EDA CASE STUDY

PRESENTATION

STEP I – Loading the Data

- ➔ As the First and foremost step after **importing the necessary libraries**, the .CSV file which hold the data (“application_data.csv”) is loaded using the ‘read_csv’ feature of Pandas.

- ➔ The **‘.head()’** feature is then used on the Data-frame, just to get a rough idea of the contents of the data set.

```
df.head()
```

Out[79]:

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

- ➔ The very useful **‘.info()’** feature is applied on the Data-frame and the datatypes of every column of the Dataset is observed.

```
In [80]: df.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 122 columns):
 #    Column                        Dtype
---   ------                        -----
 0    SK_ID_CURR                    int64
 1    TARGET                        int64
 2    NAME_CONTRACT_TYPE            object
 3    CODE_GENDER                   object
 4    FLAG_OWN_CAR                  object
 5    FLAG_OWN_REALTY               object
 6    CNT_CHILDREN                  int64
 7    AMT_INCOME_TOTAL              float64
 8    AMT_CREDIT                    float64
 9    AMT_ANNUITY                   float64
 10   AMT_GOODS_PRICE               float64
 11   NAME_TYPE_SUITE               object
 12   NAME_INCOME_TYPE              object
 13   NAME_EDUCATION_TYPE           object
 14   NAME_FAMILY_STATUS            object
```

- ➔ Just to get the Total Number of Rows and Columns present in the data set, the **‘.shape()’** feature is applied on the Data-frame.

No: of Rows = 307511
No: of Columns = 122

```
In [5]: df.shape
Out[5]: (307511, 122)
```

STEP II – Handling Missing data and Imputing values

➔ In order to **handle missing values,** we first get an estimate of the number of missing values present in each column **(in percentage).**

```
In [83]:  # Finding % of missing values
          100*(df.isnull().sum()/len(df))[(df.isnull().sum()/len(df))*100>0]

Out[83]:  AMT_ANNUITY                    0.003902
          AMT_GOODS_PRICE                0.090403
          NAME_TYPE_SUITE                0.420148
          OWN_CAR_AGE                   65.990810
          OCCUPATION_TYPE               31.345545
                                           ...
          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
          Length: 67, dtype: float64
```

➔ Now that we have enough information on the percentage of missing values in each column of the data set, we **remove columns that poses more than 50% of missing values** using the **',drop()'** feature.

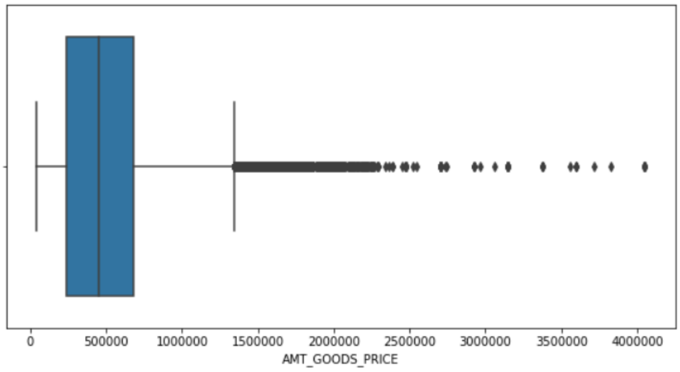
➔ Now, we take into account the **'NAME_TYPE_SUITE'** column (NAME_TYPE_SUITE - Who was accompanying client when he was applying for the loan)which is **Categorical Column** and **impute missing values** in this column.

➔ As mentioned in the previous point, as the column 'NAME_TYPE_SUIT' **is Categorical and not Numerical,** we make use of the **Mode()** function to impute missing values in this column.

➔ Now, we deal with **'AMT_GOODS_PRICE '** column (AMT_GOODS_PRICE - For consumer loans it is the price of the goods for which the loan is given) which is a **Numerical Column.** First, we check for the presence of **outliers** in this Numerical Column by making use of a **Boxplot** (with the use of Seaborn)

```
plt.figure(figsize=[10,5])
sns.boxplot(df.AMT_GOODS_PRICE)

Out[87]: <matplotlib.axes._subplots.AxesSubplot at 0x24888b1fc88>
```



From the Boxplot visualization of the ‘AMT_GOODS_PRICE” column we can see the presence of outliers in this column.

- ➔ We will now **replace** any **NULL** values **with the ‘median’ value** of this Column. We can use ‘median()’ function here as it is a Numerical column.

- ➔ We now take into account the **‘CNT_FAM_MEMBERS’** members column (this column has the count of family members in each of the applicant) and **replace** the **NULL** values **with** a **‘0’ (zero).**

- ➔ After identifying a few columns as in the following, which will be of no use for our Analysis, we remove them from the Data-frame.

'NAME_HOUSING_TYPE' – Hosting Situation of the Client
'CNT_CHILDREN' - Number of children the client has
'REGION_POPULATION_RELATIVE' - Normalized population of region where client lives
'FLAG_EMP_PHONE' - Did client provide work phone
'FLAG_WORK_PHONE' - Did client provide home phone
'FLAG_PHONE' - Did client provide home phone

- ➔ We now **change** the **data types** of the columns (**CNT_FAM_MEMBERS** and **DAYS_REGISTRATION**) from ‘float64’ to ‘int32’ where ‘CNT_FAM_MEMBERS’ has data on how many family members the client has and the ‘**DAYS_REGISTRATION’** column has data on the number of days before the application where the client changed his registration.

- ➔ We are changing their data types as the columns can practically hold only integer values and not float values. For example, the number of family numbers cannot be a float value.

- ➔ Now, in those columns that hold values in ‘Day(s)’,we convert the day(s) into year values. The columns that we will employing here will be;

'DAYS_BIRTH' - Client's age in days at the time of application.
'DAYS_EMPLOYED' - How many days before the application the person started current employment.

- ➔ For the following columns, we are changing the negative values of days to Positive values of days.

'DAYS_REGISTRATION' - How many days before the application did client change his registration.
'DAYS_ID_PUBLISH' - How many days before the application did client change the identity document with which he applied for the loan.

```python
# converting columns given in number of negative days into years like Days_birth
def yrs(x):
    return abs(x*(1/365))

df1['DAYS_BIRTH']=df1['DAYS_BIRTH'].apply(yrs).astype(int)

# converting days employed year with decimal pricesion 1
df1['DAYS_EMPLOYED']=df1['DAYS_EMPLOYED'].apply(yrs).round(1)

# Renaming modified columns
df1.rename(columns={"DAYS_BIRTH": "AGE", "DAYS_EMPLOYED": "YEARS_EMPLOYED"},inplace=True)

# converting negative days of columns 'DAYS_REGISTRATION','DAYS_ID_PUBLISH' into just days
l1=['DAYS_REGISTRATION','DAYS_ID_PUBLISH']
for i in range(0,len(l1)):
    df1[l1[i]]=abs(df1[l1[i]]).astype(int)
```

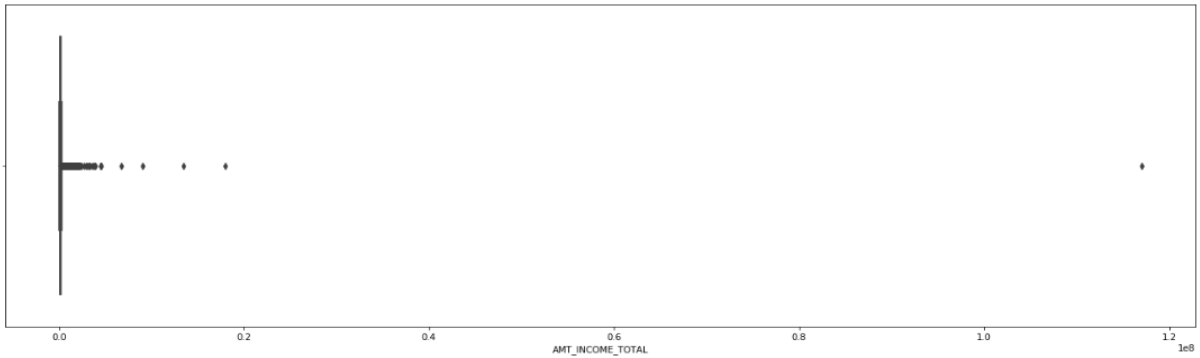
➔ In the above step we also change the names of the “DAYS_BIRTH” AND “DAYS_EMPLOYED” columns to “AGE” and “YEARS_EMPLOYED” respectively.

➔ **OUTLIERS :** We then make use of the ‘describe()’ function to examine the ‘AGE’ column . There are no outliers in AGE column because there is no significant variance from minimum to maximum value as observed with the ‘describe()’ function.

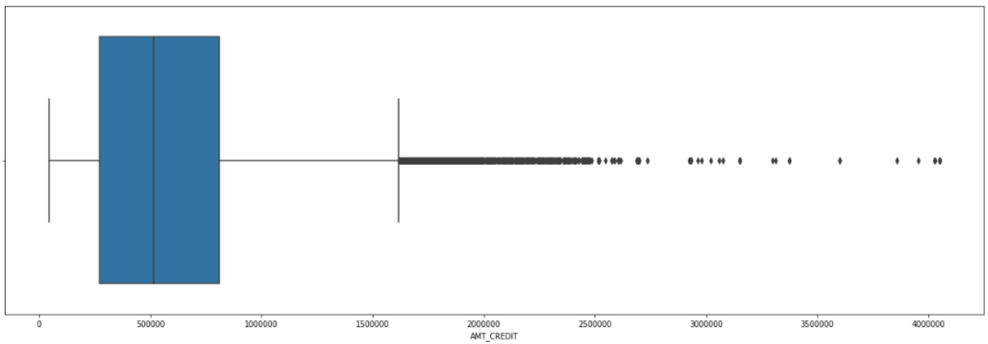
➔ We made use of the quantile function to examine the ‘AMT_GOODS_PRICE’ column and found extreme variance after 75th percent.

➔ We now plot boxplots for each of the columns ‘AMT_INCOME_TOTAL’ , ‘AMT_CREDIT’ and ‘AMT_ANNUITY’ respectively to identify outliers (if any).

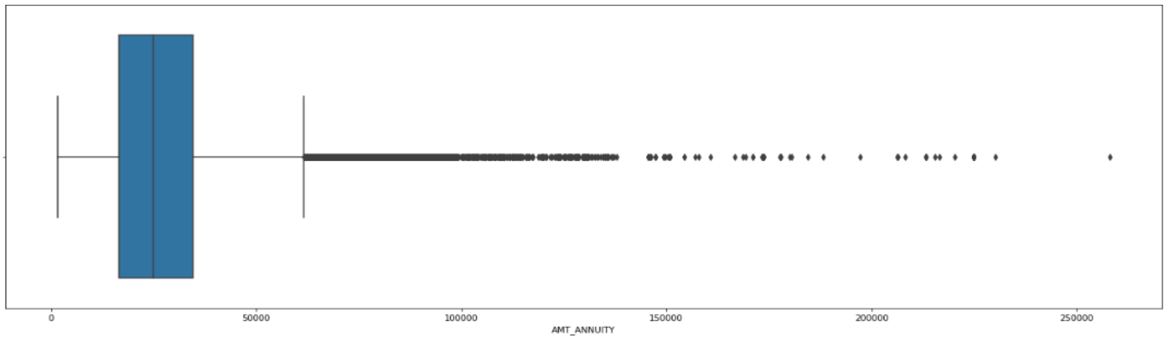
➔ Boxplot for ‘AMT_INCOME_TOTAL’ column :



➔ Boxplot for ‘AMT_CREDIT’ column :



➔ Boxplot for ‘AMT_ANNUITY’ column :



➔ As we can see in the above three Boxplots,
‘AMT_INCOME_TOTAL’ column : Has outliers with very high values such that even the IQR (Interquartile range) is not visible. This indicates the presence of people with very high income.
‘AMT_CREDIT’ column : The credit amount requested by applicants is majorly below 1 million but they are some anomalies above that amount.
‘AMT_ANNUITY’ column : Loan annuity is 50,000 for approximately 75% of data. However there is variance beyond that range.

STEP III – Binning & Correlation Observation

➔ We now bin the columns ‘AMT_INCOME_TOTAL’ and ‘AGE’ and name their binned columns as **‘INCOME_RANGE’ and ‘AGE_RANGE’** respectively.

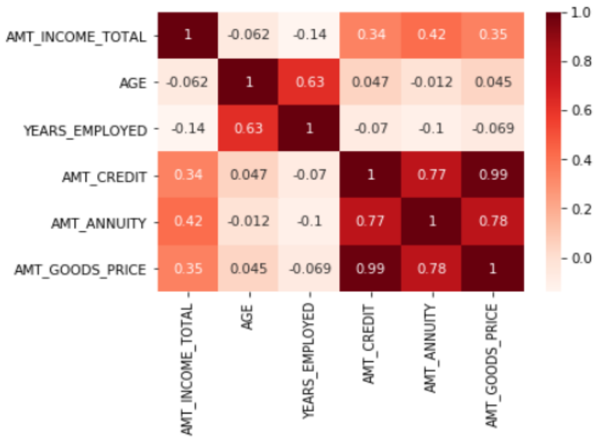
➔ The ‘TARGET’ is now taken into account and we observe the percent of values of 0’s and 1’s and we observe the imbalance percentages.

```
In [19]:  #Imbalance percentage
          100*df1.TARGET.value_counts(normalize = True)

Out[19]:  0    91.927118
          1     8.072882
          Name: TARGET, dtype: float64

In [23]:  # Dividing dataset based on target column
          t0=df1[(df1.TARGET==0)]
          t1=df1[(df1.TARGET==1)]
```

➔ We now plot a correlation map across columns 'AMT_INCOME_TOTAL','AGE','YEARS_EMPLOYED','AMT_CREDIT','AMT_ANNUITY' and 'AMT_GOODS_PRICE' with the ‘TARGET’ as 0.



- We now put up a correlation table across columns 'AMT_INCOME_TOTAL','AGE','YEARS_EMPLOYED','AMT_CREDIT','AMT_ANNUITY' and 'AMT_GOODS_PRICE' with the 'TARGET' as 1.

Out[395]:

	AMT_INCOME_TOTAL	AGE	YEARS_EMPLOYED	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
AMT_INCOME_TOTAL	1.000000	-0.003154	-0.014978	0.038131	0.046421	0.037591
AGE	-0.003154	1.000000	0.582441	0.135070	0.014028	0.135532
YEARS_EMPLOYED	-0.014978	0.582441	1.000000	0.001930	-0.081207	0.006648
AMT_CREDIT	0.038131	0.135070	0.001930	1.000000	0.752195	0.982783
AMT_ANNUITY	0.046421	0.014028	-0.081207	0.752195	1.000000	0.752295
AMT_GOODS_PRICE	0.037591	0.135532	0.006648	0.982783	0.752295	1.000000

- In both the cases where 'TARGET' is 0 and1 respectively, the highest correlation is found in columns 'AMT_CREDIT' , 'AMT_ANNUITY' and 'AMT_GOODS_PRICE' because the loan amount is directly related to loan annuity and goods price.

- We now find the correlation values for the top three highly correlated columns with 'TARGET' 0 and 1 respectively.

'TARGET' as 0 :

Out[389]:

	VAR1	VAR2	Correlation_Value
3	AMT_ANNUITY	AMT_CREDIT	0.771309
6	AMT_GOODS_PRICE	AMT_CREDIT	0.987022
7	AMT_GOODS_PRICE	AMT_ANNUITY	0.776433

'TARGET' as 1 :

Out[396]:

	VAR1	VAR2	Correlation_Value
3	AMT_ANNUITY	AMT_CREDIT	0.752195
6	AMT_GOODS_PRICE	AMT_CREDIT	0.982783
7	AMT_GOODS_PRICE	AMT_ANNUITY	0.752295

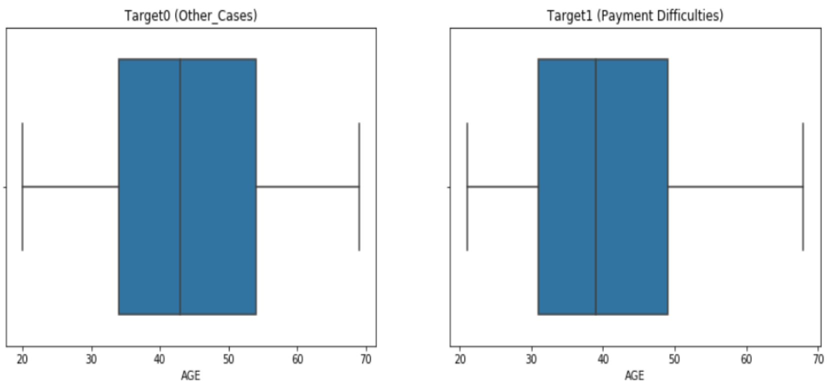
- In the further steps, we will be proceeding with the **Univariate and Bivariate Analysis.**

STEP IV – Univariate and Bivariate Analysis

- **Univariate Analysis :** Numerical Variable and Categorical Variable analysis, **Bivariate Analysis:** Numeric-Numeric , Categorical-Categorical and Numeric-Categorical analysis.

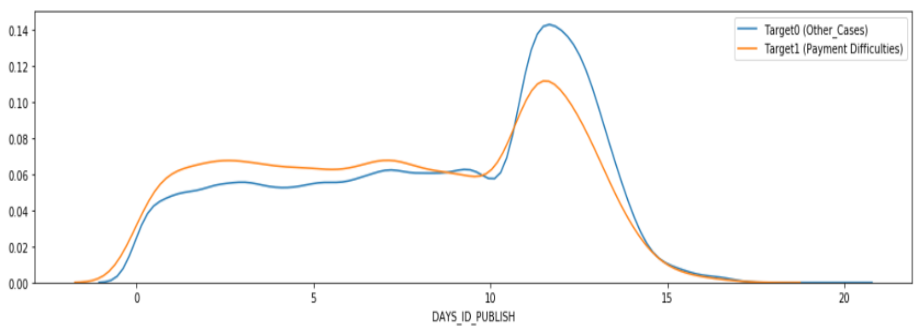
- Lets begin with **Univariate analysis.** We are now performing an Ordered categorical analysis with the 'AGE' column. We can see from the below boxplot that the ages below 40 have more payment difficulties compared to others.

```
Out[649]: <matplotlib.axes._subplots.AxesSubplot at 0x248e6a5a188>
```

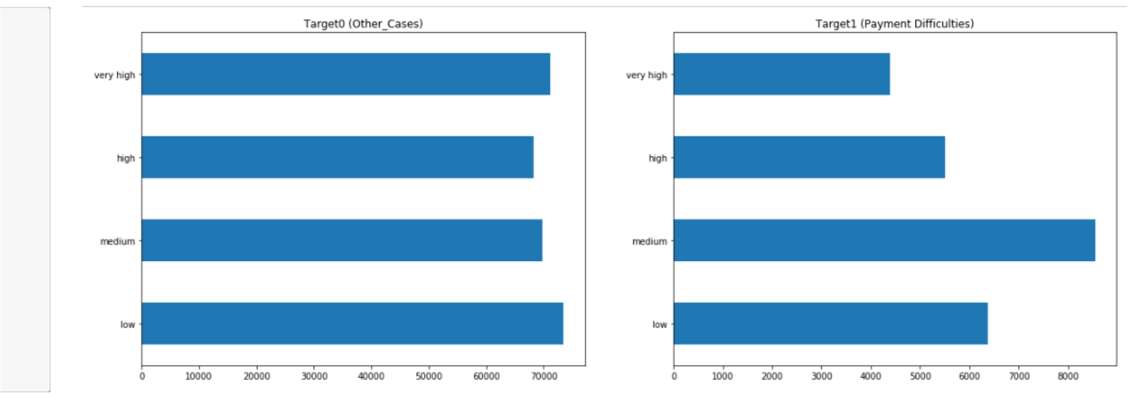


- We are now performing a **Univariate analysis** on the numerical column 'DAYS_ID_PUBLISH'. In the following distplot, we can see that the applicants holding recently published id's have much payment difficulties compared to the older id's.

```
Out[656]: <matplotlib.axes._subplots.AxesSubplot at 0x24922888748>
```

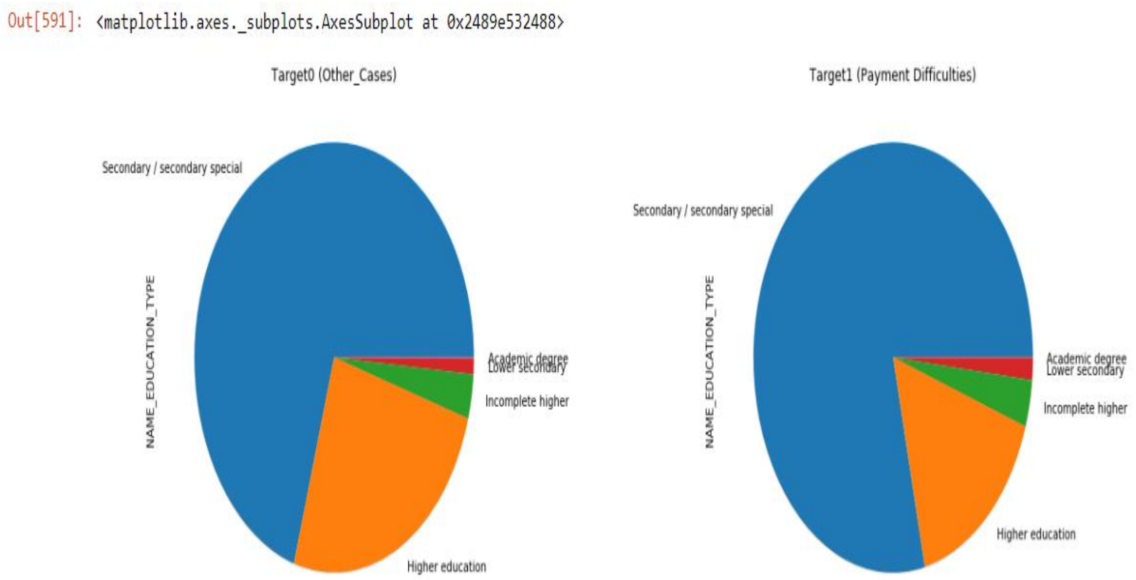


- We are now performing a **univariate** categorical **analysis** on the column 'INCOME_RANGE' .

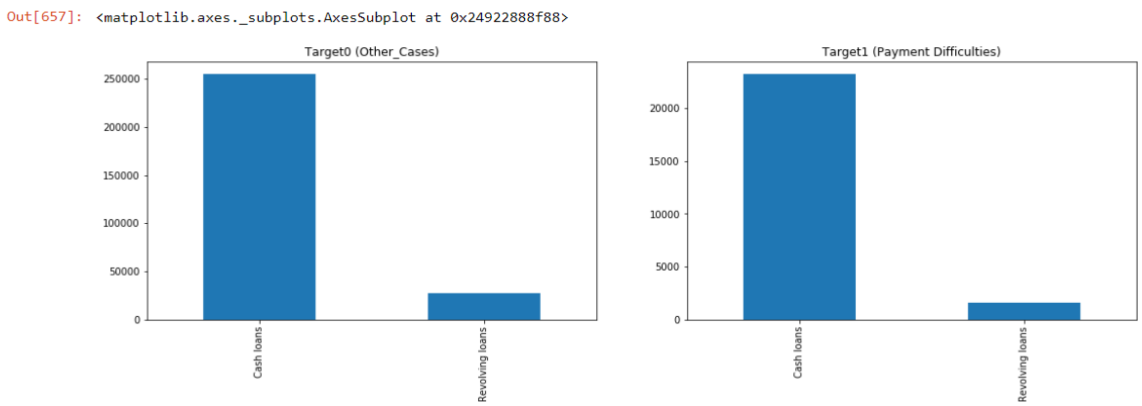


We can see from the above subplot that the loan applicants with a medium salary have more repaying difficulties and the applicants with a very high salary have less payment difficulties.

➔ We are now performing a **univariate** categorical **analysis** on the 'NAME_EDUCATION_TYPE' column. As we can see from the subplot of two pie charts, the Applicants with a high education are high in 'TARGET' value 0 and people with lower secondary are slightly more in 'TARGET' value 1.

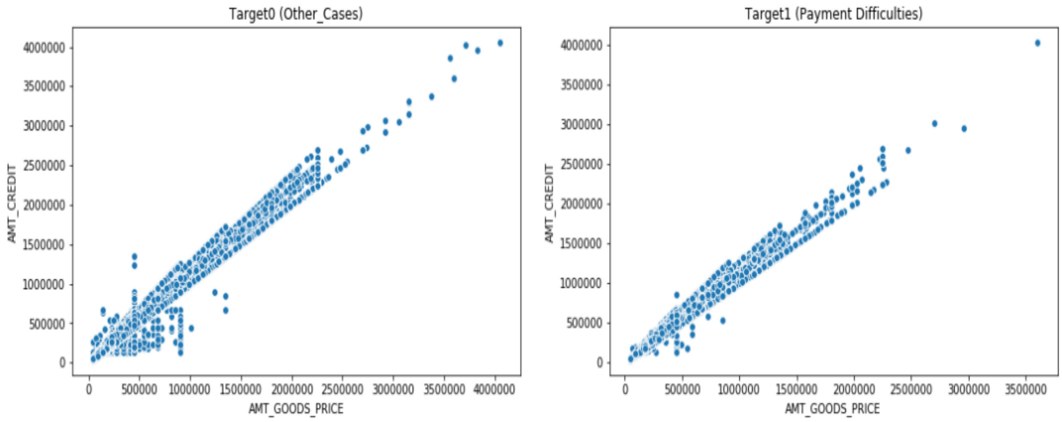


➔ We are now performing a **univariate** categorical **analysis** on the 'NAME_CONTRACT_TYPE' column. We can see that the applicants who requested for revolving loans have less payment difficulties.



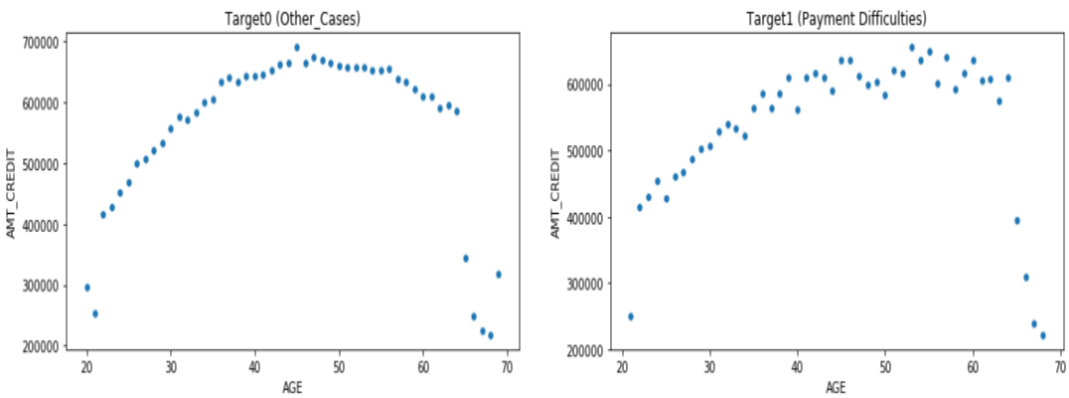
➔ Lets now move on to **Bivariate analysis.** We are now performing a Bivariate Numeric-Numeric analysis on the columns 'AMT_GOODS_PRICE' and 'AMT_CREDIT'. We can see a clear correlation between 'AMT_GOODS_PRICE' and 'AMT_CREDIT' in both the cases and the applicants got loan as per goods price.

Out[672]: <matplotlib.axes._subplots.AxesSubplot at 0x2492c0eb108>



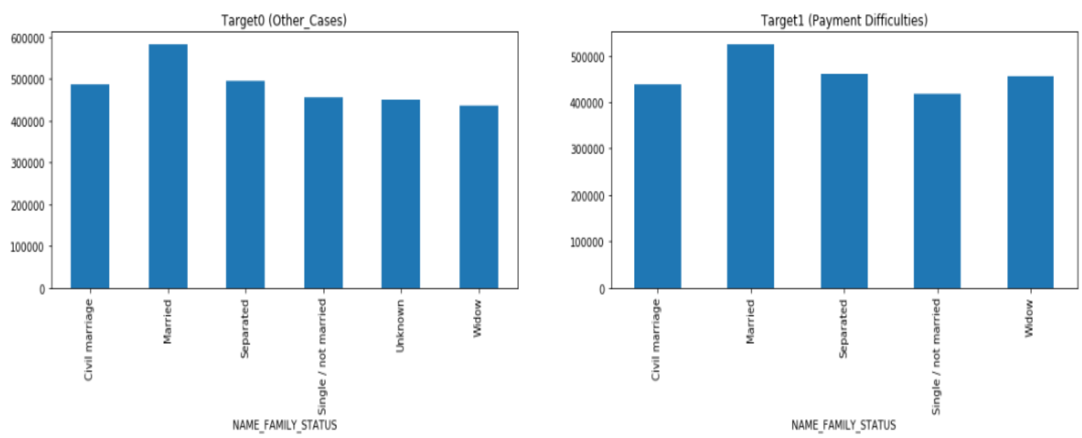
➔ We are now performing a **Bivariate** Numeric-Numeric **analysis** on the columns 'AMT_CREDIT' and 'AGE'. As we can see from the subplot, for the Ages of over 20, loan credit limit is linearly increasing upto 60 and then it's decreasing in a 'TARGET' value of 0 and in a 'TARGET' value of 1 there is variance in amount.

Out[705]: <matplotlib.axes._subplots.AxesSubplot at 0x24920fab1c8>



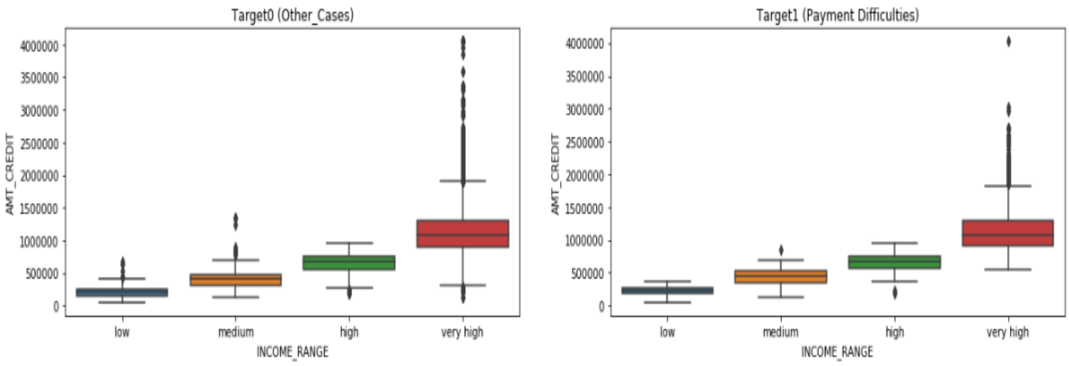
➔ We are now performing a **Bivariate** Numerical-Categorical **analysis** on the columns 'NAME_FAMILY_STATUS' and 'AMT_GOODS_PRICE'. As we can see from the subplot, married people have applied for a loan with a higher value.

Out[764]: <matplotlib.axes._subplots.AxesSubplot at 0x24939991088>



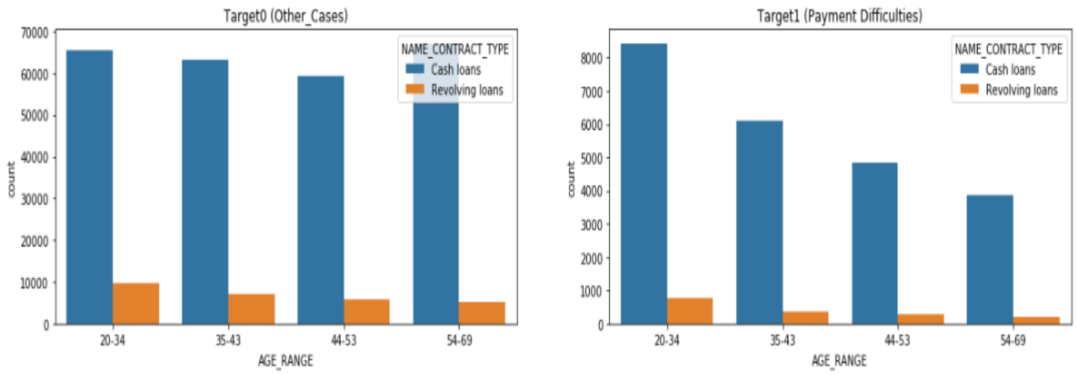
➔ We are now performing a **Bivariate** Numerical-Categorical **analysis** on the columns 'INCOME_RANGE' and 'AMT_CREDIT'. From the sublots, we can see that the average loan credit is based on the applicant's income.

```
Out[156]: <matplotlib.axes._subplots.AxesSubplot at 0x2f7eb115788>
```



➔ We are now performing a **Bivariate** Categorical-Categorical **analysis** on the columns 'AGE_RANGE' and 'NAME_CONTRACT_TYPE'. We can see that the payment difficulties in revolving loans is less than the cash loans.

```
Out[761]: <matplotlib.axes._subplots.AxesSubplot at 0x249392cae48>
```



STEP V – Loading and Merging Datasets

➔ The dataset 'previous_application.csv' which is a CSV file is now **loaded** and is assigned to a variable 'df2'.

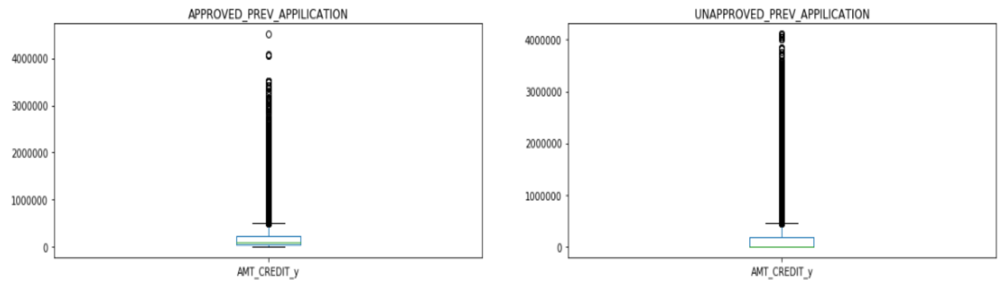
➔ We are now **taking only the first 21 columns** and **removing the unwanted columns** (ie) the columns that are of no use to our analysis. The removed columns are 'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START', 'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY', 'RATE_DOWN_PAYMENT' and 'DAYS_DECISION'.

➔ We also identify the columns that have more than 50% of null values and remove them from the data-frame.

➔ Now we will be the **Merging** the datasets.

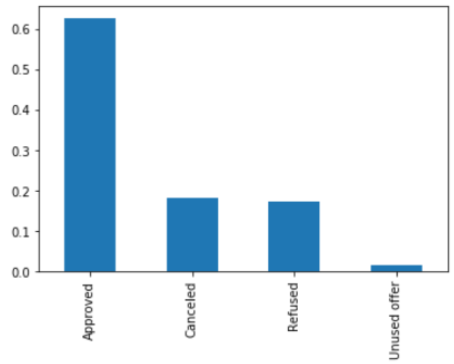
- We will be assigning the merged dataset into a variable ‘df3’.
- We will now be performing a **Univariate Analysis** on the Numerical column ‘AMT_CREDIT_y’. As we can see, the Loan credit of previous application with ‘unapproved’ has more extreme values than ‘approved’ applications. Thus applications with a higher loan amount is not easily approved.

```
Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x2f7d4983408>
```



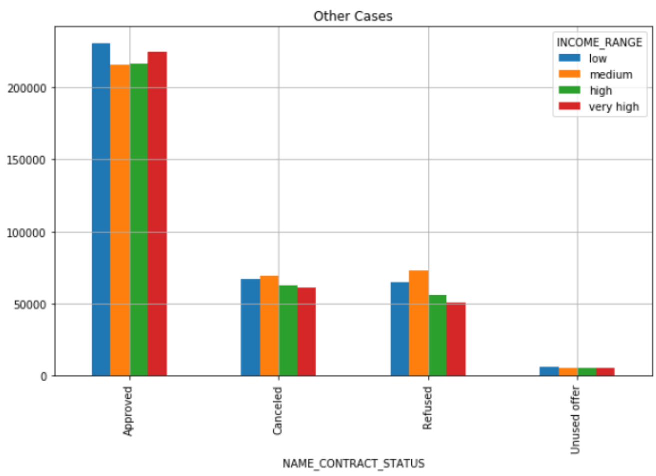
- We will now be performing a **Univariate Analysis** on the Numerical column ‘NAME_CONTRACT_STATUS’. As we can see the ‘Approval rate’ is high compared to the rest.

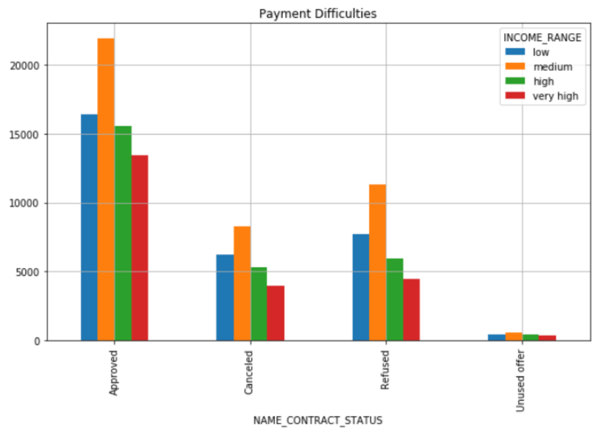
```
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x2f7d6400908>
```



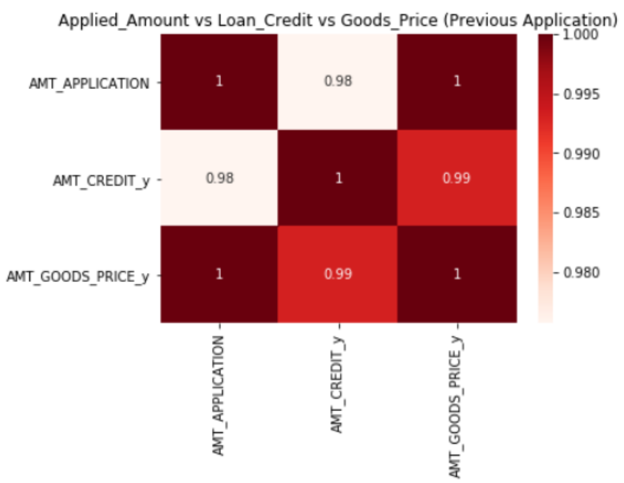
- We will now be performing a **Bivariate Analysis** on the Numerical-Catagorical columns ‘NAME_CONTRACT_STATUS’ and ‘INCOME_RANGE’. As we can see from the subplot,there is a high approval rating for low income applicants. The medium income applicants have difficulty in repayment.

```
Out[157]: <matplotlib.axes._subplots.AxesSubplot at 0x2f82eb2f108>
```



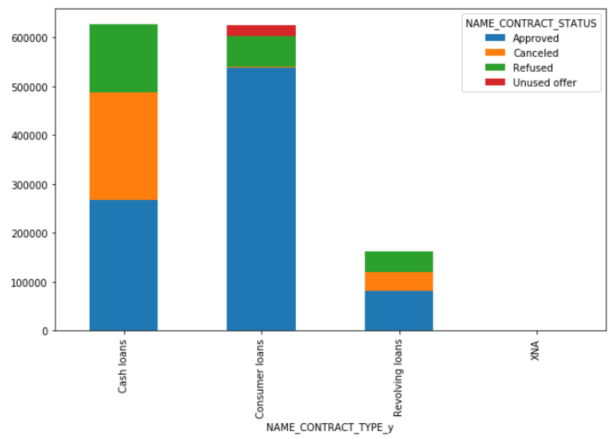


➔ We will now be **plotting a correlation map** across the columns 'AMT_APPLICATION', 'AMT_CREDIT_y' and 'AMT_GOODS_PRICE_y'. We can see that there is a high correlation between Applied_Amount, Loan_Credit and Goods_Price as goods price is inter-related to applied amount and loan credit is issued for applied amount.



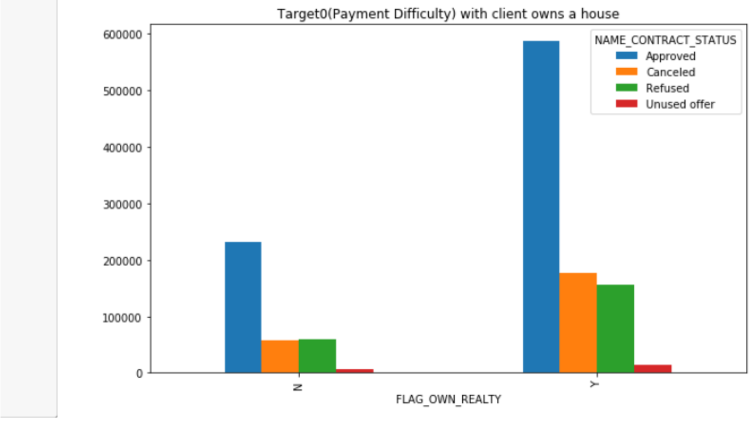
➔ We will now be performing a **Bivariate Analysis** on the Catagorical-Catagorical columns 'NAME_CONTRACT_TYPE_y' and 'NAME_CONTRACT_STATUS' .As we can see there is a higher amount of cancellations and rejection in cash loans with a few cancellations and rejection in other 2 categories and with a high approval rates for consumer loans.

```
Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x2f7d3fd8a88>
```



➔ We will now be performing a **Bivariate Analysis** on the Catagorical-Catagorical columns ‘FLAG_OWN_REALTY’ and ‘NAME_CONTRACT_STATUS’ . As we can see from the plot,the applicants owning a house have a better approval rate, but have difficulty in repaying the loan.

```
Out[171]: <matplotlib.axes._subplots.AxesSubplot at 0x2f7f84eb048>
```



CONCLUSION

Application Dataset

➔ There are only 8% of applicants with payment difficulties.
➔ Ages below 40 have difficulty in loan repayment.
➔ Married people applied for a loan to purchase high valued goods.
➔ Loan credit is given based on the income of an applicant.

Merged Dataset

➔ Applications for huge sums of loan are not approved in most cases.
➔ Requested amount to purchase goods is provided via loan credit in both the cases.
➔ Applicants with medium salary have more repaying capabilities in both the datasets.
➔ Most of the loan applications are approved.
➔ Cash loans have low approval rates and consumer loans have high approval rates.
➔ Even though applicants who own a house have a high approval rate, they have difficulty in loan repayment.

Recommendations inorder to avoid repayment risks

➔ Loan repayment capability checks have to be considered for ages below 40.
➔ More checks have to be done while lending loans to applicants with medium income.
➔ Better to avoid lending loans to applicants who own a house and have payment difficulties.