_merged_grouped.head()
```

```
            SK_ID_CURR  TARGET  MONTHS_BALANCE  CNT_INSTALMENT  \
SK_ID_PREV
1000001.0       158271       0            -9.0        8.666667
1000003.0       252457       0            -2.5       12.000000
1000004.0       260094       0           -25.5        9.625000
1000005.0       176456       0           -51.0       10.000000
1000007.0       256657       0            -3.0        6.000000

            CNT_INSTALMENT_FUTURE  SK_DPD  SK_DPD_DEF
SK_ID_PREV
1000001.0                7.666667     0.0         0.0
1000003.0               10.500000     0.0         0.0
1000004.0                6.125000     0.0         0.0
1000005.0                5.000000     0.0         0.0
1000007.0                4.000000     0.0         0.0
```

```
pos_cash_bal_merged_grouped['MONTHS_BALANCE'].describe()
```

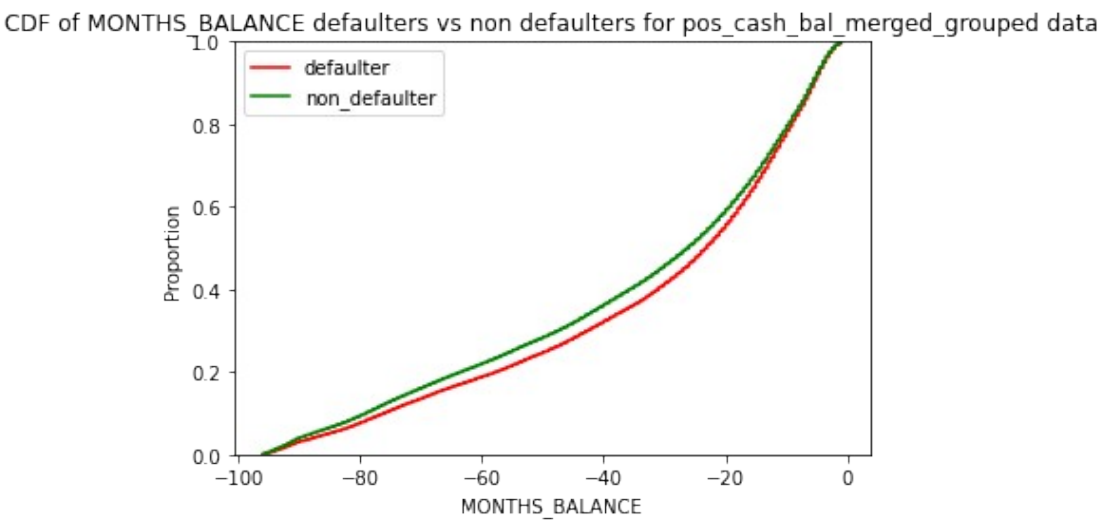
```
count    800337.000000
mean        -34.723802
std          27.237108
min         -96.000000
25%         -55.000000
50%         -26.000000
75%         -12.000000
max          -1.000000
Name: MONTHS_BALANCE, dtype: float64
```

```
pos_cash_bal_merged_grouped['MONTHS_BALANCE'].isna().sum()
# NA values
```

```
0
```

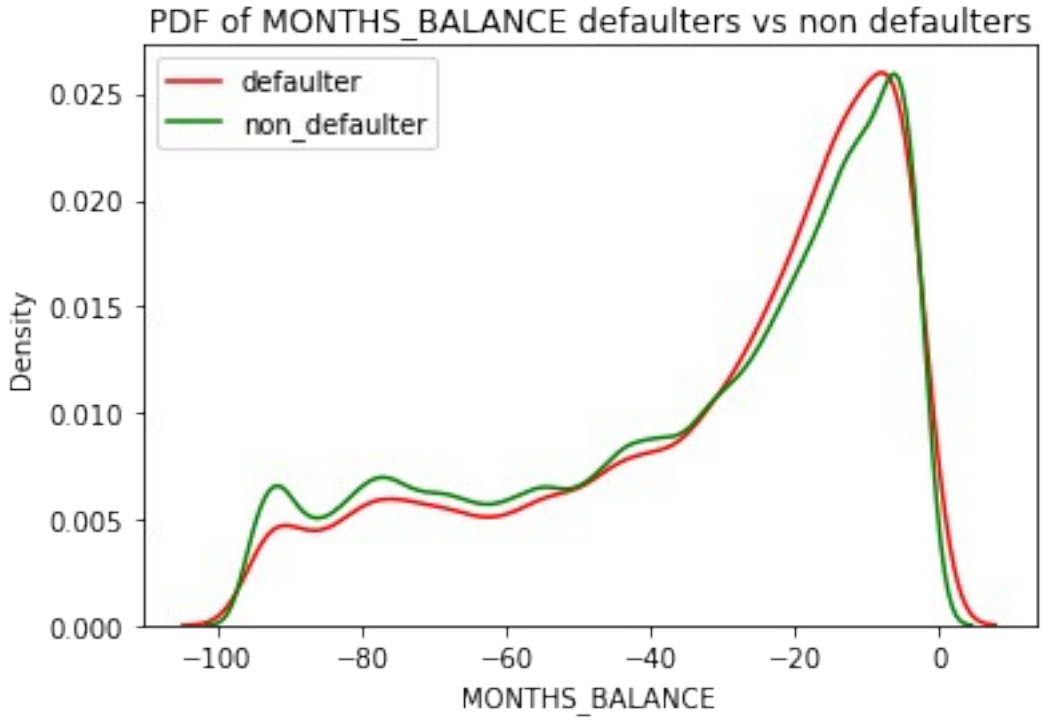
```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(pos_cash_bal_merged_grouped[pos_cash_bal_merged_grouped['
TARGET']==1]['MONTHS_BALANCE'],label='defaulter',color='red')
sns.ecdfplot(pos_cash_bal_merged_grouped[pos_cash_bal_merged_grouped['
TARGET']==0]['MONTHS_BALANCE'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of MONTHS_BALANCE defaulters vs non defaulters for
pos_cash_bal_merged_grouped data")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d3e0898e20>
```



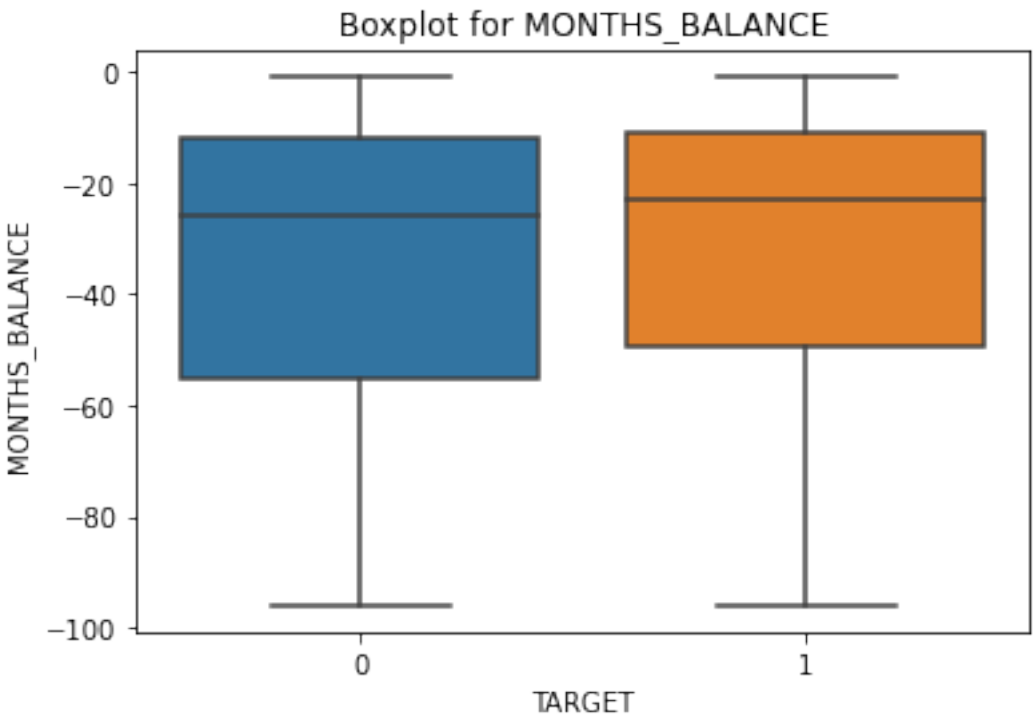

```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(pos_cash_bal_merged_grouped[pos_cash_bal_merged_grouped['T
ARGET']==1]['MONTHS_BALANCE'],label='defaulter',color='red')
sns.kdeplot(pos_cash_bal_merged_grouped[pos_cash_bal_merged_grouped['T
ARGET']==0]['MONTHS_BALANCE'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of MONTHS_BALANCE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1d3e0871e20>
```




```
sns.boxplot(x='TARGET',y='MONTHS_BALANCE',data=pos_cash_bal_merged_gro
uped)
#plt.ylim([0.0,50000])
plt.title("Boxplot for MONTHS_BALANCE")
```

```
Text(0.5, 1.0, 'Boxplot for MONTHS_BALANCE')
```




```
del pos_cash_bal_merged_grouped
```

well !! i dont see much diffrenece after aggregating with Previous id and by mean

credit_card_balance.csv

- This table contains Monthly balance of client's previous credit loans in Home Credit along with behavioral data

```
credit_card_balance.head()
```

```
   SK_ID_PREV  SK_ID_CURR  MONTHS_BALANCE  AMT_BALANCE  \
0     2562384      378907              -6       56.970
1     2582071      363914              -1    63975.555
2     1740877      371185              -7    31815.225
3     1389973      337855              -4   236572.110
4     1891521      126868              -1   453919.455

   AMT_CREDIT_LIMIT_ACTUAL  AMT_DRAWINGS_ATM_CURRENT
AMT_DRAWINGS_CURRENT  \
0                   135000                       0.0
877.5
1                    45000                    2250.0
2250.0
2                   450000                       0.0
0.0
```

```
3                   225000                    2250.0
2250.0
4                   450000                       0.0
11547.0

   AMT_DRAWINGS_OTHER_CURRENT  AMT_DRAWINGS_POS_CURRENT  \
0                         0.0                     877.5
1                         0.0                       0.0
2                         0.0                       0.0
3                         0.0                       0.0
4                         0.0                   11547.0

   AMT_INST_MIN_REGULARITY  ...  AMT_RECIVABLE
AMT_TOTAL_RECEIVABLE  \
0                 1700.325  ...          0.000                 0.000

1                 2250.000  ...      64875.555             64875.555

2                 2250.000  ...      31460.085             31460.085

3                11795.760  ...     233048.970            233048.970

4                22924.890  ...     453919.455            453919.455


   CNT_DRAWINGS_ATM_CURRENT  CNT_DRAWINGS_CURRENT
CNT_DRAWINGS_OTHER_CURRENT  \
0                       0.0                     1
0.0
1                       1.0                     1
0.0
2                       0.0                     0
0.0
3                       1.0                     1
0.0
4                       0.0                     1
0.0

   CNT_DRAWINGS_POS_CURRENT  CNT_INSTALMENT_MATURE_CUM
NAME_CONTRACT_STATUS  \
0                       1.0                       35.0
Active
1                       0.0                       69.0
Active
2                       0.0                       30.0
Active
3                       0.0                       10.0
Active
4                       1.0                      101.0
```

```
Active

   SK_DPD  SK_DPD_DEF
0       0           0
1       0           0
2       0           0
3       0           0
4       0           0

[5 rows x 23 columns]
```

```
credit_card_balance.shape
```

```
(3840312, 23)
```

- 3.8 M rows and 23 columns

```
credit_card_balance.duplicated().shape[0]
```

```
3840312
```

```
col_descriptions[col_descriptions['Table']=='credit_card_balance.csv']
```

```
                       Table                         Row  \
153  credit_card_balance.csv                 SK_ID_PREV
154  credit_card_balance.csv                  SK_ID_CURR
155  credit_card_balance.csv              MONTHS_BALANCE
156  credit_card_balance.csv                 AMT_BALANCE
157  credit_card_balance.csv     AMT_CREDIT_LIMIT_ACTUAL
158  credit_card_balance.csv    AMT_DRAWINGS_ATM_CURRENT
159  credit_card_balance.csv        AMT_DRAWINGS_CURRENT
160  credit_card_balance.csv  AMT_DRAWINGS_OTHER_CURRENT
161  credit_card_balance.csv    AMT_DRAWINGS_POS_CURRENT
162  credit_card_balance.csv     AMT_INST_MIN_REGULARITY
163  credit_card_balance.csv         AMT_PAYMENT_CURRENT
164  credit_card_balance.csv   AMT_PAYMENT_TOTAL_CURRENT
165  credit_card_balance.csv    AMT_RECEIVABLE_PRINCIPAL
166  credit_card_balance.csv               AMT_RECIVABLE
167  credit_card_balance.csv        AMT_TOTAL_RECEIVABLE
168  credit_card_balance.csv    CNT_DRAWINGS_ATM_CURRENT
169  credit_card_balance.csv        CNT_DRAWINGS_CURRENT
170  credit_card_balance.csv  CNT_DRAWINGS_OTHER_CURRENT
171  credit_card_balance.csv    CNT_DRAWINGS_POS_CURRENT
172  credit_card_balance.csv   CNT_INSTALMENT_MATURE_CUM
173  credit_card_balance.csv        NAME_CONTRACT_STATUS
174  credit_card_balance.csv                      SK_DPD
175  credit_card_balance.csv                  SK_DPD_DEF

                                          Description  \
153  ID of previous credit in Home credit related t...
154                           ID of loan in our sample
155  Month of balance relative to application date ...
```

156	Balance during the month of previous credit	
157	Credit card limit during the month of the prev...	
158	Amount drawing at ATM during the month of the ...	
159	Amount drawing during the month of the previou...	
160	Amount of other drawings during the month of t...	
161	Amount drawing or buying goods during the mont...	
162	Minimal installment for this month of the prev...	
163	How much did the client pay during the month o...	
164	How much did the client pay during the month i...	
165	Amount receivable for principal on the previou...	
166	Amount receivable on the previous credit	
167	Total amount receivable on the previous credit	
168	Number of drawings at ATM during this month on...	
169	Number of drawings during this month on the pr...	
170	Number of other drawings during this month on ...	
171	Number of drawings for goods during this month...	
172	Number of paid installments on the previous cr...	
173	Contract status (active signed,...) on the pre...	
174	DPD (Days past due) during the month on the pr...	
175	DPD (Days past due) during the month with tole...	

		Special
153		hashed
154		hashed
155	time only relative to the application	
156		NaN
157		NaN
158		NaN
159		NaN
160		NaN
161		NaN
162		NaN
163		NaN
164		NaN
165		NaN
166		NaN
167		NaN
168		NaN
169		NaN
170		NaN
171		NaN
172		NaN
173		NaN
174		NaN
175		NaN

SK_ID_PREV

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)

```
credit_card_balance['SK_ID_PREV'].nunique()
```

```
104307
```

```
#So many entries for single SK_ID_PREV
```

SK_ID_CURR

ID of loan in our sample

```
credit_card_balance['SK_ID_CURR'].nunique()
```

```
103558
```

```
for i in credit_card_balance.columns:
    print(f"For feature {i}")
    print(f"Total Uniuqe values are:
{credit_card_balance[i].nunique()}")
    print(f"Total NA values are:
{credit_card_balance[i].isna().sum()}")
    print("="*100)
    print("\n \n")
```

```
For feature SK_ID_PREV
Total Uniuqe values are: 104307
Total NA values are: 0
==========================================================================
==============================



For feature SK_ID_CURR
Total Uniuqe values are: 103558
Total NA values are: 0
==========================================================================
==============================



For feature MONTHS_BALANCE
Total Uniuqe values are: 96
Total NA values are: 0
==========================================================================
==============================
```

```
For feature AMT_BALANCE
Total Uniuqe values are: 1347904
Total NA values are: 0
==========================================================================
==============================


For feature AMT_CREDIT_LIMIT_ACTUAL
Total Uniuqe values are: 181
Total NA values are: 0
==========================================================================
==============================


For feature AMT_DRAWINGS_ATM_CURRENT
Total Uniuqe values are: 2267
Total NA values are: 749816
==========================================================================
==============================


For feature AMT_DRAWINGS_CURRENT
Total Uniuqe values are: 187005
Total NA values are: 0
==========================================================================
==============================


For feature AMT_DRAWINGS_OTHER_CURRENT
Total Uniuqe values are: 1832
Total NA values are: 749816
==========================================================================
==============================


For feature AMT_DRAWINGS_POS_CURRENT
Total Uniuqe values are: 168748
Total NA values are: 749816
==========================================================================
==============================
```

```
For feature AMT_INST_MIN_REGULARITY
Total Uniuqe values are: 312266
Total NA values are: 305236
==========================================================================
==============================


For feature AMT_PAYMENT_CURRENT
Total Uniuqe values are: 163209
Total NA values are: 767988
==========================================================================
==============================


For feature AMT_PAYMENT_TOTAL_CURRENT
Total Uniuqe values are: 182957
Total NA values are: 0
==========================================================================
==============================


For feature AMT_RECEIVABLE_PRINCIPAL
Total Uniuqe values are: 1195839
Total NA values are: 0
==========================================================================
==============================


For feature AMT_RECIVABLE
Total Uniuqe values are: 1338878
Total NA values are: 0
==========================================================================
==============================


For feature AMT_TOTAL_RECEIVABLE
Total Uniuqe values are: 1339008
Total NA values are: 0
==========================================================================
==============================
```

```
For feature CNT_DRAWINGS_ATM_CURRENT
Total Uniuqe values are: 44
Total NA values are: 749816
==========================================================================
==============================


For feature CNT_DRAWINGS_CURRENT
Total Uniuqe values are: 129
Total NA values are: 0
==========================================================================
==============================


For feature CNT_DRAWINGS_OTHER_CURRENT
Total Uniuqe values are: 11
Total NA values are: 749816
==========================================================================
==============================


For feature CNT_DRAWINGS_POS_CURRENT
Total Uniuqe values are: 133
Total NA values are: 749816
==========================================================================
==============================


For feature CNT_INSTALMENT_MATURE_CUM
Total Uniuqe values are: 121
Total NA values are: 305236
==========================================================================
==============================


For feature NAME_CONTRACT_STATUS
Total Uniuqe values are: 7
Total NA values are: 0
==========================================================================
==============================


For feature SK_DPD
```

```
Total Uniuqe values are: 917
Total NA values are: 0
==========================================================================
==============================


For feature SK_DPD_DEF
Total Uniuqe values are: 378
Total NA values are: 0
==========================================================================
==============================
```

```
#let' see which are categorical columns

credit_card_balance.select_dtypes(include= [object]).columns.to_list()
```

```
['NAME_CONTRACT_STATUS']
```

```
#So we have just one categorical columns

#before analzing that lets merge the train data for TARGET analysis

credit_card_balance_merged =
df_train.iloc[:,:2].merge(credit_card_balance, on = 'SK_ID_CURR', how
= 'left')

credit_card_balance_merged.corr()
['TARGET'].sort_values(ascending=False)[:10]
```

```
TARGET                       1.000000
AMT_BALANCE                  0.050098
AMT_TOTAL_RECEIVABLE         0.049839
AMT_RECIVABLE                0.049803
AMT_RECEIVABLE_PRINCIPAL     0.049692
AMT_INST_MIN_REGULARITY      0.039798
CNT_DRAWINGS_ATM_CURRENT     0.038437
CNT_DRAWINGS_CURRENT         0.037793
MONTHS_BALANCE               0.035695
CNT_DRAWINGS_POS_CURRENT     0.029536
Name: TARGET, dtype: float64
```

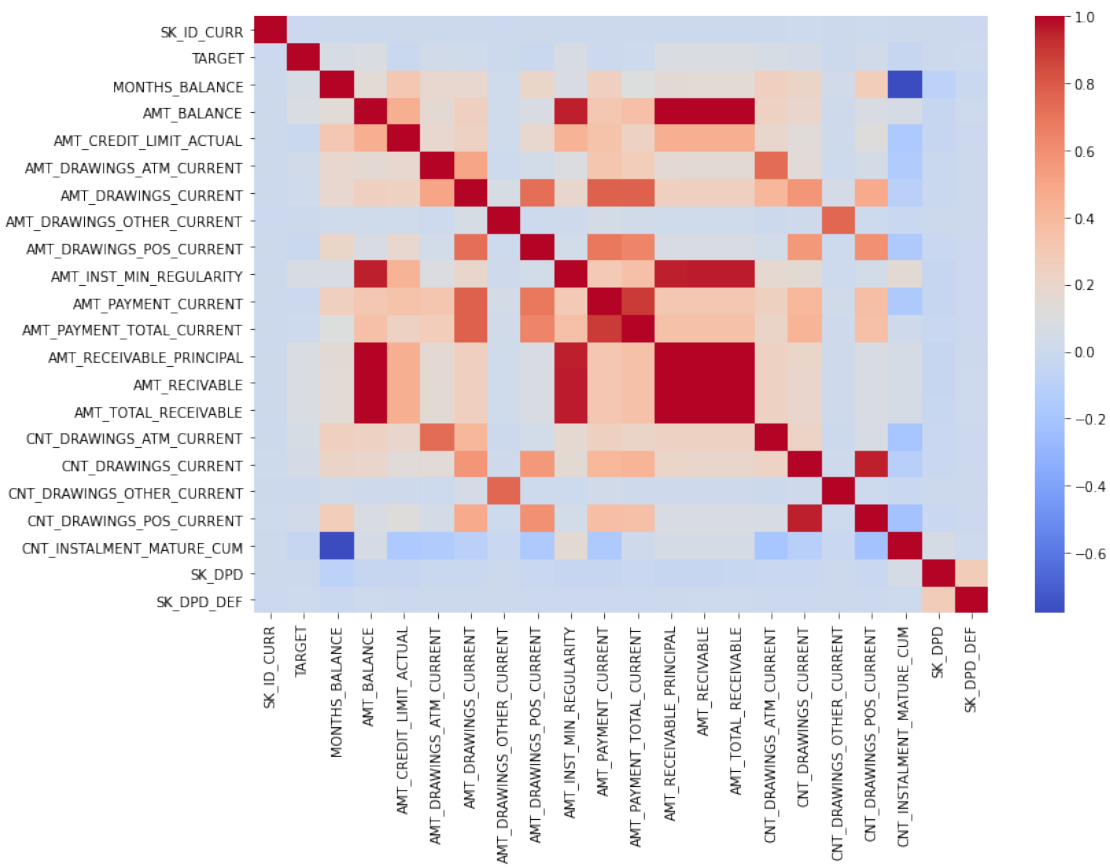
```
#Lets try to group by ID and get the median value ,However we didnt
get any very insighful results last time but it will fast the process
credit_card_balance_merged_grp=credit_card_balance_merged.groupby('SK_
ID_PREV').median()
```

```
credit_card_balance_merged_grp.corr()
['TARGET'].sort_values(ascending=False)[:10]
```

```
TARGET                       1.000000
AMT_BALANCE                  0.085441
AMT_TOTAL_RECEIVABLE         0.084827
AMT_RECIVABLE                0.084819
AMT_RECEIVABLE_PRINCIPAL     0.084624
AMT_INST_MIN_REGULARITY      0.073133
CNT_DRAWINGS_ATM_CURRENT     0.066886
MONTHS_BALANCE               0.061739
CNT_DRAWINGS_CURRENT         0.051064
CNT_DRAWINGS_POS_CURRENT     0.029338
Name: TARGET, dtype: float64
```

```
plt.figure(figsize=(12,8))
sns.heatmap(credit_card_balance_merged_grp.corr(),cmap='coolwarm')
```

```
<AxesSubplot:>
```



We can see

- No high correlated feature here for Target

- AMT_Balance ,AMT_TOTAL_RECEIVABLE, AMT_RECIVABLE,AMT_RECEIVABLE_PRINCIPAL , AMT_INST_MIN_REGULARITY are higly correlated to each other

NAME_CONTRACT_STATUS

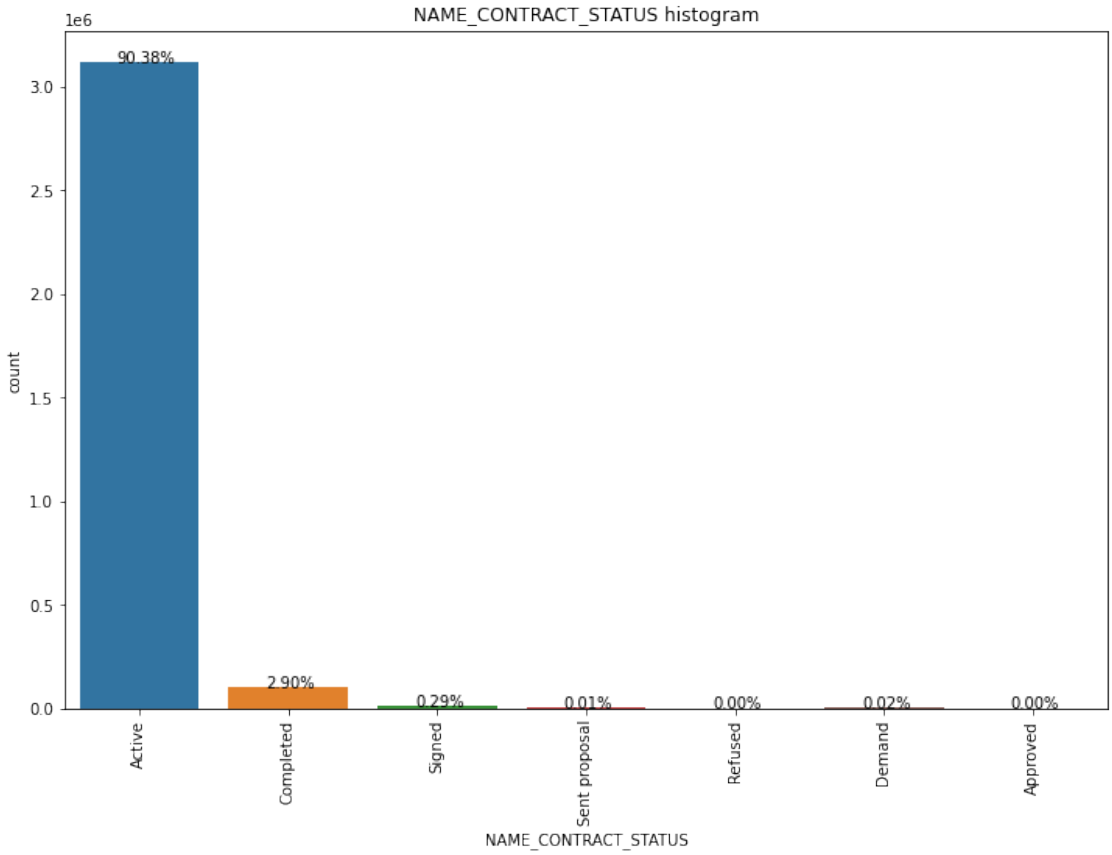
Month of balance relative to application date (-1 means the freshest balance date)

```
credit_card_balance_merged['NAME_CONTRACT_STATUS'].value_counts()
```

```
Active           3116673
Completed         100031
Signed              9939
Demand               843
Sent proposal        460
Refused               14
Approved               5
Name: NAME_CONTRACT_STATUS, dtype: int64
```

```
#lets see its distribution
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_CONTRACT_STATUS',data=credit_card_balance_mer
ged)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}
%'.format(p.get_height()*100/len(credit_card_balance_merged)),
(p.get_x()+0.25, p.get_height()+0.01))
ax.set_title("NAME_CONTRACT_STATUS histogram")
```

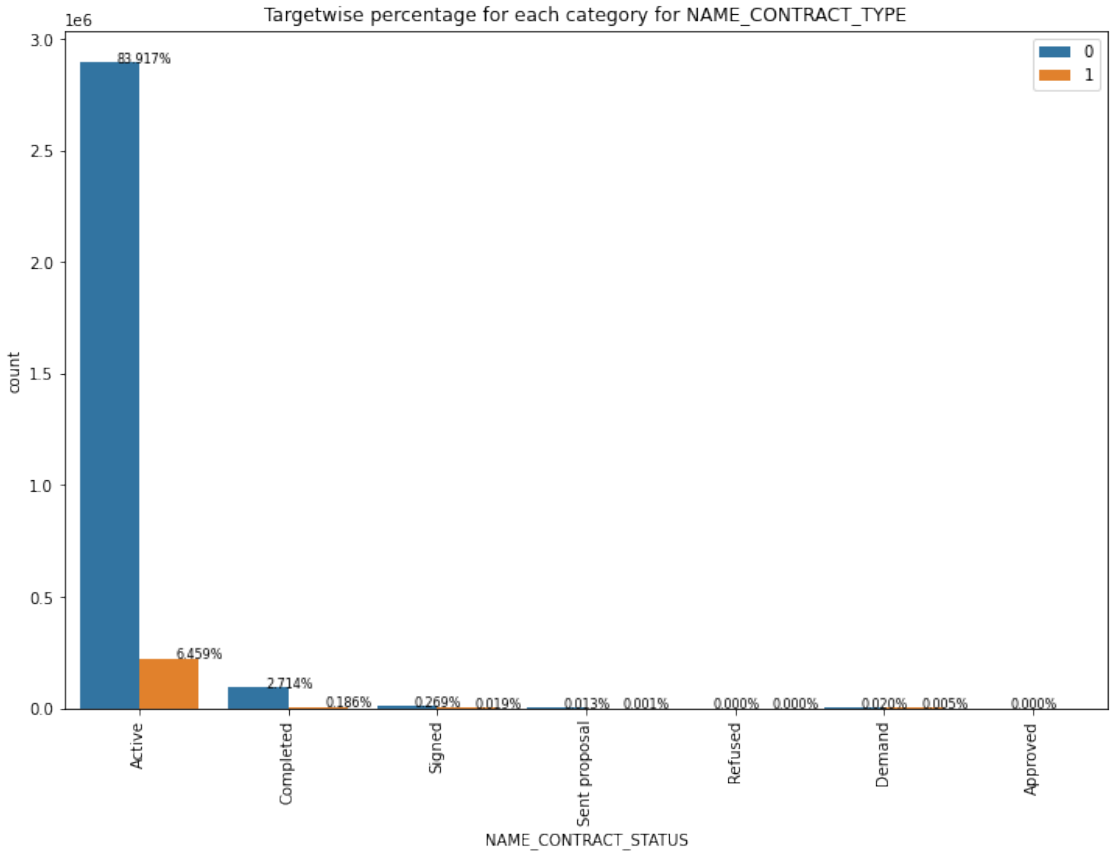
```
Text(0.5, 1.0, 'NAME_CONTRACT_STATUS histogram')
```



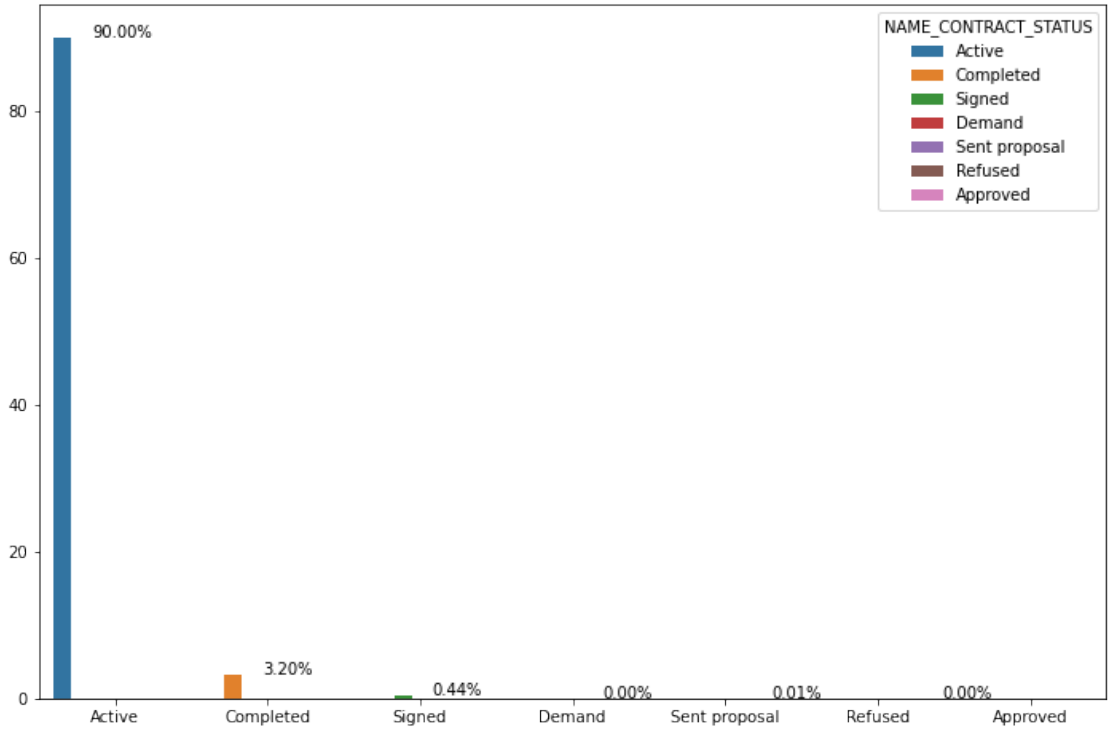
- More than 90% are Active cards only

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NAME_CONTRACT_STATUS',hue='TARGET',data=credit_car
d_balance_merged)
plt.xticks(rotation=90)
plt.legend(loc='upper right')
for p in ax.patches:
    ax.annotate('{:.3f}
%'.format(p.get_height()*100/len(credit_card_balance_merged)),
(p.get_x()+0.25, p.get_height()+0.01),size=8)
ax.set_title("Targetwise percentage for each category for
NAME_CONTRACT_TYPE")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NAME_CONTRACT_TYPE')
```




```
#lets see out of all deafulters what are the percentage of defaulters
in each category

plot_for_categorical_defaulters(credit_card_balance_merged,'NAME_CONTR
ACT_STATUS')
```



- As most client are Active so defaukters are from active only
- As we have seen in correlational plot AMT_Balance ,AMT_TOTAL_RECEIVABLE, AMT_RECIVABLE,AMT_RECEIVABLE_PRINCIPAL , AMT_INST_MIN_REGULARITY are higly correlated to each other.We can study one and all will be similar

CNT_INSTALMENT_MATURE_CUM

number of installments paid on the previous credits.

```
credit_card_balance_merged['CNT_INSTALMENT_MATURE_CUM'].describe()
```

```
count    3.227965e+06
mean     5.907332e+04
std      1.066137e+05
min     -4.202502e+05
25%      0.000000e+00
50%      0.000000e+00
75%      9.040554e+04
max      1.354829e+06
Name: AMT_BALANCE, dtype: float64
```

```
credit_card_balance_merged['CNT_INSTALMENT_MATURE_CUM'].isna().sum()
```

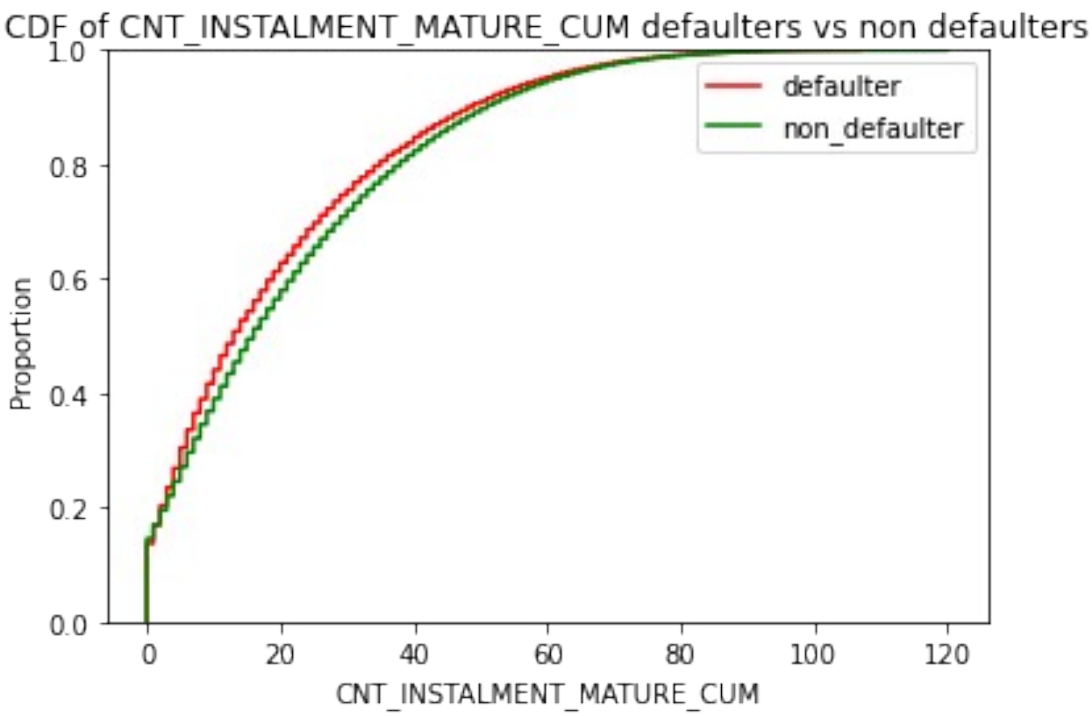
```
484990
```

```
credit_card_balance_merged['CNT_INSTALMENT_MATURE_CUM'].notnull().sum(
)
```

```
2963581
```

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(credit_card_balance_merged[credit_card_balance_merged['TA
RGET']==1]['CNT_INSTALMENT_MATURE_CUM'],label='defaulter',color='red')
sns.ecdfplot(credit_card_balance_merged[credit_card_balance_merged['TA
RGET']==0]
['CNT_INSTALMENT_MATURE_CUM'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of CNT_INSTALMENT_MATURE_CUM defaulters vs non
defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1dba8da7430>
```

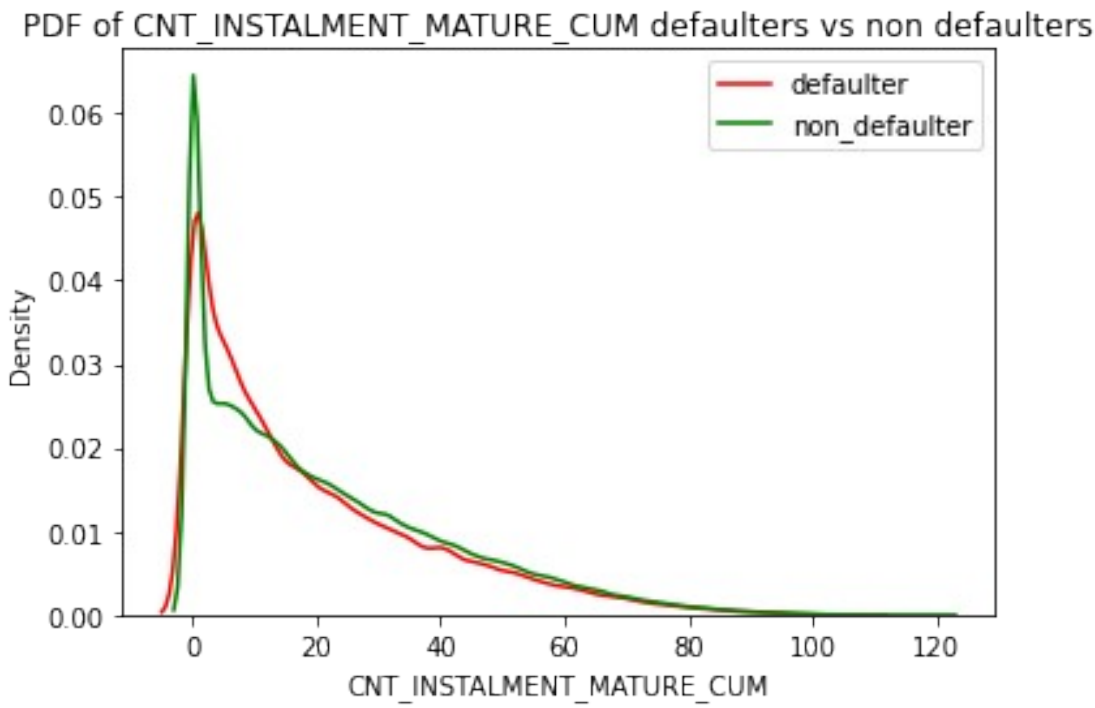


- Number of installment paid is seems to be lower than non -defaulters tha deafaulters

```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(credit_card_balance_merged[credit_card_balance_merged['TAR
GET']==1]['CNT_INSTALMENT_MATURE_CUM'],label='defaulter',color='red')
sns.kdeplot(credit_card_balance_merged[credit_card_balance_merged['TAR
GET']==0]
```

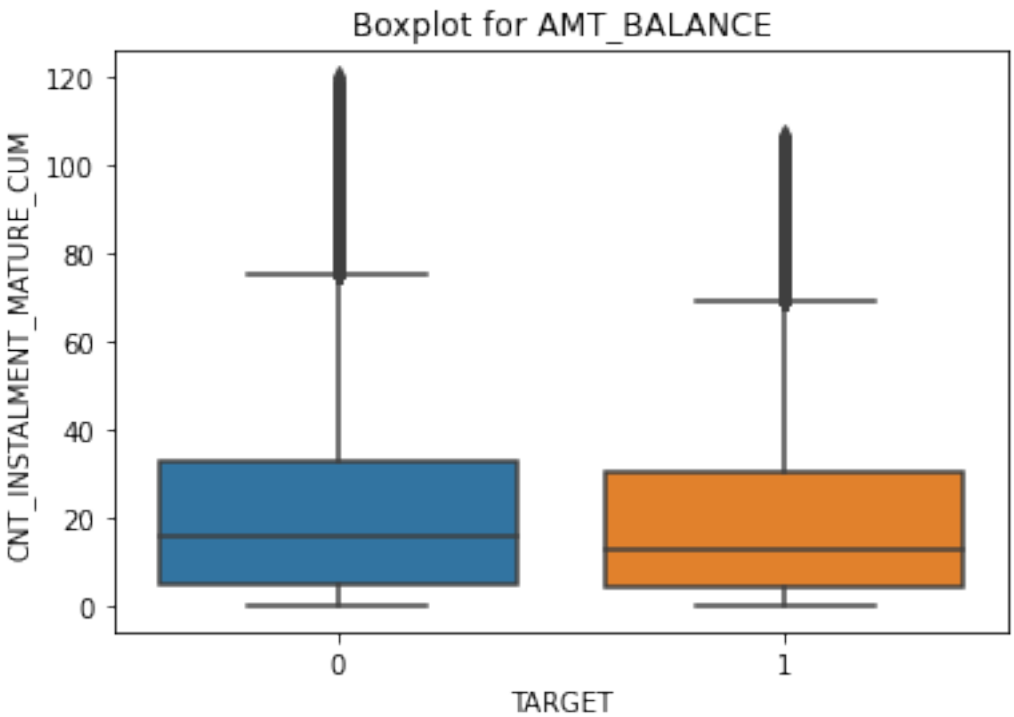
```
['CNT_INSTALMENT_MATURE_CUM'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of CNT_INSTALMENT_MATURE_CUM defaulters vs non
defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1dbb4deae50>
```




```
sns.boxplot(x='TARGET',y='CNT_INSTALMENT_MATURE_CUM',data=credit_card_
balance_merged)
#plt.ylim([0.0,50000])
plt.title("Boxplot for AMT_BALANCE")
```

```
Text(0.5, 1.0, 'Boxplot for AMT_BALANCE')
```



- This show as the normal number of installments goes high ,There are normally less payments done by the deafulters,Its obvious more defaulters show irregularity in payment for later EMI's only.

AMT_TOTAL_RECEIVABLE

Total amount receivable on the previous credit

```
credit_card_balance_merged['AMT_TOTAL_RECEIVABLE'].describe()
```

```
count    3.227965e+06
mean     5.887259e+04
std      1.062813e+05
min     -4.202502e+05
25%      0.000000e+00
50%      0.000000e+00
75%      9.022635e+04
max      1.354829e+06
Name: AMT_TOTAL_RECEIVABLE, dtype: float64
```

```
credit_card_balance_merged['AMT_TOTAL_RECEIVABLE'].isna().sum()
```

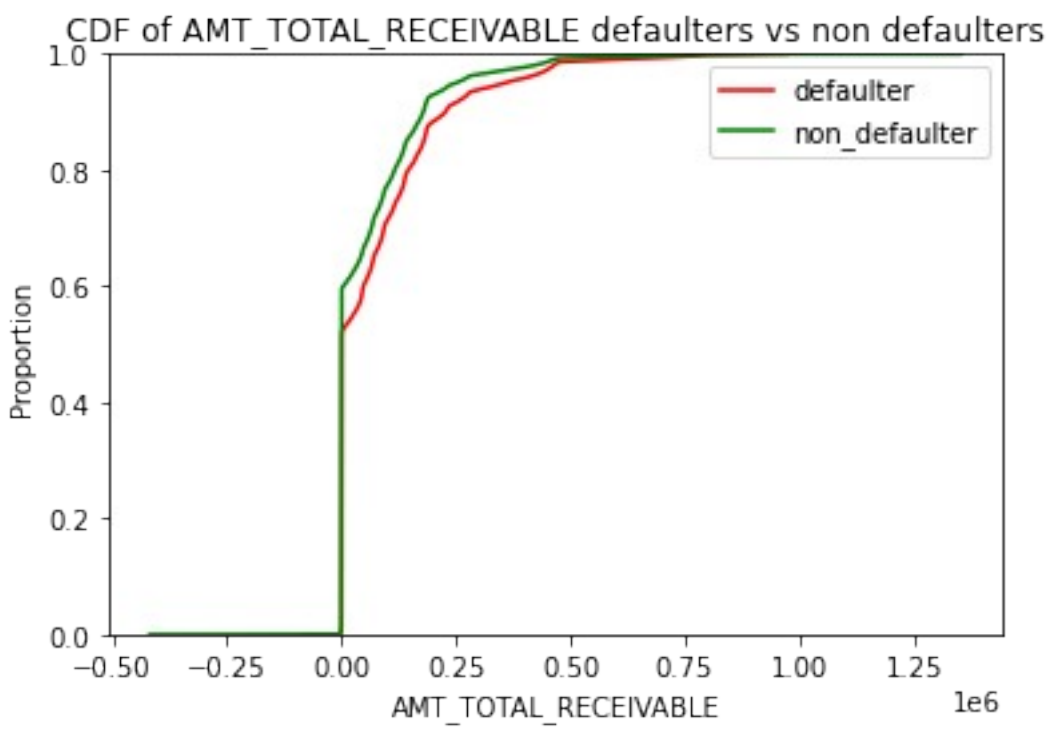
```
220606
```

```
#very high NA values ,because for non defaulters it should it not
available only

credit_card_balance_merged['AMT_TOTAL_RECEIVABLE'].notnull().sum()
```

```
3227965
```

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(credit_card_balance_merged[credit_card_balance_merged['TA
RGET']==1]['AMT_TOTAL_RECEIVABLE'],label='defaulter',color='red')
sns.ecdfplot(credit_card_balance_merged[credit_card_balance_merged['TA
RGET']==0]
['AMT_TOTAL_RECEIVABLE'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of AMT_TOTAL_RECEIVABLE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1dba977c7f0>
```

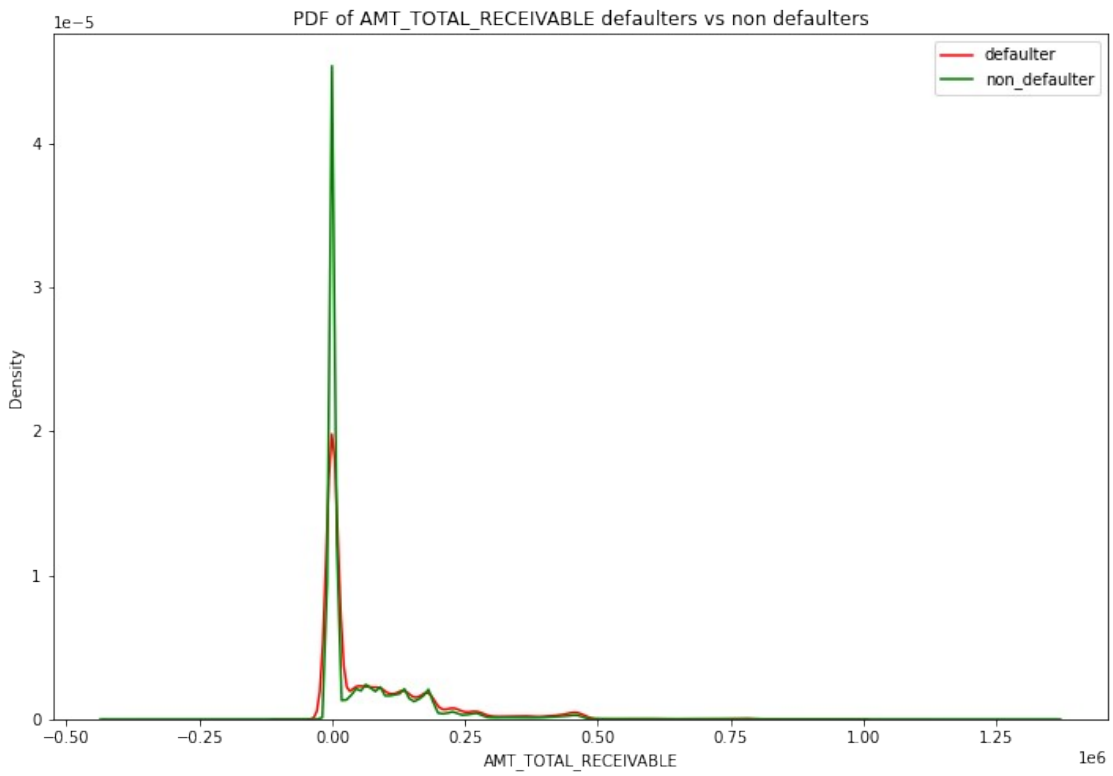


- AMT_TOTAL_RECEIVABLE is seems to be higher than non -defaulters for deafaulters;

```
#Lets see its PDF plot for both defaulters and non defaulters

plt.figure(figsize=(12,8))
sns.kdeplot(credit_card_balance_merged[credit_card_balance_merged['TAR
```

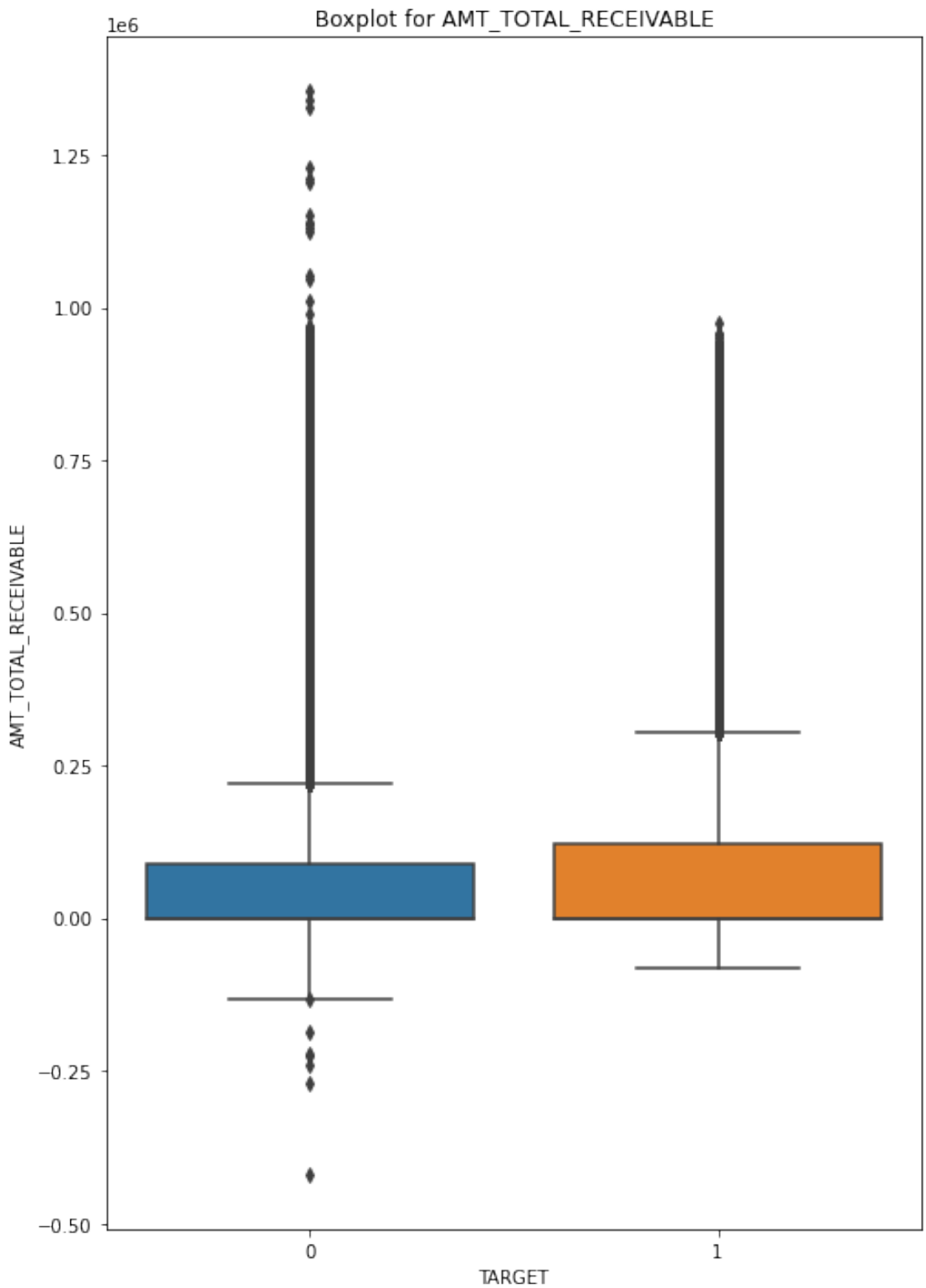
```
GET']==1]['AMT_TOTAL_RECEIVABLE'],label='defaulter',color='red')
sns.kdeplot(credit_card_balance_merged[credit_card_balance_merged['TAR
GET']==0]['AMT_TOTAL_RECEIVABLE'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of AMT_TOTAL_RECEIVABLE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1dbaa13ff10>
```




```
plt.figure(figsize=(8,12))
sns.boxplot(x='TARGET',y='AMT_TOTAL_RECEIVABLE',data=credit_card_balan
ce_merged)
#plt.ylim([0.0,50000])
plt.title("Boxplot for AMT_TOTAL_RECEIVABLE")
```

```
Text(0.5, 1.0, 'Boxplot for AMT_TOTAL_RECEIVABLE')
```



- This show some differnece between both categories ,after 1st qunatile downwards ,there are some outliers for non defaulters This could be negative if the client has paid extra amounts

- There are also some defaulters with negative amount recievable but very less than non defaulters

SK_DPD_DEF

DPD (Days past due) during the month with tolerance (debts with low loan amounts are ignored) of the previous credit

```
credit_card_balance_merged['SK_DPD_DEF'].describe()
```

```
count    3.227965e+06
mean     2.857060e-01
std      1.957034e+01
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      0.000000e+00
max      2.800000e+03
Name: SK_DPD_DEF, dtype: float64
```

```
credit_card_balance_merged['SK_DPD_DEF'].isna().sum()
```

```
220606
```

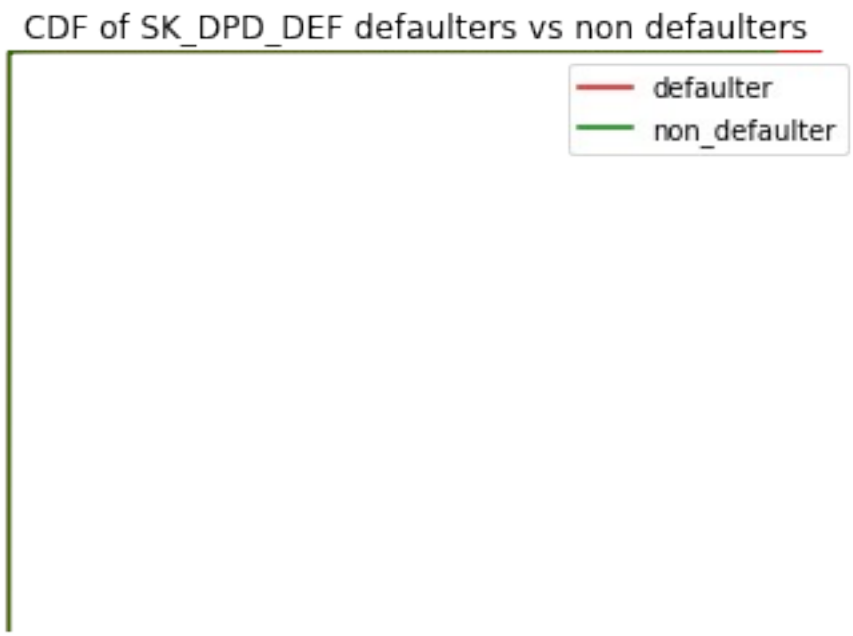
```
#very high NA values ,because for non defaulters it should it not
available only
```

```
credit_card_balance_merged['SK_DPD_DEF'].notnull().sum()
```

```
3227965
```

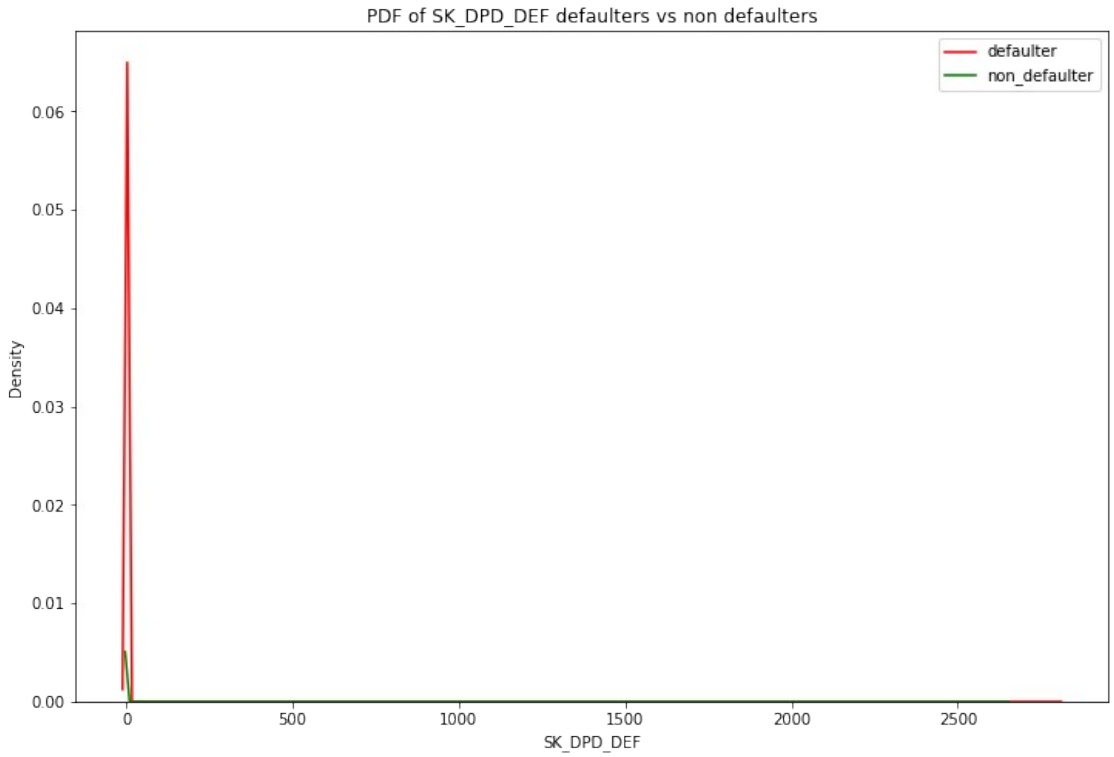
```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(credit_card_balance_merged[credit_card_balance_merged['TA
RGET']==1]['SK_DPD_DEF'],label='defaulter',color='red')
sns.ecdfplot(credit_card_balance_merged[credit_card_balance_merged['TA
RGET']==0]['SK_DPD_DEF'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of SK_DPD_DEF defaulters vs non defaulters")
plt.legend()
plt.axis('off')
```

```
(-140.0, 2940.0, 0.0, 1.0)
```




```
#Lets see its PDF plot for both defaulters and non defaulters

plt.figure(figsize=(12,8))
sns.kdeplot(credit_card_balance_merged[credit_card_balance_merged['TAR
GET']==1]['SK_DPD_DEF'],label='defaulter',color='red')
sns.kdeplot(credit_card_balance_merged[credit_card_balance_merged['TAR
GET']==0]['SK_DPD_DEF'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of SK_DPD_DEF defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1dbacd9aaf0>
```

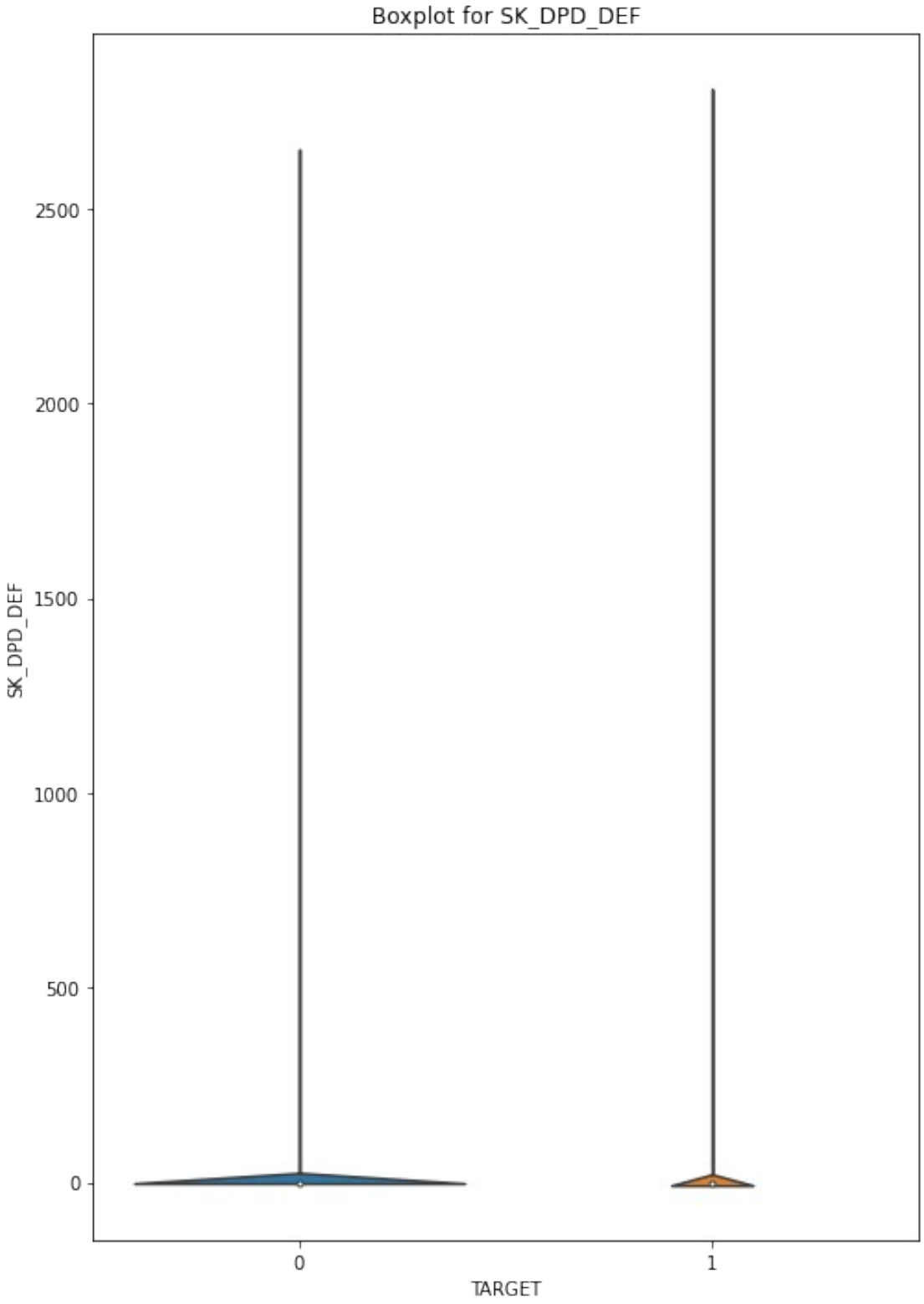



```
np.quantile(credit_card_balance_merged['SK_DPD_DEF'].dropna(),0.97)
```

```
0.0
```

```
plt.figure(figsize=(8,12))
sns.violinplot(x='TARGET',y='SK_DPD_DEF',data=credit_card_balance_merg
ed)
#plt.ylim([0,100])
plt.title("Boxplot for SK_DPD_DEF")
```

```
Text(0.5, 1.0, 'Boxplot for SK_DPD_DEF')
```



- This show some most DPD (97% after dropping NA) is zero and both categories lies here
- At zero also there are so many defaulters which have DPD =0

AMT_BALANCE

Balance during the month of previous credit

```
credit_card_balance_merged['AMT_BALANCE'].describe()
```

```
count    3.227965e+06
mean     5.907332e+04
std      1.066137e+05
min     -4.202502e+05
25%      0.000000e+00
50%      0.000000e+00
75%      9.040554e+04
max      1.354829e+06
Name: AMT_BALANCE, dtype: float64
```

```
credit_card_balance_merged['AMT_BALANCE'].isna().sum()
```

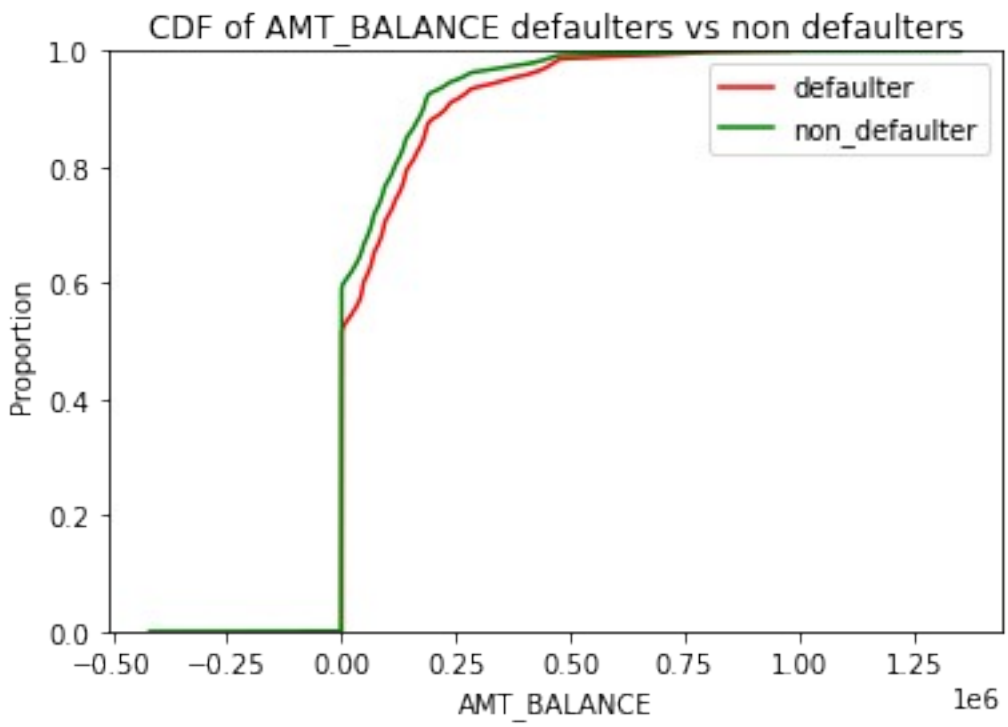
```
220606
```

```
credit_card_balance_merged['AMT_BALANCE'].notnull().sum()
```

```
3227965
```

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(credit_card_balance_merged[credit_card_balance_merged['TA
RGET']==1]['AMT_BALANCE'],label='defaulter',color='red')
sns.ecdfplot(credit_card_balance_merged[credit_card_balance_merged['TA
RGET']==0]['AMT_BALANCE'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of AMT_BALANCE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

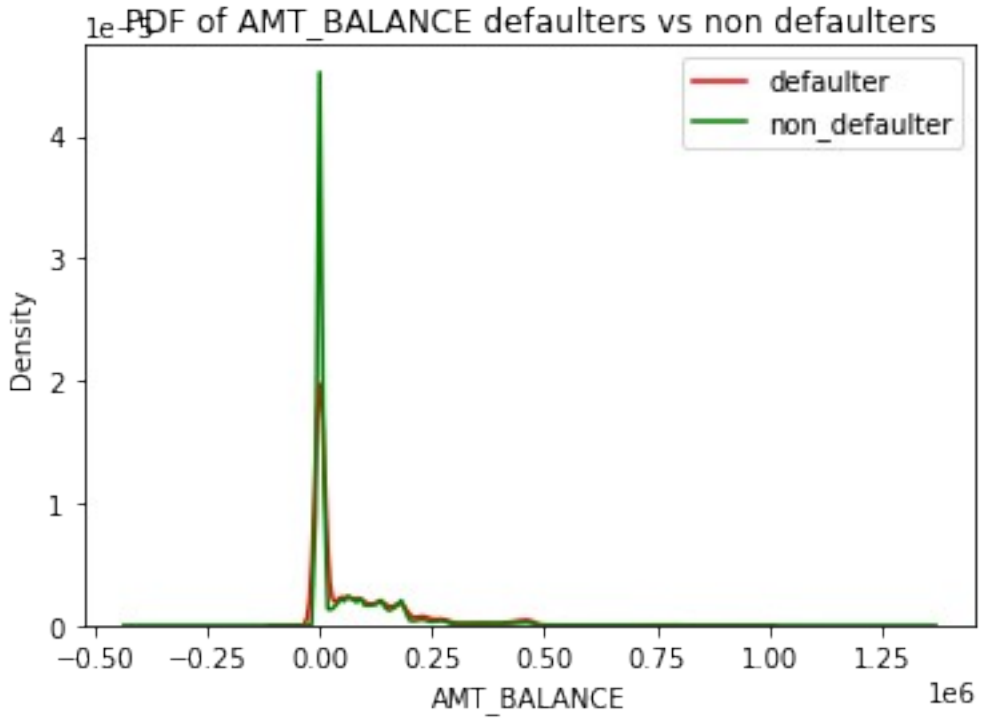
```
<matplotlib.legend.Legend at 0x1db90d51eb0>
```



- here we can see after AMT_balabce more than 0 some deaulters are showing higher AMT_BALANCE more than the non defaulters

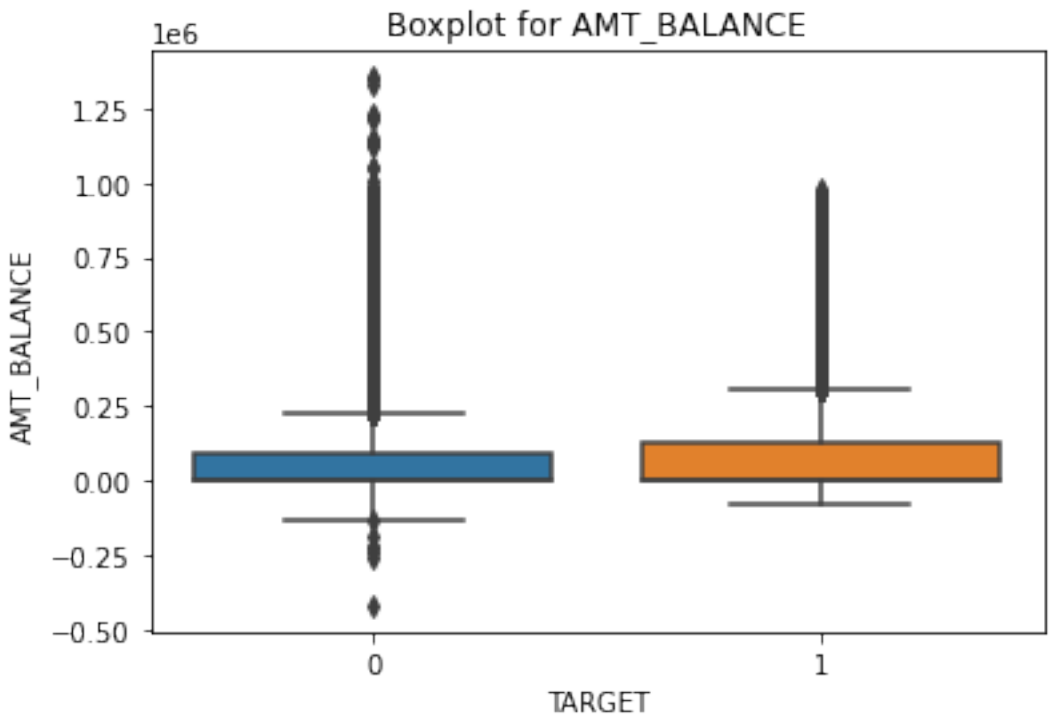
```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(credit_card_balance_merged[credit_card_balance_merged['TAR
GET']==1]['AMT_BALANCE'],label='defaulter',color='red')
sns.kdeplot(credit_card_balance_merged[credit_card_balance_merged['TAR
GET']==0]['AMT_BALANCE'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of AMT_BALANCE defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1dba4bfa7f0>
```




```
sns.boxplot(x='TARGET',y='AMT_BALANCE',data=credit_card_balance_merged
)
#plt.ylim([0.0,50000])
plt.title("Boxplot for AMT_BALANCE")
```

```
Text(0.5, 1.0, 'Boxplot for AMT_BALANCE')
```



- AMT_BALANCE normal values mostly more at 70 pecentile is slightly higher for defaulters
- rest features dont show even considerable amount of liner relationship with TARGET,So we didn;t studtied it

installments_payments.csv

This table consist past payment data for each installment of previous credits in Home credit realted to loans in our sample

```
installments_payments.head()
```

```
   SK_ID_PREV  SK_ID_CURR  NUM_INSTALMENT_VERSION
NUM_INSTALMENT_NUMBER  \
0     1054186      161674                     1.0
6
1     1330831      151639                     0.0
34
2     2085231      193053                     2.0
1
3     2452527      199697                     1.0
3
4     2714724      167756                     1.0
2

   DAYS_INSTALMENT  DAYS_ENTRY_PAYMENT  AMT_INSTALMENT  AMT_PAYMENT
```

```
0          -1180.0             -1187.0        6948.360     6948.360
1          -2156.0             -2156.0        1716.525     1716.525
2            -63.0               -63.0       25425.000    25425.000
3          -2418.0             -2426.0       24350.130    24350.130
4          -1383.0             -1366.0        2165.040     2160.585
```

```
installments_payments.shape
```

```
(13605401, 8)
```

we have 13.6 M rows and 8 columns

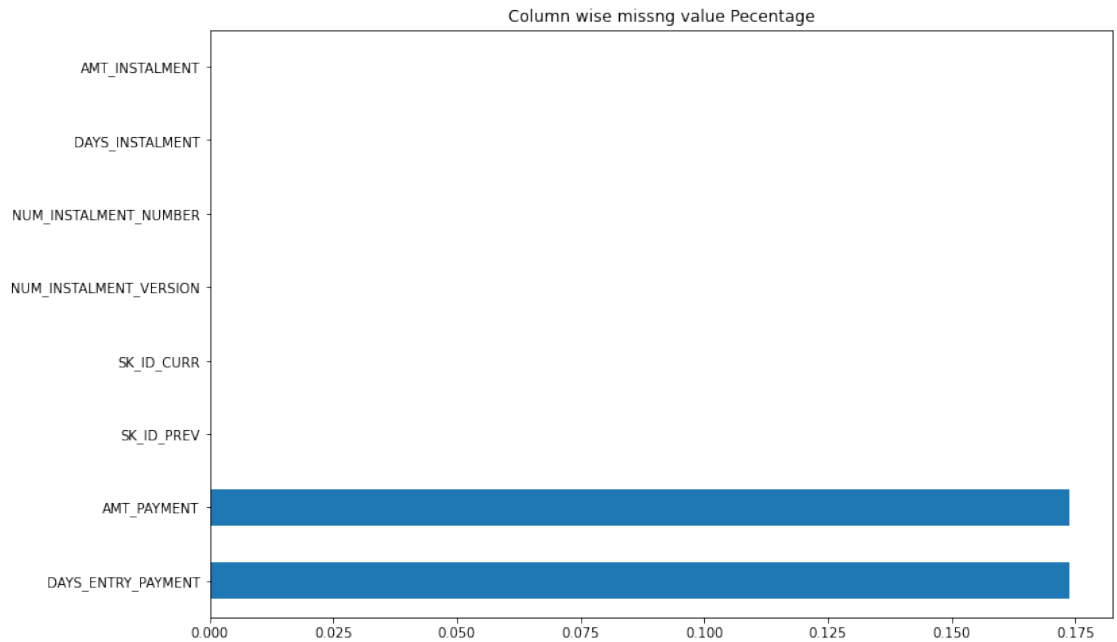
```
percent_missing = installments_payments.isnull().sum() * 100 /
len(previous_application)
```

```
percent_missing.sort_values(ascending=False)
```

```
DAYS_ENTRY_PAYMENT        0.17393
AMT_PAYMENT               0.17393
SK_ID_PREV                0.00000
SK_ID_CURR                0.00000
NUM_INSTALMENT_VERSION    0.00000
NUM_INSTALMENT_NUMBER     0.00000
DAYS_INSTALMENT           0.00000
AMT_INSTALMENT            0.00000
dtype: float64
```

```
plt.figure(figsize=(12,8))
percent_missing.sort_values(ascending=False).plot(kind='barh')
plt.title("Column wise missng value Pecentage")
```

```
Text(0.5, 1.0, 'Column wise missng value Pecentage')
```




```
# very less NA data here only AMT_PAYMENT and DAYS_ENTRY_PAYMENT have
some NA values

## For further studies we will merge it with train data with
SK_ID_CURR


installments_payments_merged =
df_train.iloc[:,:2].merge(installments_payments, on = 'SK_ID_CURR',
how = 'left')

installments_payments_merged.corr()['TARGET'].sort_values()[:10]
```

```
NUM_INSTALMENT_NUMBER    -0.016190
NUM_INSTALMENT_VERSION   -0.009896
AMT_PAYMENT              -0.003623
SK_ID_CURR               -0.002533
AMT_INSTALMENT           -0.001498
SK_ID_PREV               -0.000212
DAYS_INSTALMENT           0.034974
DAYS_ENTRY_PAYMENT        0.035122
TARGET                    1.000000
Name: TARGET, dtype: float64
```

```
installments_payments_merged.corr()
['TARGET'].sort_values(ascending=False)[:10]
```

```
TARGET                    1.000000
DAYS_ENTRY_PAYMENT        0.035122
DAYS_INSTALMENT           0.034974
SK_ID_PREV               -0.000212
AMT_INSTALMENT           -0.001498
```

```
SK_ID_CURR                -0.002533
AMT_PAYMENT               -0.003623
NUM_INSTALMENT_VERSION    -0.009896
NUM_INSTALMENT_NUMBER     -0.016190
Name: TARGET, dtype: float64
```

- DAYS_ENTRY_PAYMENT ,DAYS_INSTALMENT are showing slight correlation for TARGET

```
for i in installments_payments_merged.columns:
    print(f"For feature {i}")
    print(f"Total Uniuqe values are: {installments_payments_merged[i].nunique()}")
    print(f"Total NA values are: {installments_payments_merged[i].isna().sum()}")
    print(f"Standard Deviation is : {installments_payments_merged[i].std()}")
    print("="*100)
    print("\n \n")
```

```
For feature SK_ID_CURR
Total Uniuqe values are: 307511
Total NA values are: 0
Standard Deviation is : 102639.993713151
==========================================================================
==============================



For feature TARGET
Total Uniuqe values are: 2
Total NA values are: 0
Standard Deviation is : 0.26486506484756767
==========================================================================
==============================



For feature SK_ID_PREV
Total Uniuqe values are: 853344
Total NA values are: 15868
Standard Deviation is : 536231.442616615
==========================================================================
==============================



For feature NUM_INSTALMENT_VERSION
Total Uniuqe values are: 55
Total NA values are: 15868
Standard Deviation is : 1.0029096076264572
```

```
==========================================================================
==============================


For feature NUM_INSTALMENT_NUMBER
Total Uniuqe values are: 277
Total NA values are: 15868
Standard Deviation is : 26.49067314238672
==========================================================================
==============================


For feature DAYS_INSTALMENT
Total Uniuqe values are: 2922
Total NA values are: 15868
Standard Deviation is : 799.4779398617428
==========================================================================
==============================


For feature DAYS_ENTRY_PAYMENT
Total Uniuqe values are: 3039
Total NA values are: 18451
Standard Deviation is : 799.0789754063951
==========================================================================
==============================


For feature AMT_INSTALMENT
Total Uniuqe values are: 831108
Total NA values are: 15868
Standard Deviation is : 50331.26504090165
==========================================================================
==============================


For feature AMT_PAYMENT
Total Uniuqe values are: 875297
Total NA values are: 18451
Standard Deviation is : 54456.281886707766
==========================================================================
==============================
```

```
#Lets try to group by ID and get the median value ,However we didnt
get any very insighful results last time but it will fast the process
installments_payments_merged_grp=installments_payments_merged.groupby(
'SK_ID_PREV').median()
```

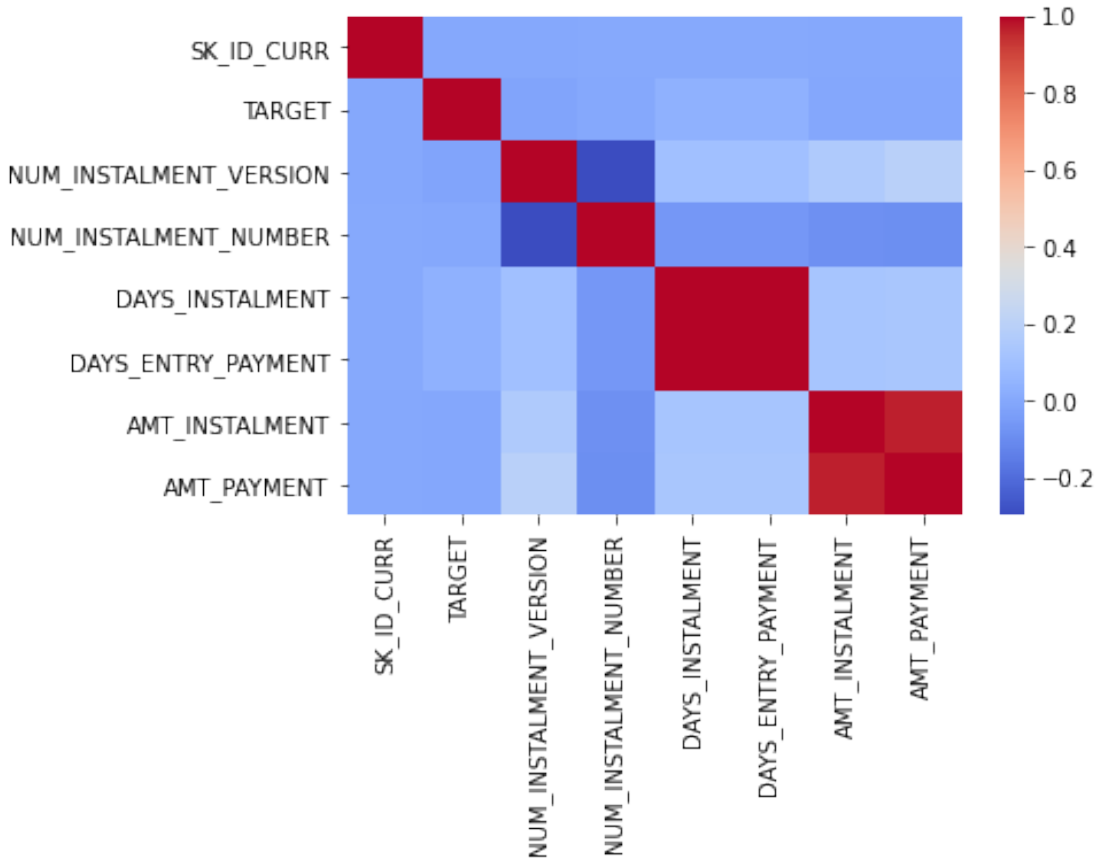
```
installments_payments_merged_grp.shape
```

```
(853344, 8)
```

```
#lets see its correlation heatmap
```

```
sns.heatmap(installments_payments_merged_grp.corr(),cmap='coolwarm')
```

```
<AxesSubplot:>
```



- AMT_INSTALLMENT is highly correlated with AMT_PAYNENT
- DAYS INSTALLMENT is higly correlated with DAYS_ENTRY_PAYMENT

NUM_INSTALMENT_VERSION

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

```
installments_payments_merged_grp['NUM_INSTALMENT_VERSION'].value_count
s()
```

```
1.0     714567
0.0      61512
2.0      33756
1.5      17519
3.0      14900
2.5       2460
5.0       2271
4.0       1795
3.5        824
4.5        704
7.0        634
6.0        499
5.5        358
6.5        276
9.0        208
8.0        185
7.5        152
8.5        102
9.5         99
10.0        85
11.0        79
10.5        70
13.0        45
12.0        45
11.5        30
13.5        20
14.0        20
12.5        18
15.0        15
16.0        13
14.5        11
15.5         8
19.0         8
18.0         7
17.5         6
20.0         5
22.5         5
17.0         5
21.0         4
16.5         4
```

```
19.5         3
18.5         3
20.5         3
23.0         2
22.0         2
26.0         2
89.5         1
23.5         1
21.5         1
24.0         1
39.0         1
Name: NUM_INSTALMENT_VERSION, dtype: int64
```

```
installments_payments_merged_grp['NUM_INSTALMENT_VERSION'].isna().sum(
)
```

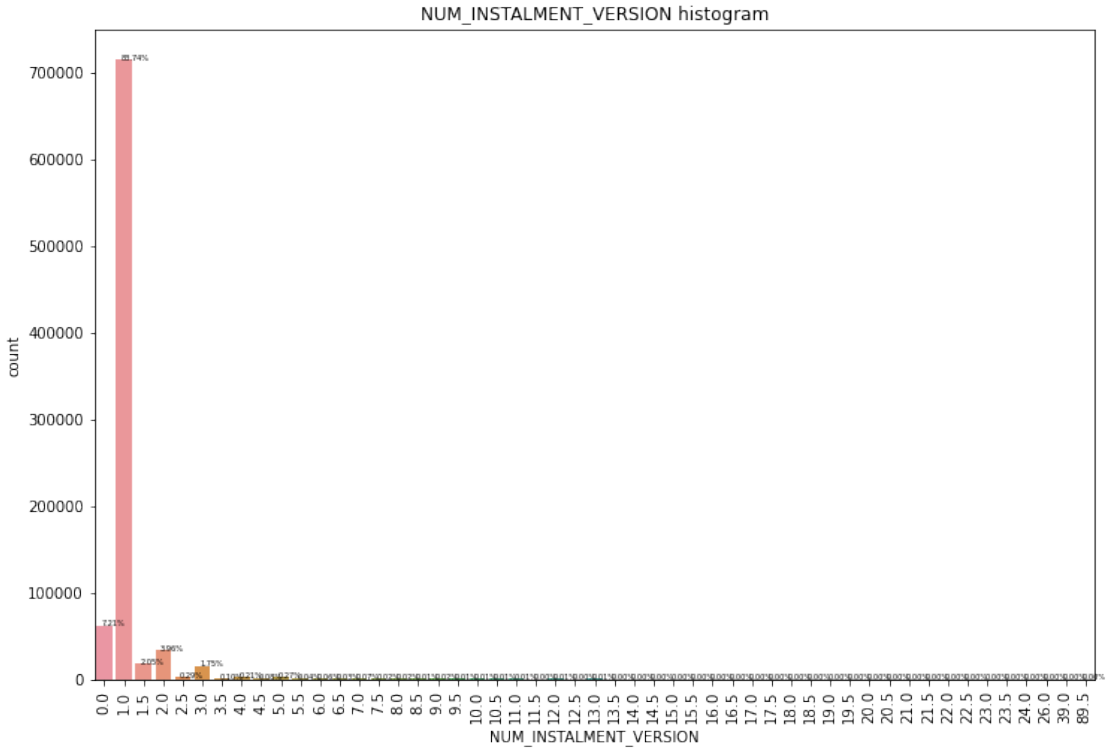
```
0
```

```
installments_payments_merged_grp['NUM_INSTALMENT_VERSION'].notnull().s
um()
```

```
853344
```

```
#lets see its distribution
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NUM_INSTALMENT_VERSION',data=installments_payments
_merged_grp)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.2f}
%'.format(p.get_height()*100/len(installments_payments_merged_grp)),
(p.get_x()+0.25, p.get_height()+0.01),size=5)
ax.set_title("NUM_INSTALMENT_VERSION histogram")
```

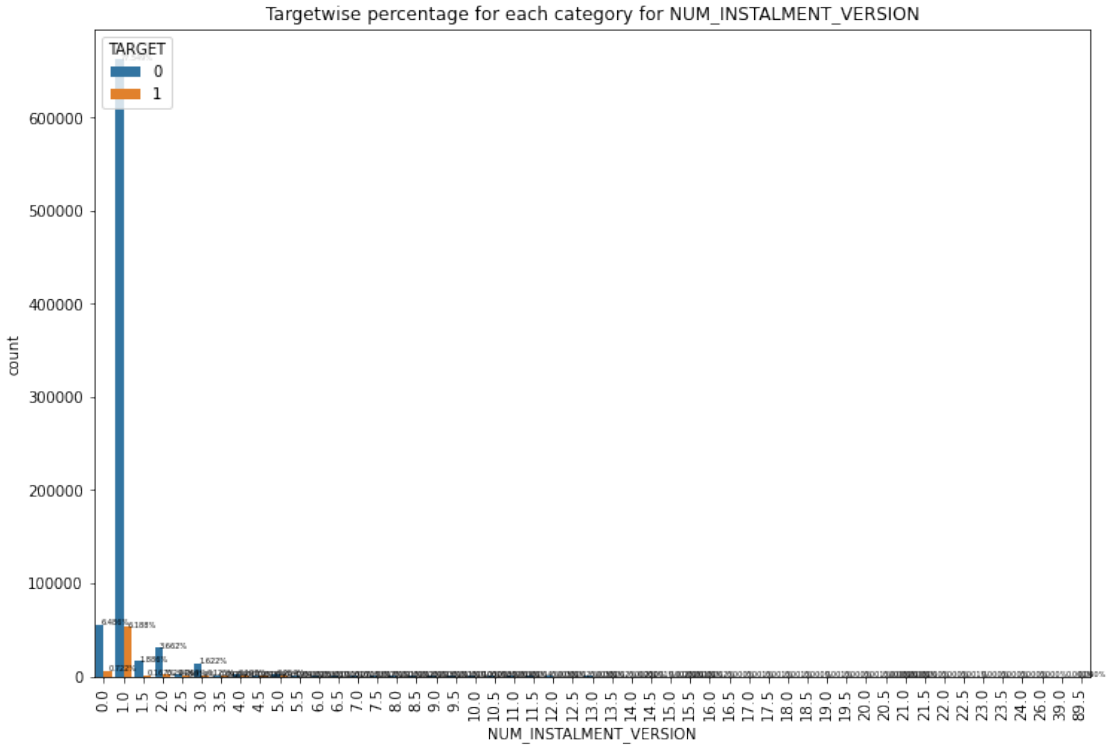
```
Text(0.5, 1.0, 'NUM_INSTALMENT_VERSION histogram')
```



- 83% NUM_INSTALMENT_VERSION belong to value 1

```
#lets see its distribution with resepect to target variables
plt.figure(figsize=(12,8))
ax=sns.countplot(x='NUM_INSTALMENT_VERSION',hue='TARGET',data=installm
ents_payments_merged_grp)
plt.xticks(rotation=90)
for p in ax.patches:
    ax.annotate('{:.3f}
%'.format(p.get_height()*100/len(installments_payments_merged_grp)),
(p.get_x()+0.25, p.get_height()+0.01),size=5)
ax.set_title("Targetwise percentage for each category for
NUM_INSTALMENT_VERSION")
```

```
Text(0.5, 1.0, 'Targetwise percentage for each category for
NUM_INSTALMENT_VERSION')
```



- Most defaulters and non defaulters belong to same values of 0 and 1 with similar proportion

DAYS_INSTALMENT

When the installment of previous credit was supposed to be paid (relative to application date of current loan)

```
installments_payments_merged_grp['DAYS_INSTALMENT'].describe()
```

```
count    853344.000000
mean      -1013.885873
std         819.320297
min       -2922.000000
25%       -1599.000000
50%        -751.000000
75%        -322.000000
max          -2.000000
Name: DAYS_INSTALMENT, dtype: float64
```

```
installments_payments_merged_grp['DAYS_INSTALMENT'].isna().sum()
```

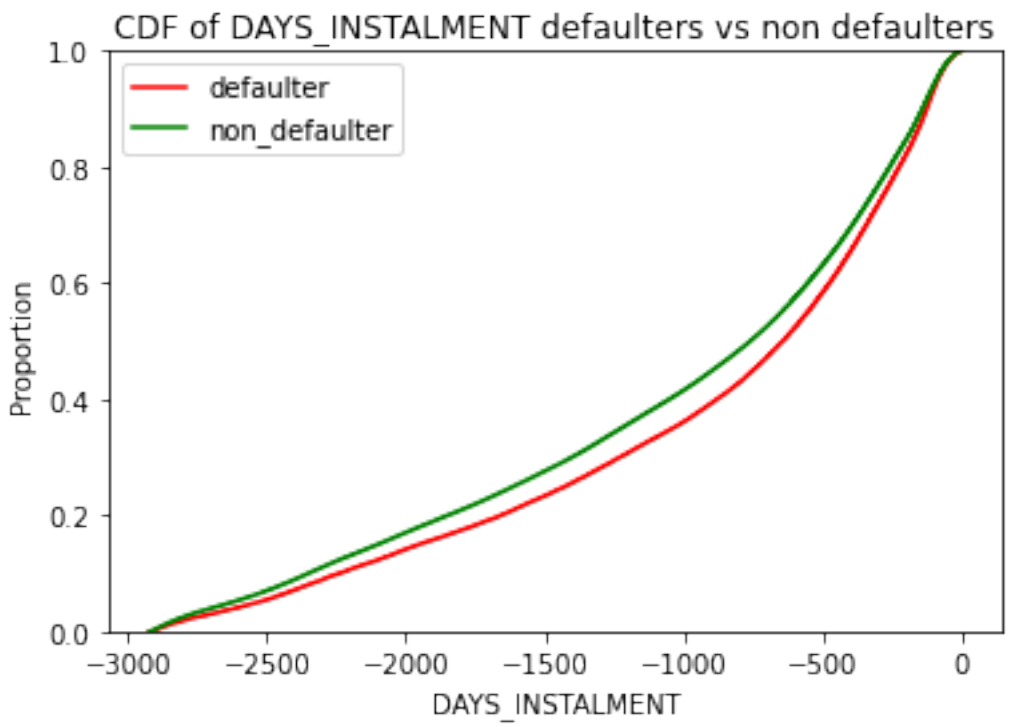
```
0
```

```
installments_payments_merged_grp['DAYS_INSTALMENT'].notnull().sum()
```

853344

```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(installments_payments_merged_grp[installments_payments_me
rged_grp['TARGET']==1]
['DAYS_INSTALMENT'],label='defaulter',color='red')
sns.ecdfplot(installments_payments_merged_grp[installments_payments_me
rged_grp['TARGET']==0]
['DAYS_INSTALMENT'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of DAYS_INSTALMENT defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1dba824a550>
```

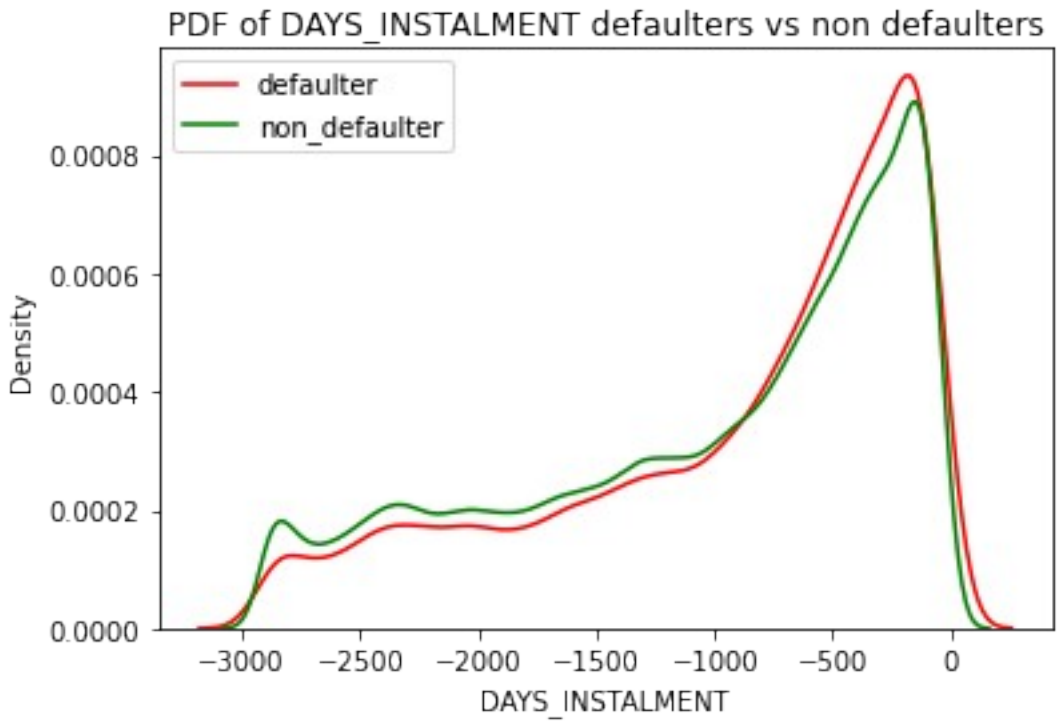



```
# This is litle insightful DAYS_INSTALMENT is liitle higher for
defaulters
```

```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(installments_payments_merged_grp[installments_payments_mer
ged_grp['TARGET']==1]
['DAYS_INSTALMENT'],label='defaulter',color='red')
sns.kdeplot(installments_payments_merged_grp[installments_payments_mer
ged_grp['TARGET']==0]
['DAYS_INSTALMENT'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of DAYS_INSTALMENT defaulters vs non defaulters")
```

```
plt.legend()
#plt.axis('off')
```

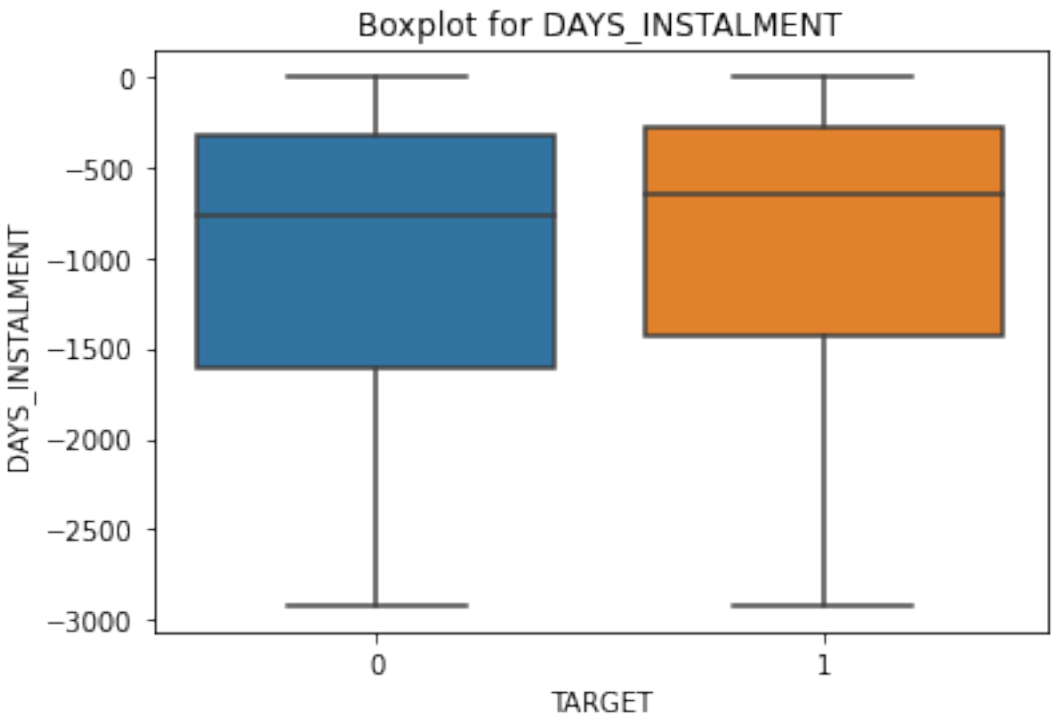
```
<matplotlib.legend.Legend at 0x1dbab89d820>
```



- from -3000 to -1000 defaulters has lower density
- from -1000 to negative values defaulters has higher density

```
sns.boxplot(x='TARGET',y='DAYS_INSTALMENT',data=installments_payments_
merged_grp)
#plt.ylim([0.0,50000])
plt.title("Boxplot for DAYS_INSTALMENT")
```

```
Text(0.5, 1.0, 'Boxplot for DAYS_INSTALMENT')
```



- upper and lower limits are same for botth defaulters and non defaulters
- defaulter's both qunatiles seems to be both qunatiles.

NUM_INSTALMENT_NUMBER

On which installment we observe payment

```
installments_payments_merged_grp['NUM_INSTALMENT_NUMBER'].describe()
```

```
count    853344.000000
mean          7.161918
std           9.041660
min           1.000000
25%           3.500000
50%           5.500000
75%           7.000000
max         162.000000
Name: NUM_INSTALMENT_NUMBER, dtype: float64
```

```
installments_payments_merged_grp['NUM_INSTALMENT_NUMBER'].notnull().su
m()
```

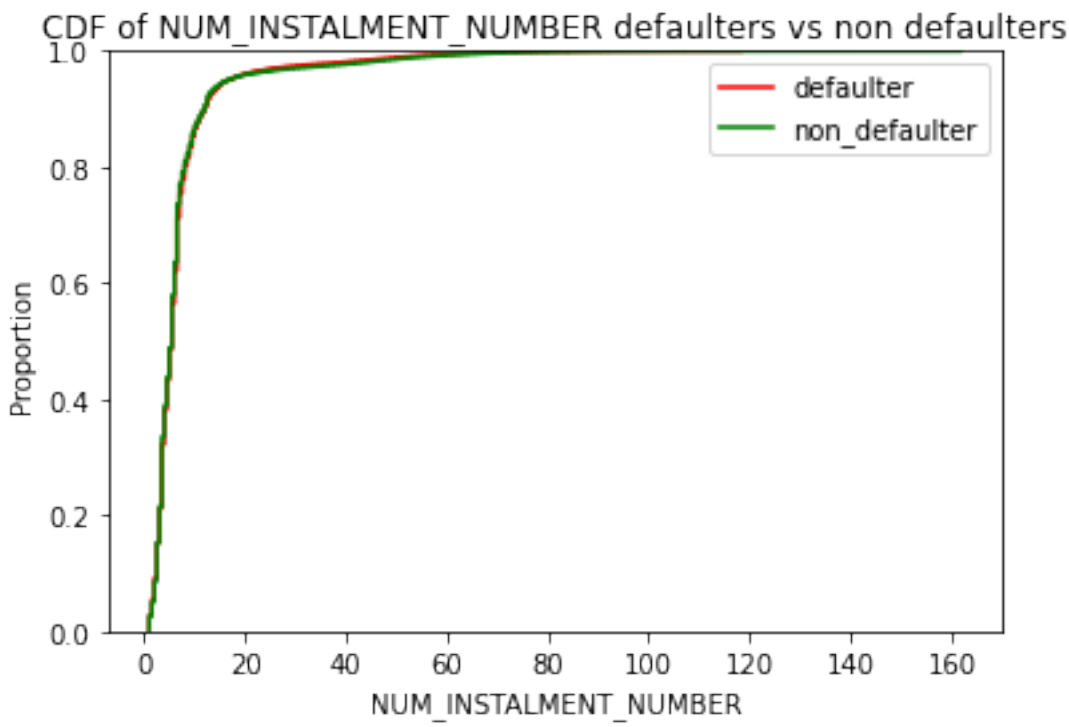
```
853344
```

```
installments_payments_merged_grp['NUM_INSTALMENT_NUMBER'].isna().sum()
```

```
0
```

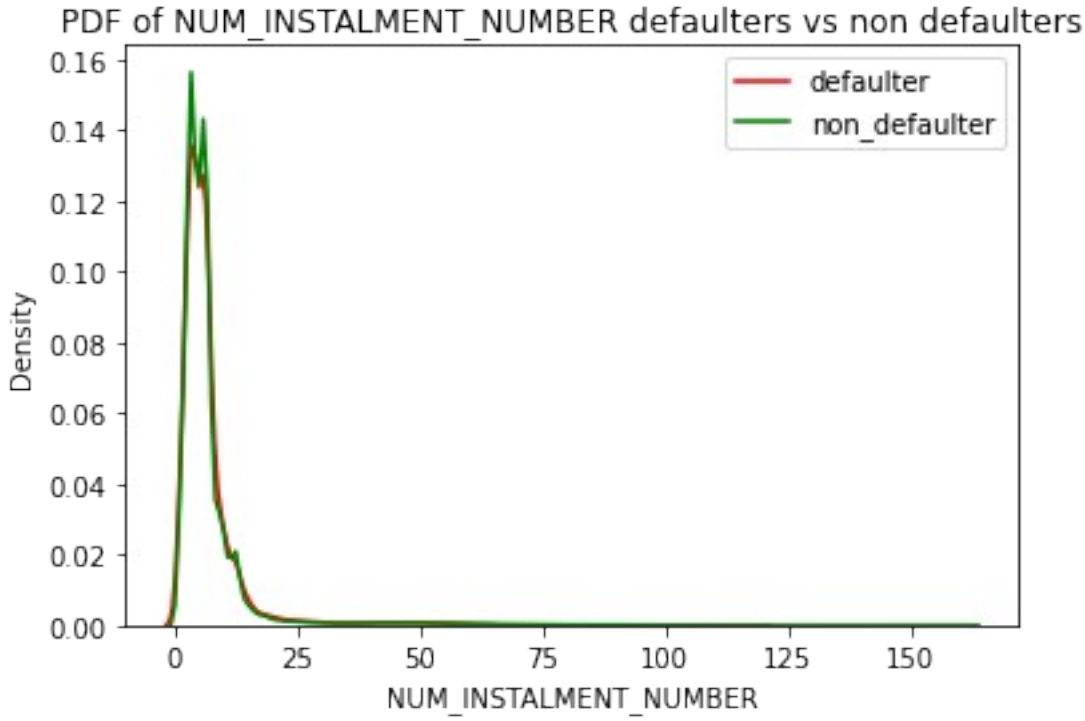
```
#Lets see its CDF plot for both defaulters and non defaulters
sns.ecdfplot(installments_payments_merged_grp[installments_payments_me
rged_grp['TARGET']==1]
['NUM_INSTALMENT_NUMBER'],label='defaulter',color='red')
sns.ecdfplot(installments_payments_merged_grp[installments_payments_me
rged_grp['TARGET']==0]
['NUM_INSTALMENT_NUMBER'],label='non_defaulter',color='green')
#plt.xlim([0,40000])
plt.title("CDF of NUM_INSTALMENT_NUMBER defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1dba8a4ba30>
```



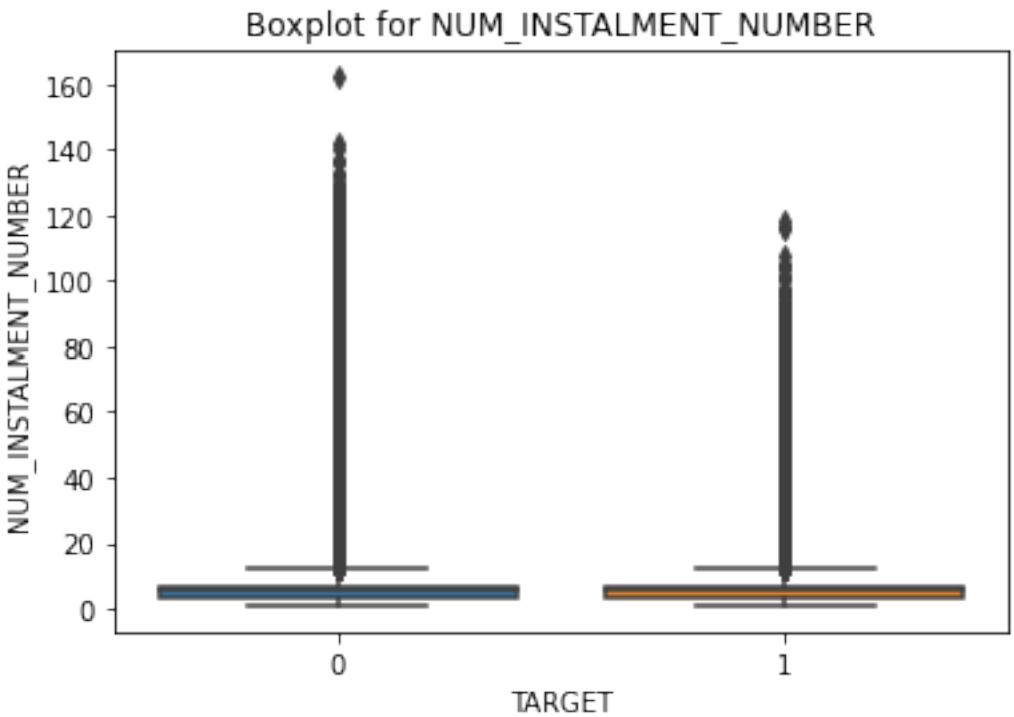

```
#Lets see its PDF plot for both defaulters and non defaulters
sns.kdeplot(installments_payments_merged_grp[installments_payments_mer
ged_grp['TARGET']==1]
['NUM_INSTALMENT_NUMBER'],label='defaulter',color='red')
sns.kdeplot(installments_payments_merged_grp[installments_payments_mer
ged_grp['TARGET']==0]
['NUM_INSTALMENT_NUMBER'],label='non_defaulter',color='green')
#plt.ylim([0,1000])
plt.title("PDF of NUM_INSTALMENT_NUMBER defaulters vs non defaulters")
plt.legend()
#plt.axis('off')
```

```
<matplotlib.legend.Legend at 0x1dba8a4bca0>
```



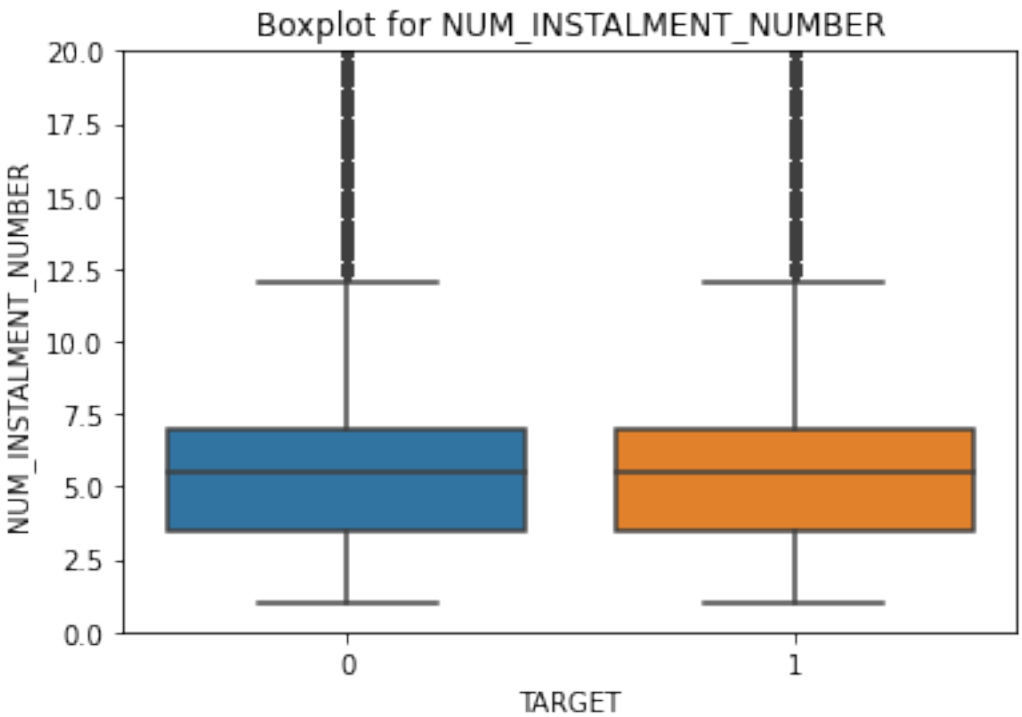

```
sns.boxplot(x='TARGET',y='NUM_INSTALMENT_NUMBER',data=installments_pay
ments_merged_grp)
#plt.ylim([0.0,20])
plt.title("Boxplot for NUM_INSTALMENT_NUMBER")
```

```
Text(0.5, 1.0, 'Boxplot for NUM_INSTALMENT_NUMBER')
```




```
sns.boxplot(x='TARGET',y='NUM_INSTALMENT_NUMBER',data=installments_pay
ments_merged_grp)
plt.ylim([0.0,20])
plt.title("Boxplot for NUM_INSTALMENT_NUMBER")
```

```
Text(0.5, 1.0, 'Boxplot for NUM_INSTALMENT_NUMBER')
```



- distribution looks similar for both TARGET Category with some potentail outlier in both category

Observation after EDA

After doing detailed and exhaustive data analysis for data with more than 6 tables (except text partially) and more than 200 columns we have below conclusion

- In application data, building details are mostly none and not so much informative.
- Categorical features are more informative features.
- We need to do more analysis in Feature Engineering to get relevant features from all relevant tables.
- There are many ways to encode the categorical features we can use HIT and TRIAL method for each way.
- we need to find a way to fill NA values in more intuitive way. Just think like a banker : "if we don't have that value from customer application than what be the default value OR Can we see any other customer having similar properties? we can fill value relative to him/her".
- We have found some features have high correlation amongst them. To avoid multicollinearity, we need to remove them and just keep one which is more informative and better variance.

- we have also found some feature who standard deviation is pretty low. This means those are not much informative ,Removing them help us to reduce the dimensionality.
- There are some features which following power law and log normal distribution, we need to work on that.
- If we closely see, even if our data is imbalanced (Normal behavior : There are always lesser Defaulters compare to good customers), Most of Defaulters financial and non financial behaviours are much similar to normal customers. We need to fina way to segregate it.I think here tree based models would perform better.
- There are some errors in some columns. e.g.: 112 years old loan (No ways).we need to remove it. Tree based model can handle it.
- There are some features which is very high range, Log transformation may help here.