Bank Loan Case Study

by

Ruchita Parmar

Project Description

This project is about finance company that provide the loan to the customer what variable is important for provide the loan and understand how data is used to minimize the risk of losing money while lending to customers

The dataset contains 3 csv file

1) application_data.csv for currant application.

2) previous_application.csv for previous status.

3) Columns_describtion.csv for columns understanding

,it has been used in this project for the analysis.

The libraries for data analysis and visualization used in this project are

Numpy, Seaborn, Metplotlib & Pandas.

a. **Missing values** Identify the missing data and use appropriate method to deal with it. (Remove columns/or replace it with an appropriate value)

b. Identify if there are **outliers** in the dataset. Also, mention why do you think it is an outlier. Again, remember that for this exercise, it is not necessary to remove any data points.

c. **Data imbalance** in the data find the ratio of misbalancing

Hint: Since there are a lot of columns, you can run your analysis in loops for the appropriate columns and find the insights

d. Explain the **results of univariate, segmented univariate, bivariate analysis, etc.** in business terms

e. Find the top **10 correlation** for the Client with payment difficulties and all other cases (Target variable). Note that you have to find the top correlation by segmenting the data frame w.r.t to the target variable and then find the top correlation for each of the segmented data and find if any insight is there. Say, there are 5+1(target) variables in a dataset: Var1, Var2, Var3, Var4, Var5, Target. And if you have to find top 3 correlation, it can be: Var1 & Var2, Var2 & Var3, Var1 & Var3. Target variable will not feature in this correlation as it is a categorical variable and not a continuous variable which is increasing or decreasing

f. **Include visualizations** and **summarize** the most important results in the presentation. You are free to choose the graphs which explain the numerical/categorical variables. Insights should explain why the variable is important for differentiating the clients with payment difficulties with all other cases

Approach and tech use

For this project I used Jupyter Notebook (Anaconda) to run my queries and charts.

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data analysis projects

A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. In other words: it's a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable.

Dataset

Import required library:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Next step read the dataset file given:

loan_app=pd.read_csv(r'D:\PROJECT_DATA\application_data.csv')

loan_app.head()

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREI
0	100002	1	Cash loans	M	N	Y	0	202500.0	40659
1	100003	0	Cash loans	F	N	N	0	270000.0	129350
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3	100006	0	Cash loans	F	N	Y	0	135000.0	31268
4	100007	0	Cash loans	M	N	Y	0	121500.0	51300

5 rows × 122 columns

Cleaning the data

We find out the number of null values in the dataset:

For column-wise null count in percent :

null_col=loan_app.isnull().sum()/len(loan_app)*100

```
SK_ID_CURR                     0.000000
TARGET                         0.000000
NAME_CONTRACT_TYPE             0.000000
CODE_GENDER                    0.000000
FLAG_OWN_CAR                   0.000000
FLAG_OWN_REALTY                0.000000
CNT_CHILDREN                   0.000000
AMT_INCOME_TOTAL               0.000000
AMT_CREDIT                     0.000000
AMT_ANNUITY                    0.003902
AMT_GOODS_PRICE                0.090403
NAME_TYPE_SUITE                0.420148
NAME_INCOME_TYPE               0.000000
NAME_EDUCATION_TYPE            0.000000
NAME_FAMILY_STATUS             0.000000
NAME_HOUSING_TYPE              0.000000
REGION_POPULATION_RELATIVE     0.000000
DAYS_BIRTH                     0.000000
DAYS_EMPLOYED                  0.000000
DAYS_REGISTRATION              0.000000
DAYS_ID_PUBLISH                0.000000
OWN_CAR_AGE                   65.990810
FLAG_MOBIL                     0.000000
FLAG_EMP_PHONE                 0.000000
FLAG_WORK_PHONE                0.000000
FLAG_CONT_MOBILE               0.000000
FLAG_PHONE                     0.000000
FLAG_EMAIL                     0.000000
OCCUPATION_TYPE               31.345545
CNT_FAM_MEMBERS                0.000650
REGION_RATING_CLIENT           0.000000
REGION_RATING_CLIENT_W_CITY    0.000000
WEEKDAY_APPR_PROCESS_START     0.000000
HOUR_APPR_PROCESS_START        0.000000
REG_REGION_NOT_LIVE_REGION     0.000000
```

```
REG_CITY_NOT_WORK_CITY          0.000000
LIVE_CITY_NOT_WORK_CITY         0.000000
ORGANIZATION_TYPE               0.000000
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
dtype: float64
```

For row-wise null count:

null_rows=loan_app.isnull().sum(axis=1).sort_values(ascending=False)

```
Out[47]: 185713    61
         133770    61
         197736    61
         116937    61
         269492    61
                   ..
         129942     0
         129929     0
         129924     0
         129911     0
         153755     0
         Length: 307511, dtype: int64
```

grafical representation of columns having % null values

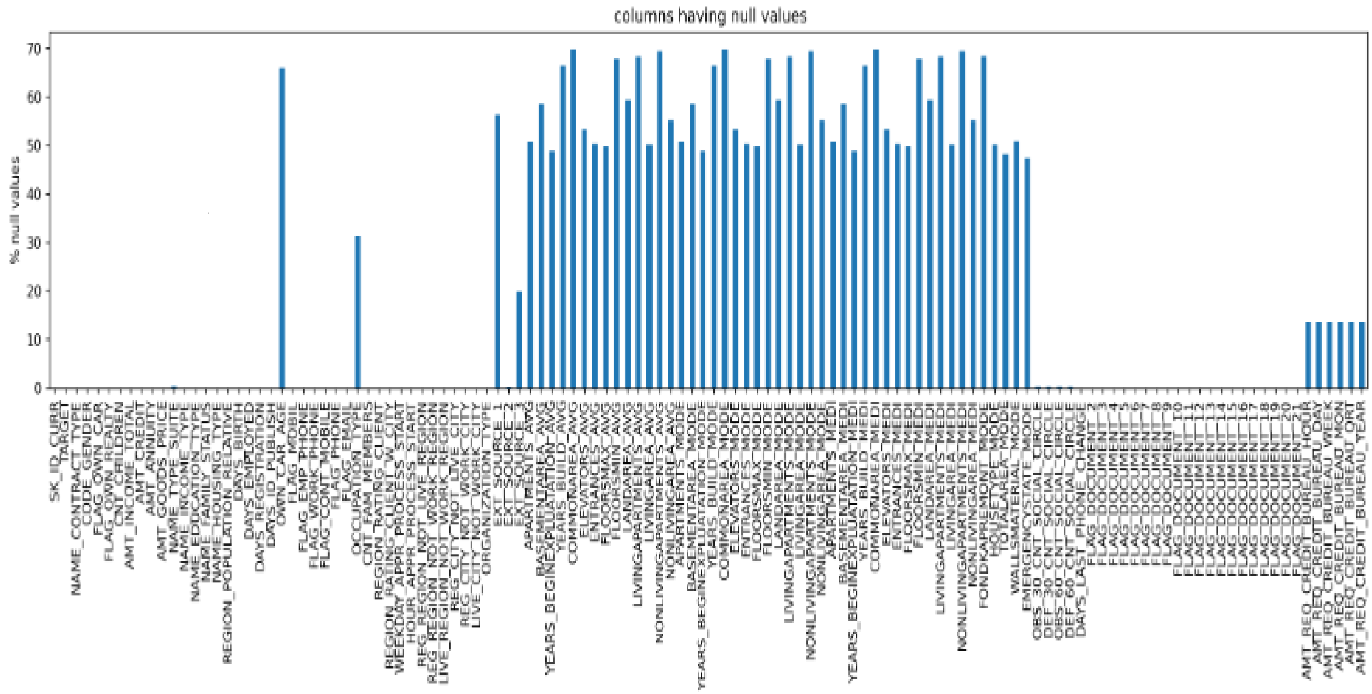
plt.figure(figsize= (20,4),dpi=300)

null_col.plot(kind = 'bar')

plt.title (' columns having null values')

plt.ylabel('% null values')

plt.show()




```
# Find the column with null values more than 45%

null_col_45 = null_col[null_col>45]

print("Number of columns having null value more than 45% :",
len(null_col_45.index))

print(null_col_45)
```

```
Number of columns having null value more than 45% : 4
OWN_CAR_AGE                     65.990810
EXT_SOURCE_1                    56.381073
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
dtype: float64
```

Now DROP the null columns having more than 45% null

loan_app = loan_app.drop(null_col_45.index, axis =1)

loan_app.shape

(307511, 73)

There are many columns which are not that important for our study so

we will drop those columns:

#List of non_relevant columns

nonrelevant=['FLAG_MOBIL','FLAG_EMP_PHONE','FLAG_WORK_PHONE','FLAG_CONT_MOBILE','FLAG_PHONE','FLAG_EMAIL', 'REGION_RATING_CLIENT','CNT_FAM_MEMBERS','REGION_RATING_CLIENT_W_CITY','DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2','FLAG_DOCUMENT_3','FLAG_DOCUMENT_4','FLAG_DOCUMENT_5','FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7','FLAG_DOCUMENT_8','FLAG_DOCUMENT_9','FLAG_DOCUMENT_10','FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12','FLAG_DOCUMENT_13','FLAG_DOCUMENT_14','FLAG_DOCUMENT_15','FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17','FLAG_DOCUMENT_18','FLAG_DOCUMENT_19','FLAG_DOCUMENT_20','FLAG_DOCUMENT_21']

#Dropping non_relevant columns from the main dataframe

loan_app.drop(labels=nonrelevant,axis=1,inplace=True) now again check the null values and fill them/replace them.

```
loan_app['ORGANIZATION_TYPE'] =
loan_app['ORGANIZATION_TYPE'].replace('XNA', 'Pensioner')

loan_app['OCCUPATION_TYPE'].fillna('Pensioner' , inplace = True)
```

EXT_SOURCE fill by median

```
loan_app.EXT_SOURCE_2.fillna(loan_app.EXT_SOURCE_2.median() , inplace = True)

loan_app.EXT_SOURCE_3.fillna(loan_app.EXT_SOURCE_3.median() , inplace = True)
```

#Imputing the undefined value of the column CODE_GENDER with F as the number is too less

```
loan_app['CODE_GENDER'] = loan_app['CODE_GENDER'].replace('XNA', 'F')

loan_app['CODE_GENDER'].fillna('F' , inplace = True)
```

check the outliers in numerical col

numerical_col = loan_app.select_dtypes(include='number').columns

len(numerical_col)

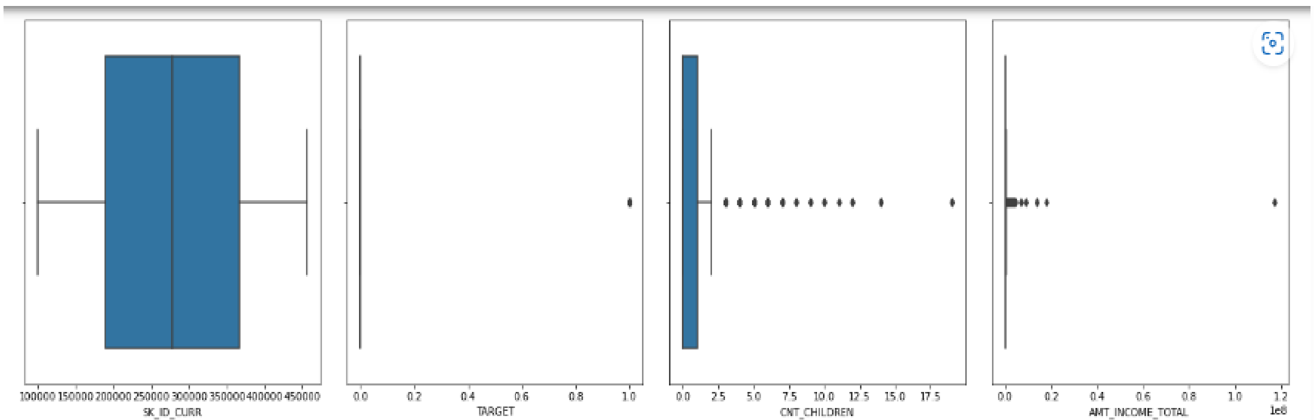
o/p: 31

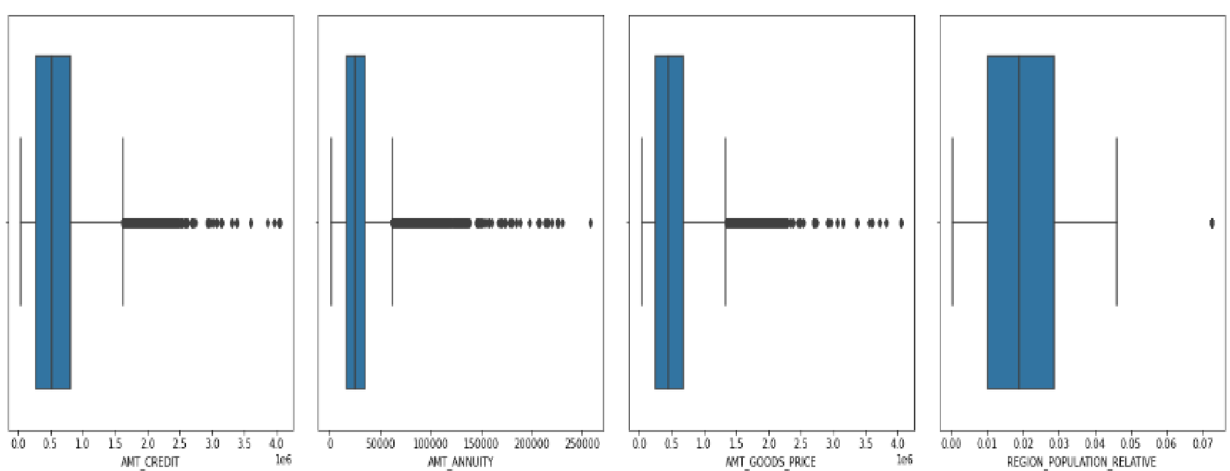
fig , axes = plt.subplots(nrows=8, ncols=4, constrained_layout=True)

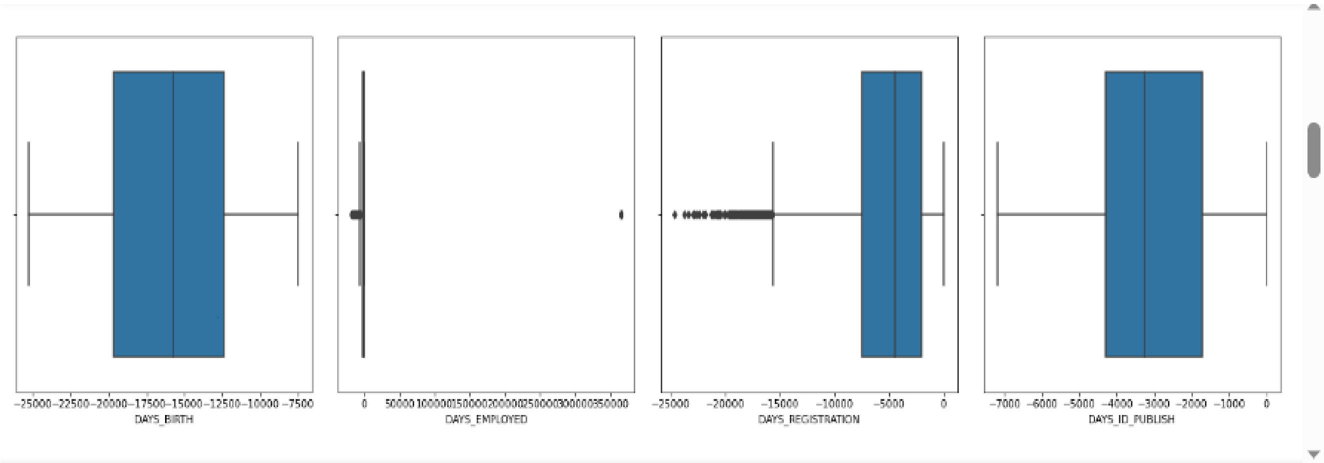
fig.subplots_adjust(left= 0, bottom=0, right=3, top=12, wspace=0.09, hspace=0.3)

for ax, column in zip(axes.flatten(),numerical_col):   #Using For loop

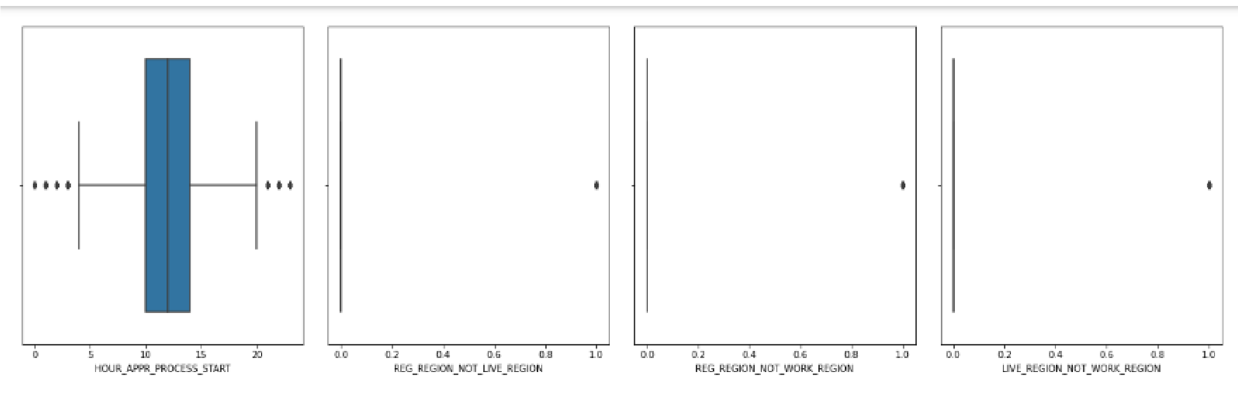
sns.boxplot(loan_app[column],ax=ax)   #Ploting



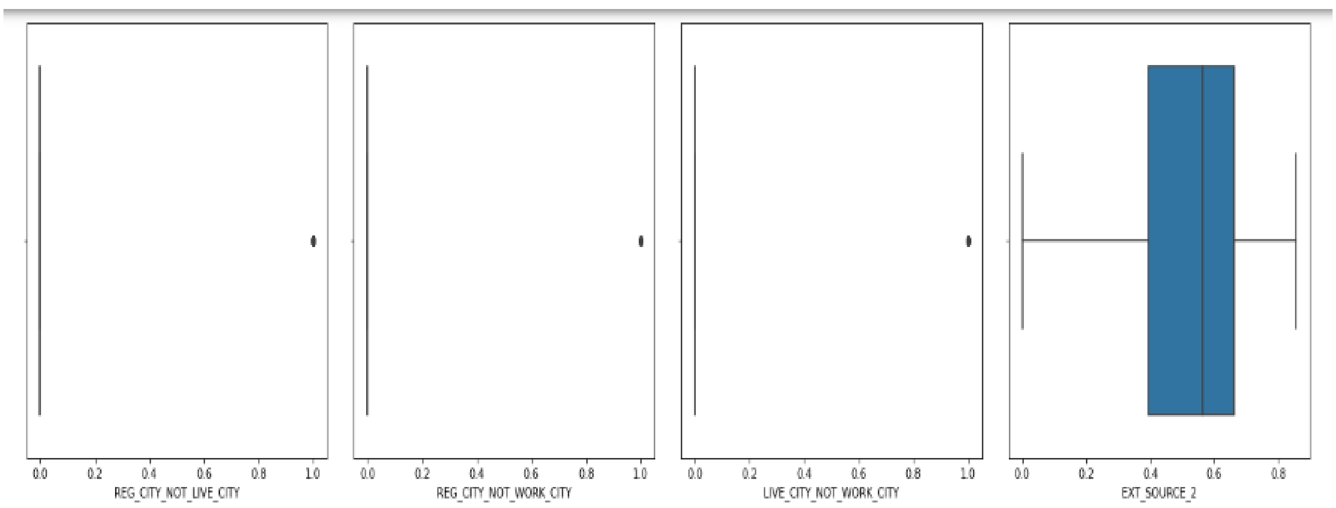




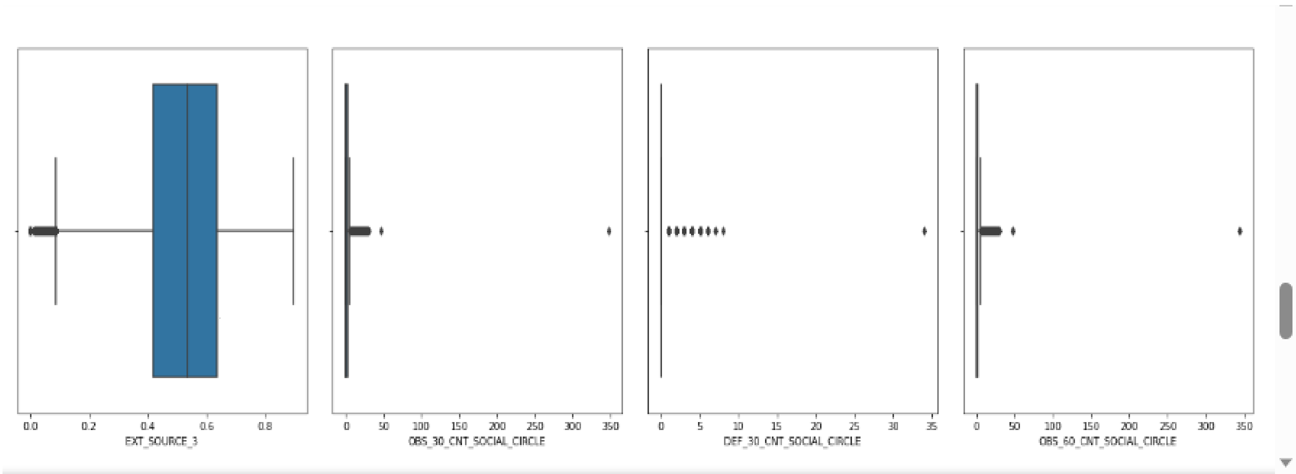
−25000 −22500 −20000 −17500 −15000 −12500 −10000 −7500
DAYS_BIRTH
0 50000 100000 150000 200000 250000 300000 350000
DAYS_EMPLOYED
−25000 −20000 −15000 −10000 −5000 0
DAYS_REGISTRATION
−7000 −6000 −5000 −4000 −3000 −2000 −1000 0
DAYS_ID_PUBLISH

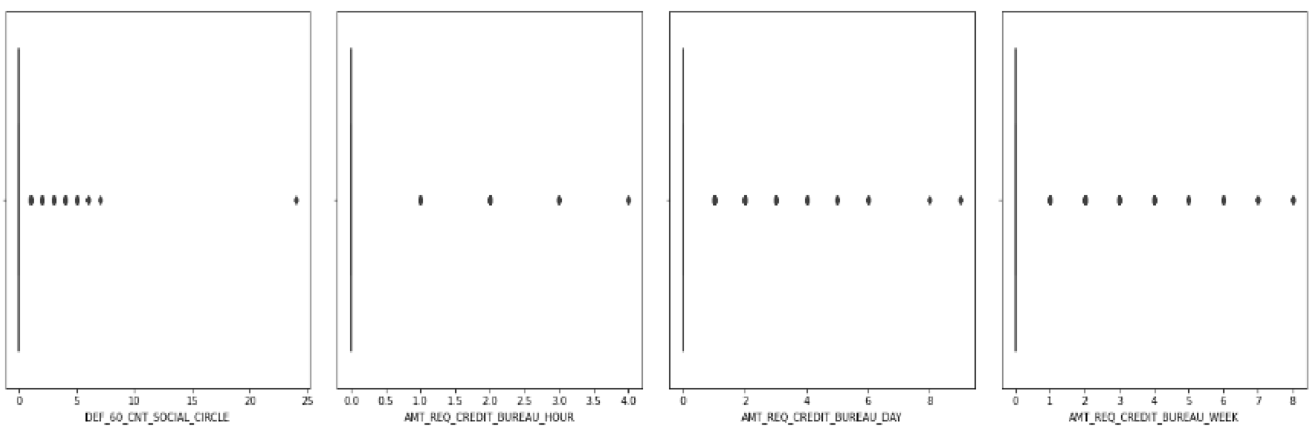


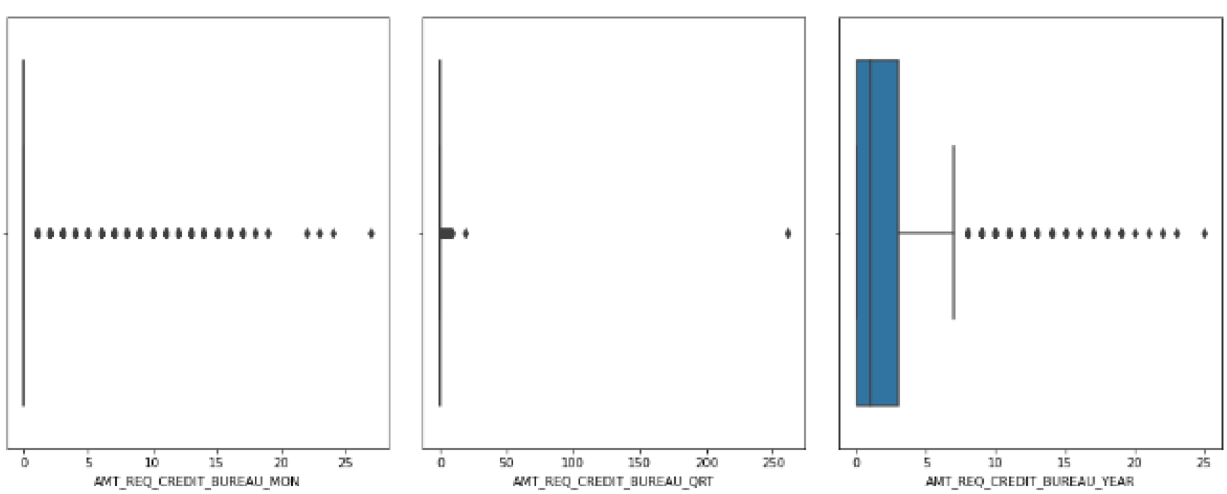
0 5 10 15 20
HOUR_APPR_PROCESS_START
0.0 0.2 0.4 0.6 0.8 1.0
REG_REGION_NOT_LIVE_REGION
0.0 0.2 0.4 0.6 0.8 1.0
REG_REGION_NOT_WORK_REGION
0.0 0.2 0.4 0.6 0.8 1.0
LIVE_REGION_NOT_WORK_REGION



0.0 0.2 0.4 0.6 0.8 1.0
REG_CITY_NOT_LIVE_CITY
0.0 0.2 0.4 0.6 0.8 1.0
REG_CITY_NOT_WORK_CITY
0.0 0.2 0.4 0.6 0.8 1.0
LIVE_CITY_NOT_WORK_CITY
0.0 0.2 0.4 0.6 0.8
EXT_SOURCE_2


EXT_SOURCE_3
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
OBS_60_CNT_SOCIAL_CIRCLE


DEF_60_CNT_SOCIAL_CIRCLE
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

Data imbalance

Target0 = loan_app.loc[loan_app["TARGET"]==0]

Target1 = loan_app.loc[loan_app["TARGET"]==1]

print("datat imbalance ratio is")

round(len(Target0)/len(Target1),2)

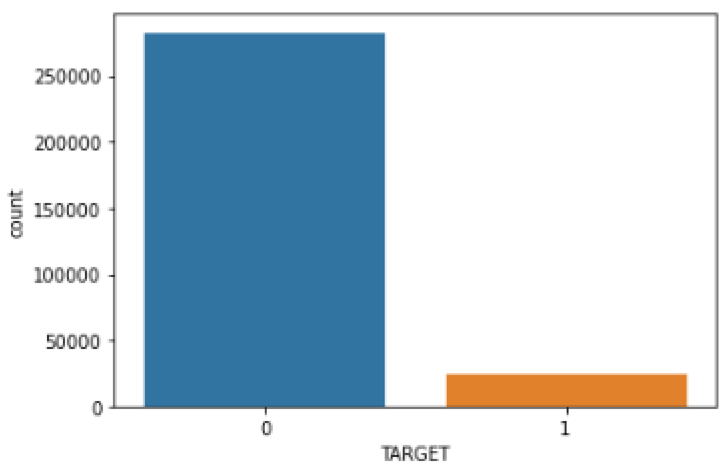
```
datat imbalance ratio is

11.39
```

Imbalance in our target variable

sns.countplot(loan_app["TARGET"])

loan_app["TARGET"].value_counts()

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



WE CAN CLEARLY SEEN THE DATA IMBALNCING IN OUR TARGET VARIABLE,PEOPLE PAY THEIR LOAN MORE THAN PEOPLE NOT PAY THEIR LOAN

NOW SEGRIGATE THE INCOME RANGE and CREDIT RANGE

```
bins = [0,100000,200000,300000,400000,500000,600000,700000,800000,900000,10000000000]

slot = ['<100000','100000-200000','200000-300000','300000-400000','400000-500000', '500000-600000', '600000-700000','700000-800000','850000-900000','900000 and above']

loan_app['AMT_CREDIT_RANGE']=pd.cut(loan_app['AMT_CREDIT'],bins,labels=slot)
```

For INCOME RANGE

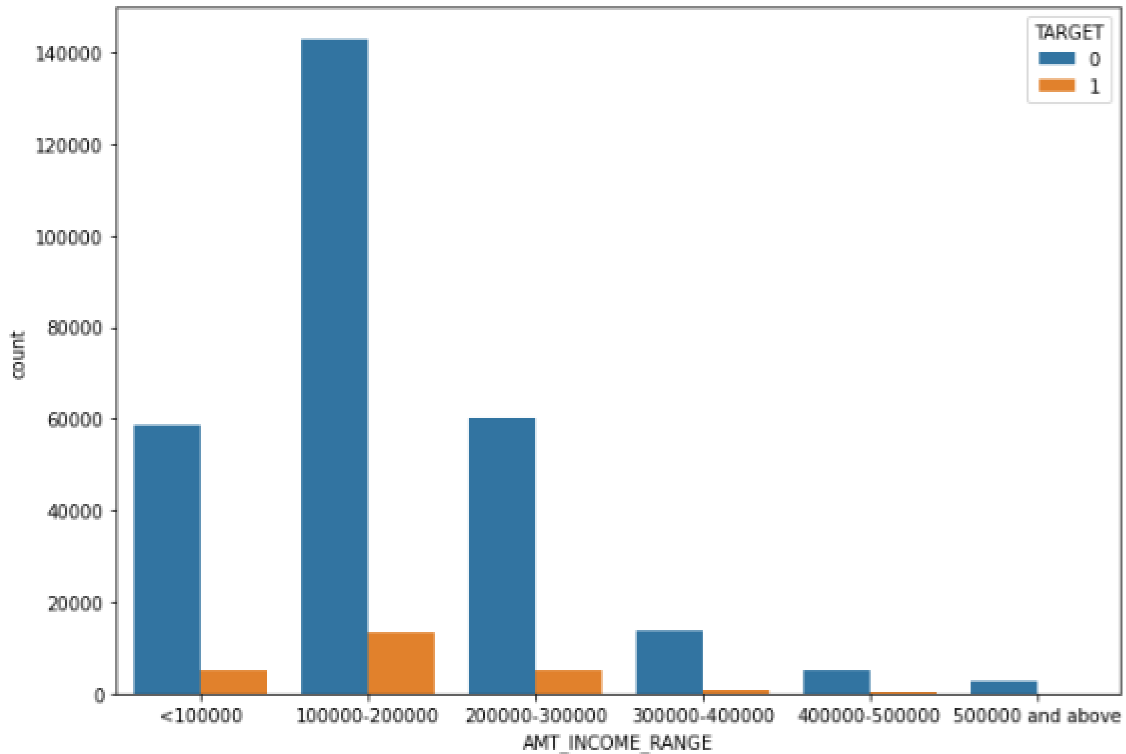
```
bins = [0,100000,200000,300000,400000,500000,10000000000]

slot = ['<100000', '100000-200000','200000-300000','300000-400000','400000-500000', '500000 and above']

loan_app['AMT_INCOME_RANGE']=pd.cut(loan_app['AMT_INCOME_TOTAL'],bins,labels=slot)
```

univariant analysis for numerical and categorical columns

plt.figure(figsize=(10,7))

sns.countplot(x ='AMT_INCOME_RANGE', data=loan_app,hue="TARGET")

```
<AxesSubplot:xlabel='AMT_INCOME_RANGE', ylabel='count'>
```

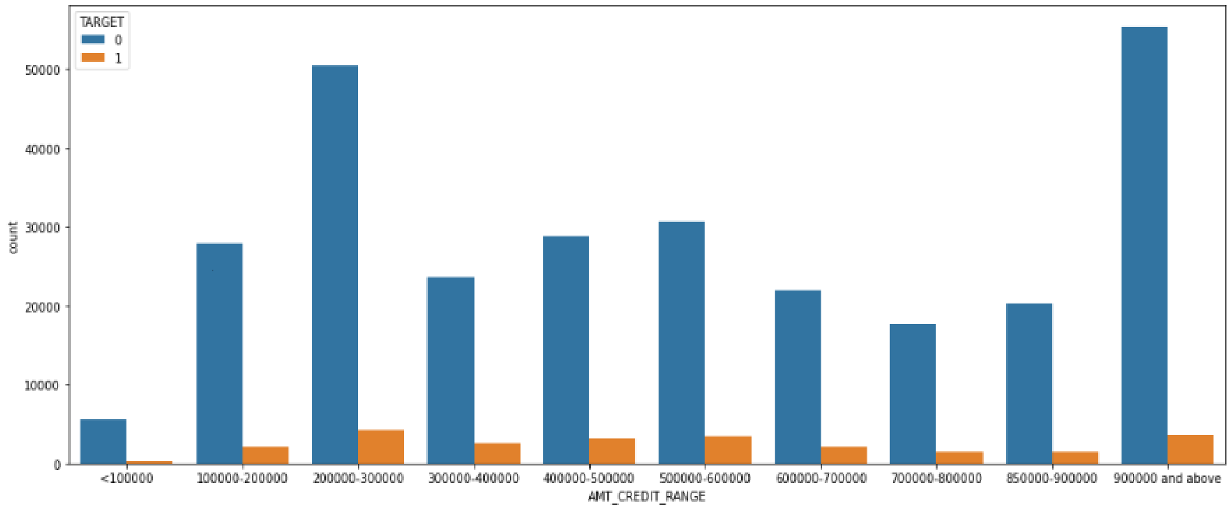


The people having 100000-200000 are having higher number of loan and also having higher in defaulter Then income segment having >500000 are having less defaulter.

plt.figure(figsize=(18,7))

sns.countplot(x ='AMT_CREDIT_RANGE', data=loan_app,hue="TARGET")

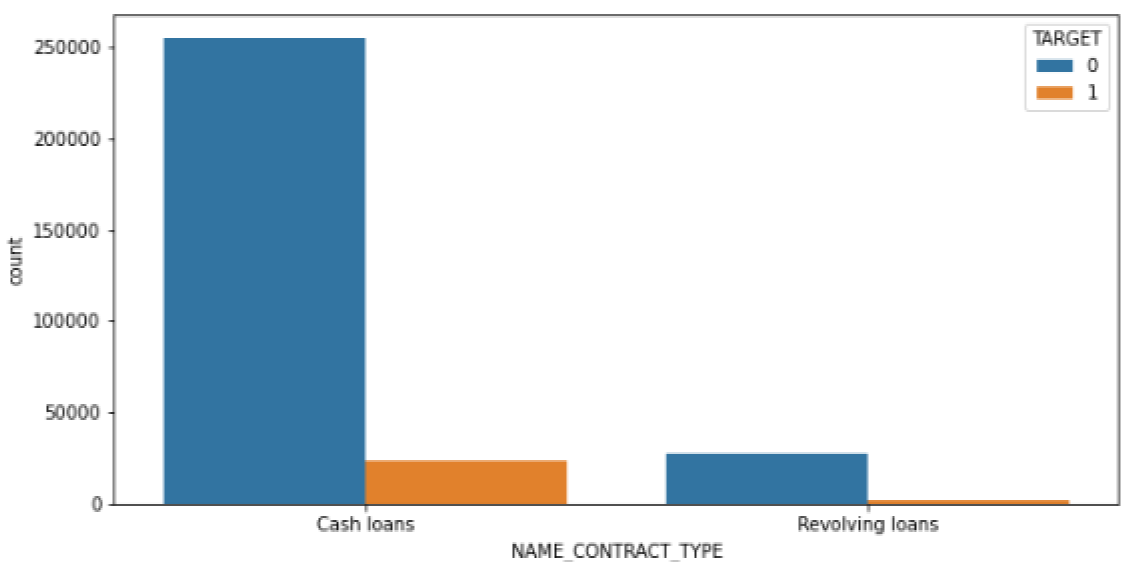
```
<AxesSubplot:xlabel='AMT_CREDIT_RANGE', ylabel='count'>
```



<100000 having less defaulter >900000 more defaulter

```
plt.figure(figsize=(10,5))

sns.countplot(x ='NAME_CONTRACT_TYPE', data=loan_app,hue="TARGET")
```

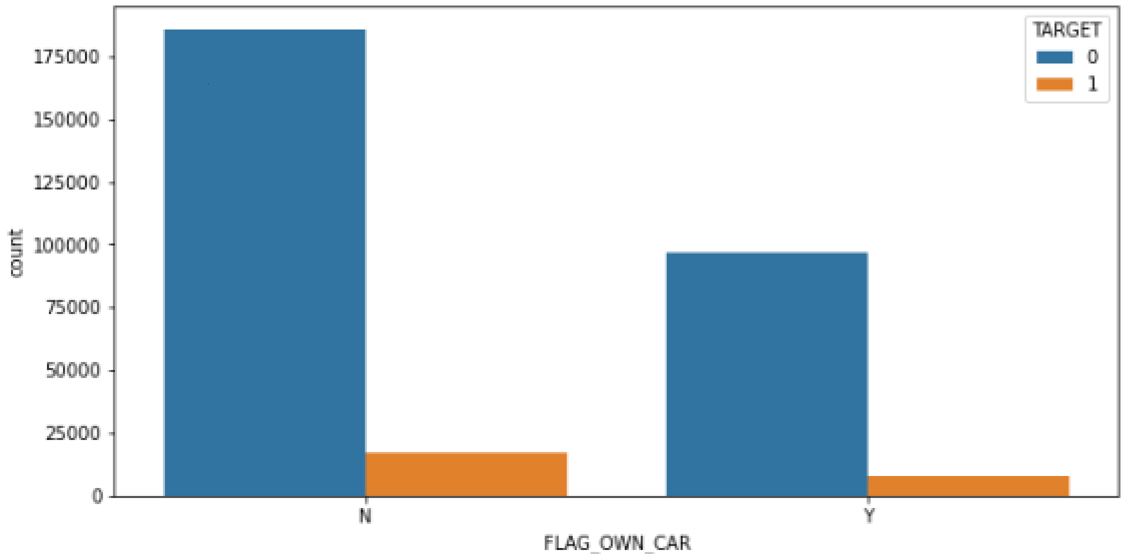
```
<AxesSubplot:xlabel='NAME_CONTRACT_TYPE', ylabel='count'>
```



plt.figure(figsize=(10,5))

sns.countplot(x ='FLAG_OWN_CAR', data =loan_app,hue="TARGET")

```
<AxesSubplot:xlabel='FLAG_OWN_CAR', ylabel='count'>
```

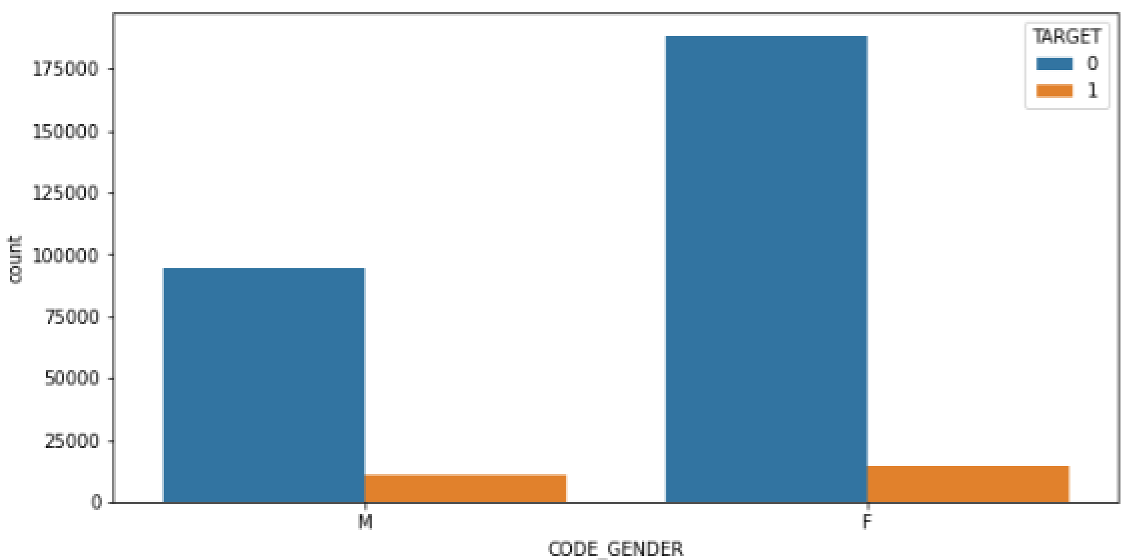


---

plt.figure(figsize=(10,5))

sns.countplot(x ='CODE_GENDER', data =loan_app,hue="TARGET")

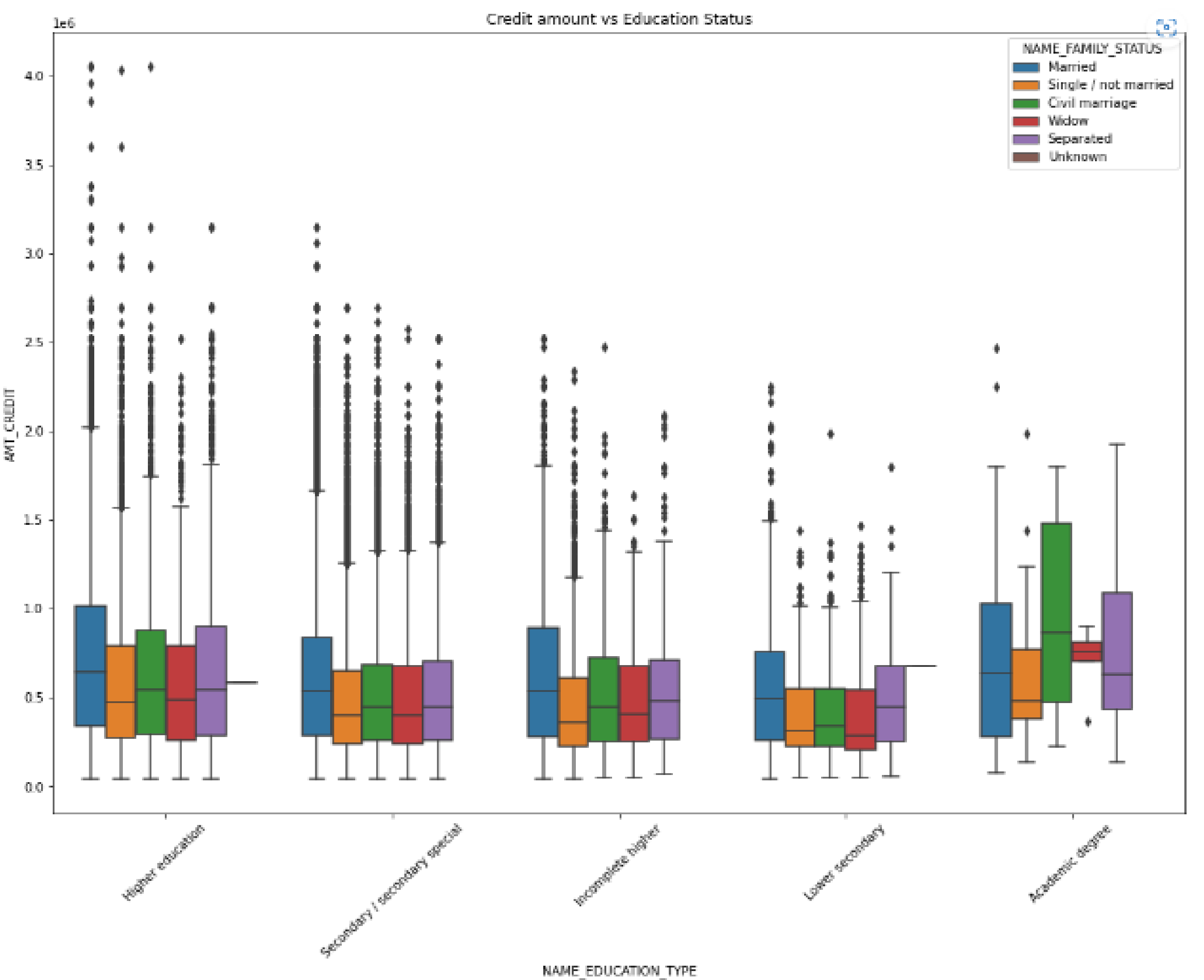
```
<AxesSubplot:xlabel='CODE_GENDER', ylabel='count'>
```



The % of defaulters are more in Male than Female

Bivariate analysis

```
plt.figure(figsize=(16,12))

plt.xticks(rotation=45)

sns.boxplot(data =Target0,
x='NAME_EDUCATION_TYPE',y='AMT_CREDIT', hue
='NAME_FAMILY_STATUS',orient='v')

plt.title('Credit amount vs Education Status')

plt.show()
```



Family status of 'civil marriage', 'marriage' and 'separated' of Academic degree education are having higher number of credits than others. Also, higher education of family status of 'marriage', 'single' and 'civil marriage' are having more outliers. Civil marriage for Academic degree is having most of the credits in the third quartile.

 # Top 10 correlated variables: target 0 dataaframe

```
corr = Target0.corr()

corrdf = corr.where(np.triu(np.ones(corr.shape),
k=1).astype(np.bool))

corrdf = corrdf.unstack().reset_index()

corrdf.columns = ['Var1', 'Var2', 'Correlation']

corrdf.dropna(subset = ['Correlation'], inplace = True)

corrdf['Correlation'] = round(corrdf['Correlation'], 2)

corrdf['Correlation'] = abs(corrdf['Correlation'])

corrdf.sort_values(by = 'Correlation', ascending = False).head(10)
```

	Var1	Var2	Correlation
734	OBS_60_CNT_SOCIAL_CIRCLE	OBS_30_CNT_SOCIAL_CIRCLE	1.00
190	AMT_GOODS_PRICE	AMT_CREDIT	0.99
479	LIVE_REGION_NOT_WORK_REGION	REG_REGION_NOT_WORK_REGION	0.86
766	DEF_60_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE	0.86
575	LIVE_CITY_NOT_WORK_CITY	REG_CITY_NOT_WORK_CITY	0.83
191	AMT_GOODS_PRICE	AMT_ANNUITY	0.78
159	AMT_ANNUITY	AMT_CREDIT	0.77
287	DAYS_EMPLOYED	DAYS_BIRTH	0.62
447	REG_REGION_NOT_WORK_REGION	REG_REGION_NOT_LIVE_REGION	0.45
543	REG_CITY_NOT_WORK_CITY	REG_CITY_NOT_LIVE_CITY	0.44

#Top 10 correlated variables: target 1 dataaframe

```
corr = Target1.corr()

corrdf = corr.where(np.triu(np.ones(corr.shape),
k=1).astype(np.bool))

corrdf = corrdf.unstack().reset_index()

corrdf.columns = ['Var1', 'Var2', 'Correlation']

corrdf.dropna(subset = ['Correlation'], inplace = True)

corrdf['Correlation'] = round(corrdf['Correlation'], 2)

corrdf['Correlation'] = abs(corrdf['Correlation'])

corrdf.sort_values(by = 'Correlation', ascending = False).head(10)
```

	Var1	Var2	Correlation
734	OBS_60_CNT_SOCIAL_CIRCLE	OBS_30_CNT_SOCIAL_CIRCLE	1.00
190	AMT_GOODS_PRICE	AMT_CREDIT	0.98
766	DEF_60_CNT_SOCIAL_CIRCLE	DEF_30_CNT_SOCIAL_CIRCLE	0.87
479	LIVE_REGION_NOT_WORK_REGION	REG_REGION_NOT_WORK_REGION	0.85
575	LIVE_CITY_NOT_WORK_CITY	REG_CITY_NOT_WORK_CITY	0.78
159	AMT_ANNUITY	AMT_CREDIT	0.75
191	AMT_GOODS_PRICE	AMT_ANNUITY	0.75
287	DAYS_EMPLOYED	DAYS_BIRTH	0.58
447	REG_REGION_NOT_WORK_REGION	REG_REGION_NOT_LIVE_REGION	0.50
543	REG_CITY_NOT_WORK_CITY	REG_CITY_NOT_LIVE_CITY	0.47

From the above correlation analysis it is infered that the highest corelation (1.0) is between (OBS_60_CNT_SOCIAL_CIRCLE with OBS_30_CNT_SOCIAL_CIRCLE) same for both.

Read “Previous Application” data and merging with “application data”

All abow process done with previous application dataset

```
pre_app.head()
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	WEEKD
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN	607500.0	
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN	112500.0	
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	NaN	450000.0	
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	NaN	337500.0	

5 rows × 37 columns

 # Finding percentage of null values in columns

```
null_val=pre_app.isnull().sum()/len(pre_app)*100

null_val
```

```
SK_ID_PREV                      0.000000
SK_ID_CURR                      0.000000
NAME_CONTRACT_TYPE              0.000000
AMT_ANNUITY                    22.286665
AMT_APPLICATION                 0.000000
AMT_CREDIT                      0.000060
AMT_DOWN_PAYMENT               53.636480
AMT_GOODS_PRICE                23.081773
WEEKDAY_APPR_PROCESS_START      0.000000
HOUR_APPR_PROCESS_START         0.000000
FLAG_LAST_APPL_PER_CONTRACT     0.000000
NFLAG_LAST_APPL_IN_DAY          0.000000
RATE_DOWN_PAYMENT              53.636480
RATE_INTEREST_PRIMARY          99.643698
RATE_INTEREST_PRIVILEGED       99.643698
NAME_CASH_LOAN_PURPOSE          0.000000
NAME_CONTRACT_STATUS            0.000000
DAYS_DECISION                   0.000000
NAME_PAYMENT_TYPE               0.000000
CODE_REJECT_REASON              0.000000
NAME_TYPE_SUITE                49.119754
NAME_CLIENT_TYPE                0.000000
NAME_GOODS_CATEGORY             0.000000
NAME_PORTFOLIO                  0.000000
NAME_PRODUCT_TYPE               0.000000
CHANNEL_TYPE                    0.000000
SELLERPLACE_AREA                0.000000
NAME_SELLER_INDUSTRY            0.000000
CNT_PAYMENT                    22.286366
NAME_YIELD_GROUP                0.000000
PRODUCT_COMBINATION             0.020716
DAYS_FIRST_DRAWING             40.298129
DAYS_FIRST_DUE                 40.298129
DAYS_LAST_DUE_1ST_VERSION      40.298129
DAYS_LAST_DUE                  40.298129
```

Graphical presentation of null values columns wise.

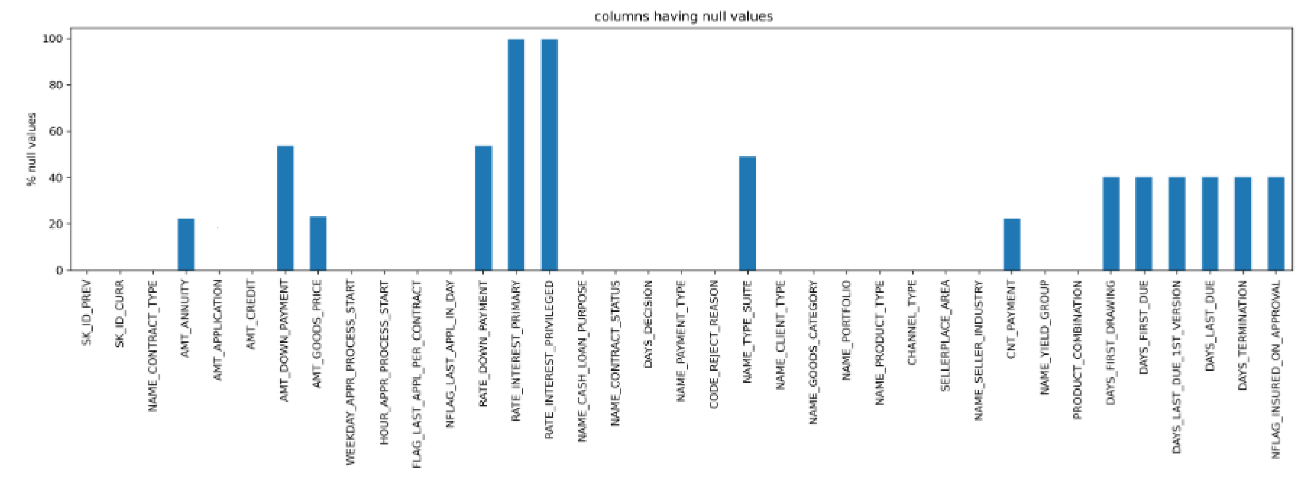
plt.figure(figsize= (20,4),dpi=300)

null_val.plot(kind = 'bar')

plt.title (' columns having null values')

plt.ylabel('% null values')

plt.show()



#Get the column with null values more than 50%

null_val = null_val[null_val>50]

print("Number of columns having null value more than 50% :", len(null_val.index))

print(null_val)

```
Number of columns having null value more than 50% : 4
AMT_DOWN_PAYMENT           53.636480
RATE_DOWN_PAYMENT          53.636480
RATE_INTEREST_PRIMARY      99.643698
RATE_INTEREST_PRIVILEGED   99.643698
dtype: float64
```

 # drop 4 columns having null percentage more than 50%.

pre_app = pre_app.drop(null_val.index, axis =1)

pre_app.shape

(1670214, 33)

 # Merging the Application dataset with previous application dataset

```
comb_data =
pd.merge(left=loan_app,right=pre_app,how='inner',on='SK_ID_CURR',suff
ixes='_x')

comb_data.shape

(1413701, 77)
```

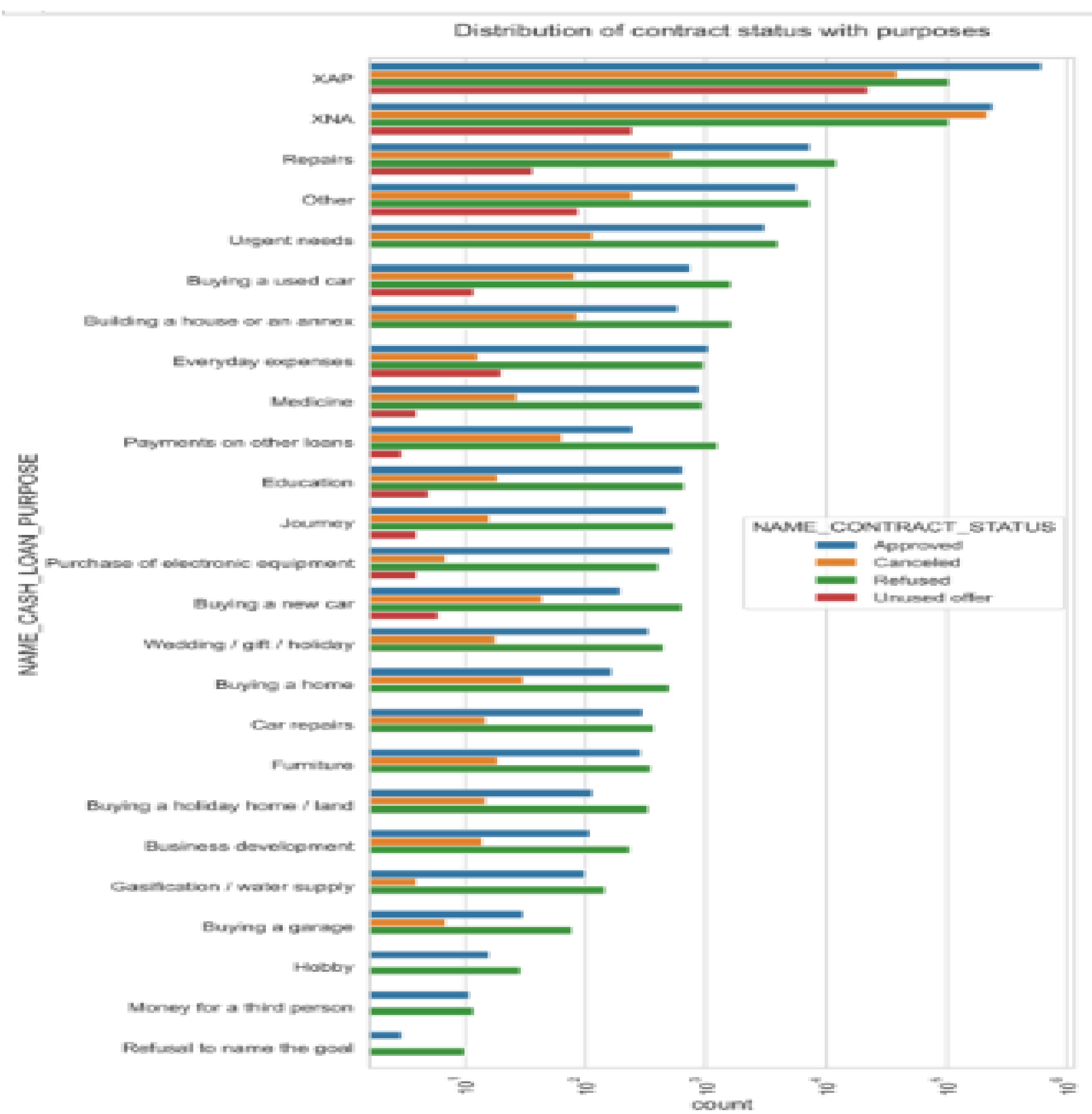
#Removing unwanted columns from combined dataframe for analysis

```
comb_data.drop(['SK_ID_CURR','WEEKDAY_APPR_PROCESS_STARTx', '
HOUR_APPR_PROCESS_STARTx','REG_REGION_NOT_LIVE_REGION',
                  'REG_REGION_NOT_WORK_REGION','LIVE_REGION_NO
T_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
                  'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_
CITY',
                   'FLAG_LAST_APPL_PER_CONTRACT','NFLAG_LAST_AP
PL_IN_DAY'],axis=1,inplace=True)
```

 # Distribution of contract status in logarithmic scale

```
sns.set_style('whitegrid')

sns.set_context('talk')

plt.figure(figsize=(10,25),dpi = 300)

plt.rcParams["axes.labelsize"] = 20
```

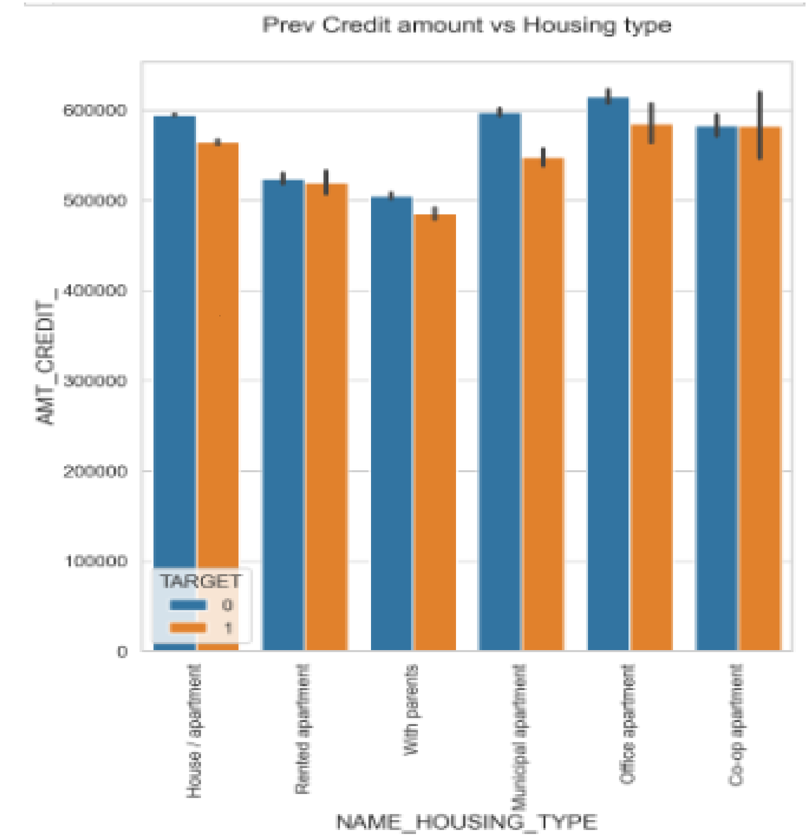
```
plt.rcParams['axes.titlesize'] = 22

plt.rcParams['axes.titlepad'] = 30

plt.xticks(rotation=90)

plt.xscale('log')

plt.title('Distribution of contract status with purposes')

ax = sns.countplot(data = comb_data, y=
'NAME_CASH_LOAN_PURPOSE',
order=comb_data['NAME_CASH_LOAN_PURPOSE'].value_counts().index
,hue = 'NAME_CONTRACT_STATUS')
```



Loan purposes with 'Repairs' are facing more difficulties in payment on time. There are few places where loan payment is significant higher than facing difficulties. They are 'Buying a garage', 'Business development', 'Buying land', 'Buying a new car' and 'Education' Hence we can focus on these purposes for which the client is having for minimal payment difficulties.

Bivariate

```
# Box plotting for Credit amount prev vs Housing type in logarithmic scale

plt.figure(figsize=(10,11),dpi = 150)

plt.xticks(rotation=90)

sns.barplot(data =comb_data, y='AMT_CREDIT_',hue='TARGET',x='NAME_HOUSING_TYPE')

plt.title('Prev Credit amount vs Housing type')

plt.show()
```


Prev Credit amount vs Housing type
AMT_CREDIT_
600000
500000
400000
300000
200000
100000
0
TARGET
0
1
House / apartment
Rented apartment
With parents
Municipal apartment
Office apartment
Co-op apartment
NAME_HOUSING_TYPE

Thank you