EDA Case Study

Loan Application Assessment

Project Team: *Shivram J & Pappu Dindayal Kapgate*

Introduction:

As part of this case study exercise we were given two datasets of a bank namely

1. application_data.csv
2. previous_application_data.csv

The objective for our analysis was to understand this dataset, clean and identify patterns and behaviors which can help determine risk in banking sector while lending money to customers. Key factors that needed to be considered were

1. Lending money to customers that are incapable of paying back
2. Not lending money to customer who can pay back resulting in revenue loss.

Processing Current Application:

We started looking at the application data csv file and explored data inconsistencies and null value columns.

1. Exploring the Data in Application Data file

```
[ ]: appl_data.head()
```

```
[4]: #Checking number of rows and columns in the file

     appl_data.shape
```

```
[4]: (307511, 122)
```

```
[5]: #Check the column data size and data types
     appl_data.info()

     #This is a lot of data!!!
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

```
[6]: #checking statistical data

     appl_data.describe()
```

**We have cleared output to save space*

Data cleansing:

1. Before analyzing the data we identified percentage of null values in data columns.

```
[10]: # Identify the % of missing values and list greater than 25% null value columns

      null_cols=((appl_data.isnull().sum()*100)/appl_data.shape[0]).round(2)
      null_cols

[10]: SK_ID_CURR                    0.0
      TARGET                        0.0
      NAME_CONTRACT_TYPE            0.0
      CODE_GENDER                   0.0
      FLAG_OWN_CAR                  0.0
                                   ...
      AMT_REQ_CREDIT_BUREAU_DAY    13.5
      AMT_REQ_CREDIT_BUREAU_WEEK   13.5
      AMT_REQ_CREDIT_BUREAU_MON    13.5
      AMT_REQ_CREDIT_BUREAU_QRT    13.5
      AMT_REQ_CREDIT_BUREAU_YEAR   13.5
      Length: 122, dtype: float64
```

2. As there were a lot of columns having higher percentage of missing values. We decided to drop any column which has more than 25% of null values.

```
[9]: #Drop the columns
     appl_data.drop(appl_data.loc[:,appl_data.isnull().mean()>=.25],axis=1,inplace=True)

[10]: #Once dropped the total number of columns are 72
      appl_data.shape

[10]: (307511, 72)
```

3. Impute missing values with mean, median, mode, 0 where applicable.

Columns Name	Impute Strategy
AMT_GOODS_PRICE	Mean
NAME_TYPE_SUITE	Mode
CNT_FAM_MEMBERS	0
OBS_60_CNT_SOCIAL_CIRCLE	Mean
DEF_60_CNT_SOCIAL_CIRCLE	Mean
OBS_30_CNT_SOCIAL_CIRCLE	Mean
DEF_30_CNT_SOCIAL_CIRCLE	Mean
AMT_REQ_CREDIT_BUREAU_HOUR	0
AMT_REQ_CREDIT_BUREAU_DAY	0

AMT_REQ_CREDIT_BUREAU_WEEK	0
AMT_REQ_CREDIT_BUREAU_MON	0
AMT_REQ_CREDIT_BUREAU_QRT	0
AMT_REQ_CREDIT_BUREAU_YEAR	Median
AMT_ANNUITY	Mean
DAYS_LAST_PHONE_CHANGE	0

4. Checking and converting data type to appropriate type based on columns data.

   a. **Object**

```
[55]: #Check if any Objects need to be converted. Looks good!
      appl_data.select_dtypes('object')
```

   b. **Float:** We converted the below columns from float to integers as it had only whole numbers. Counts and days are whole numbers in real life, so converting them into integers.

```
[56]: #Check if any float columns need to be converted. Looks like some can be converted into integers!
      appl_data.select_dtypes('float')
```

Columns Name	Old Data type	New Data Type
CNT_FAM_MEMBERS	Float64	Int64
DAYS_LAST_PHONE_CHANGE	Float64	Int64
DAYS_REGISTRATION	Float64	Int64
AMT_REQ_CREDIT_BUREAU_HOUR	Float64	Int64
AMT_REQ_CREDIT_BUREAU_DAY	Float64	Int64
AMT_REQ_CREDIT_BUREAU_WEEK	Float64	Int64
AMT_REQ_CREDIT_BUREAU_MON	Float64	Int64
AMT_REQ_CREDIT_BUREAU_QRT	Float64	Int64
AMT_REQ_CREDIT_BUREAU_YEAR	Float64	Int64

5. Converting all flag variables values [1,0] to a Binary variable form (Y or N)

[67]:
```
#converting all flag columns to Y or N instead of 1 and 0

appl_data[flag_columns]=appl_data[flag_columns].replace((0, 1), ('N', 'Y'))
```

[68]:
```
#All the flag have been converted to Y and N

appl_data[list(appl_data.filter(regex='FLAG'))]
```

[68]:

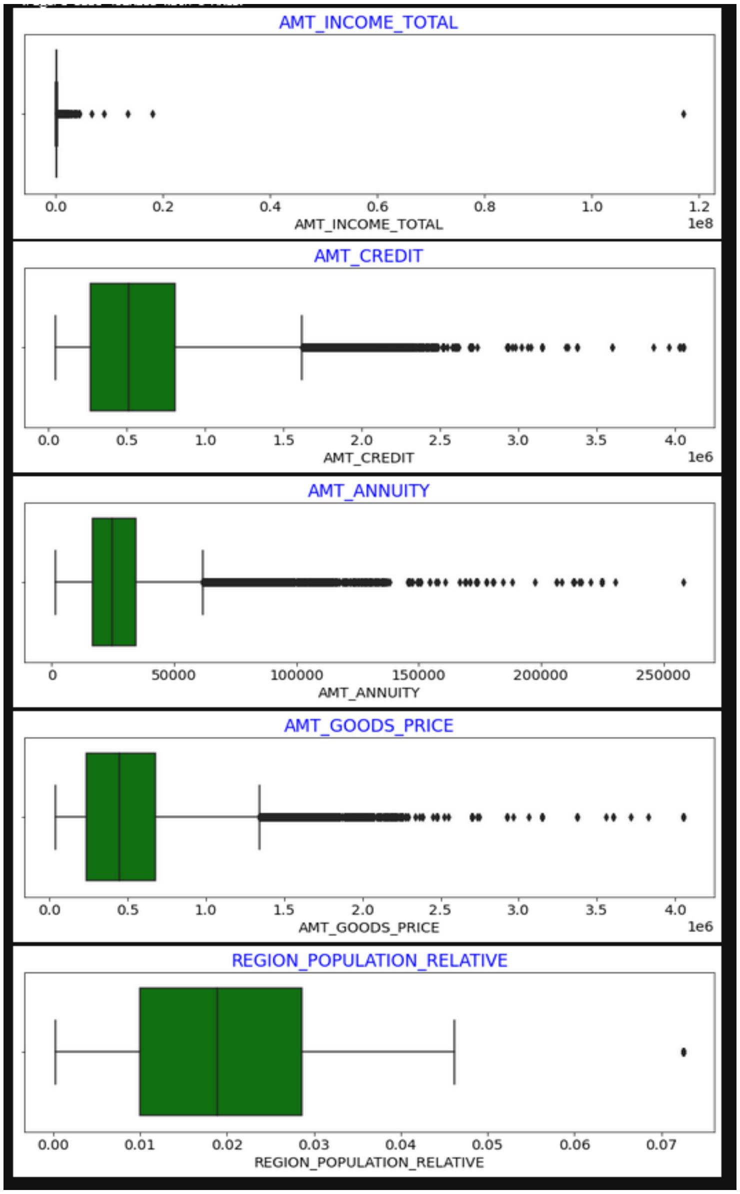
	FLAG_OWN_CAR	FLAG_OWN_REALTY	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE	FLAG_CONT_MOBILE
0	N	Y	Y	Y	N	Y
1	N	N	Y	Y	N	Y
2	Y	Y	Y	Y	Y	Y
3	N	Y	Y	Y	N	Y
4	N	Y	Y	Y	N	Y
...	...	...	...	...	...	...
307506	N	N	Y	Y	N	Y
307507	N	Y	Y	N	N	Y

Handling Outliers:

We identified outliers for both floating and integer columns.

Float Outliers:

We can see the outliers depicted in the boxplot below are far away from the 75% quantile.

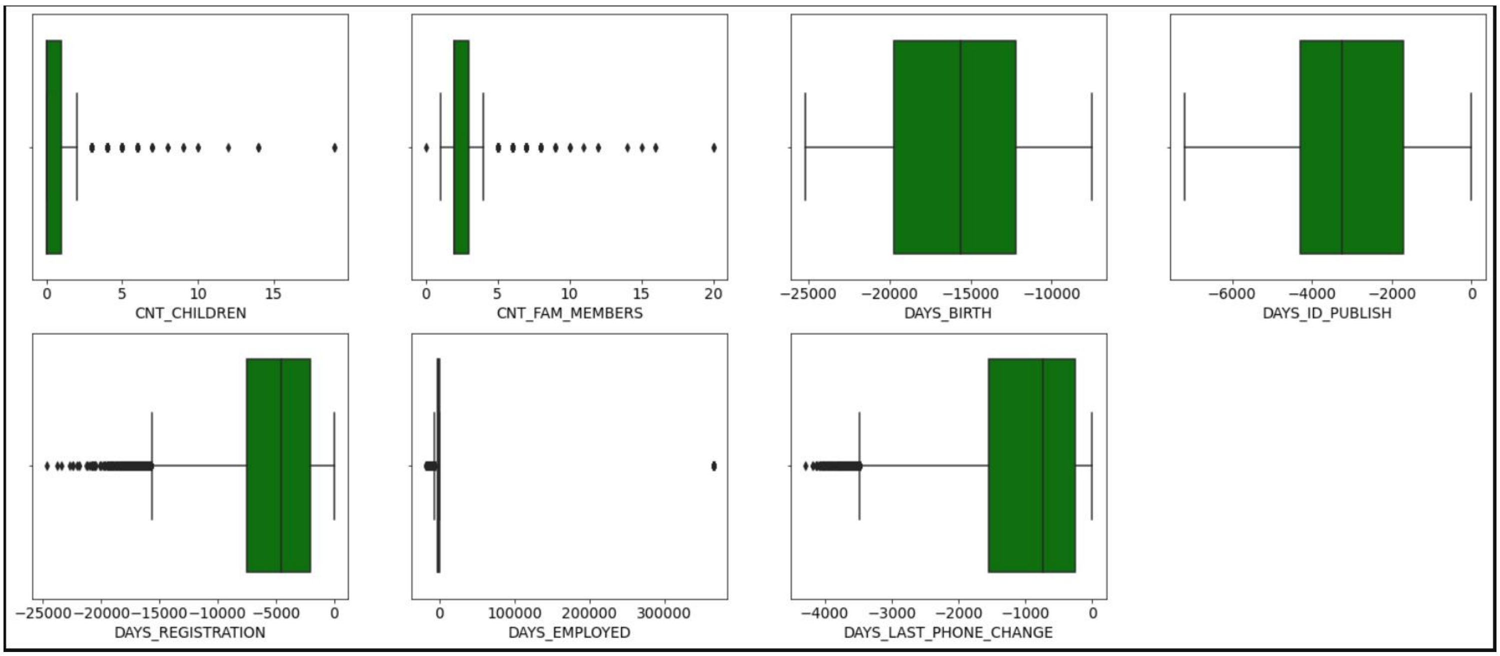


By looking at the data columns individually we decided on what percentile of data should be retained in the dataset. Which can make analysis accurate and does not result in skewed patterns. For determining the percentile (%) we looked at max values and retained only those outliers which are relatively closer to the 75% quantile values. We made sure all high value anomalies are completed removed.

Following is the list of columns and the corresponding quantiles below which we have retain the data.

Columns Name	Retain % data
AMT_INCOME_TOTAL	95%
AMT_CREDIT	99%
AMT_ANNUITY	99%
AMT_GOODS_PRICE	90%

Integer outliers



Along with outliers we observed that there were a huge population of negative values. Just like float attributes we handled the outliers by looking at the data and figuring out the anomalies.

Columns Name	Retain % data
CNT_CHILDREN	99%
CNT_FAM_MEMBERS	99%
DAYS_REGISTRATION	90%
DAYS_EMPLOYED	75%
DAYS_LAST_PHONE_CHANGE	90%

Derived Columns

We organized the applicable data columns into ranges and summarized it for effective analysis.

- **Converting DAYS into YEARS**

```
# Converting days into years

appl_data["YEARS_BIRTH"]= appl_data.DAYS_BIRTH.apply(lambda x: x/365)
appl_data["YEARS_ID_PUBLISH"]= appl_data.DAYS_ID_PUBLISH.apply(lambda x: x/365)
appl_data["YEARS_REGISTRATION"]= appl_data.DAYS_REGISTRATION.apply(lambda x: x/365)
appl_data["YEARS_LAST_PHONE_CHANGE"]= appl_data.DAYS_LAST_PHONE_CHANGE.apply(lambda x: x/365)
```

- **Categorical Variables**
  - **INCOME_RANGE**

A) AMT_INCOME

```
8]: #creating categories for customer incomes.

    label = ['Very Low', 'Low', 'Moderate', 'High', 'Very High']
    appl_data["AMT_INCOME_CATEG"]= pd.qcut(appl_data.AMT_INCOME_TOTAL,q=[0, .2, .4, .6, .8, 1],labels=label)
    appl_data.AMT_INCOME_CATEG.value_counts()

8]: Low          48663
    Very Low     31909
    High         28609
    Very High    27430
    Moderate     18632
    Name: AMT_INCOME_CATEG, dtype: int64
```

  - **AGE_GROUP RANGE**

```
[109]: # Creating age groups of customer
       bins = [18, 30, 40, 50, 60, 70, 120]
       age_group_labels = ['18-29', '30-39', '40-49', '50-59', '60-69', '70+']
       appl_data['AGE_GROUP_CUSTOMER'] = pd.cut(appl_data.AGE_CUSTOMER,bins,labels=age_group_labels,include_lowest=True)
       appl_data['AGE_GROUP_CUSTOMER'].value_counts()

[109]: 30-39    57167
       18-29    39999
       40-49    38551
       50-59    17733
       60-69     1793
       70+          0
       Name: AGE_GROUP_CUSTOMER, dtype: int64
```

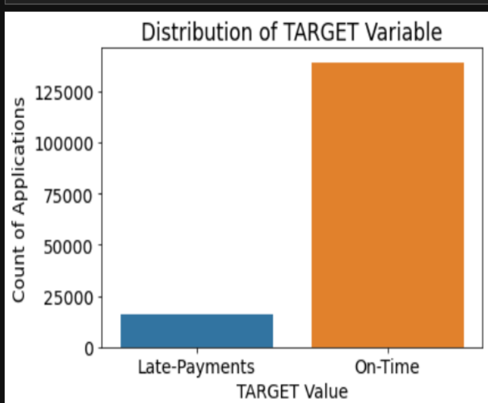
Performing Data Analysis and Identifying Insights

1. Categorical unordered univariate analysis

a. TARGET Variable

We created a new column and divided the dataset into on-time and late paying customers and assigned this category to paystatus column.

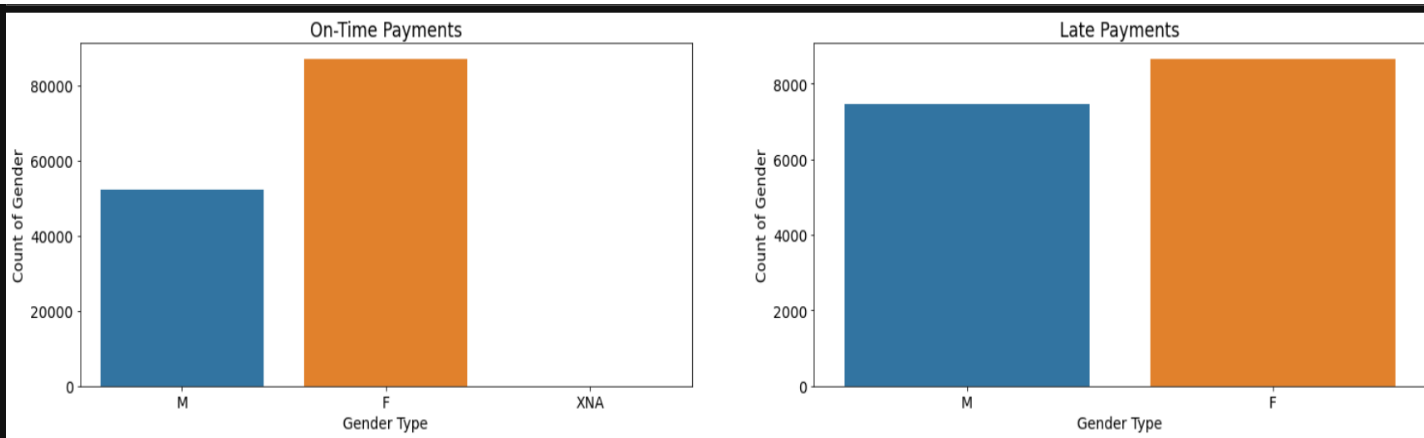
```
[112]: sns.countplot(appl_data.paystatus)
plt.rcParams.update({'font.size': 14})
plt.xlabel("TARGET Value")
plt.ylabel("Count of Applications")
plt.title("Distribution of TARGET Variable")
plt.show()
```



Observation: In the given population, the number of people having difficulties making payments is less than other applicants.The data been leaning towards ontime payments which may be due to random sampling.

*Late payments are < 25K and others are >125K

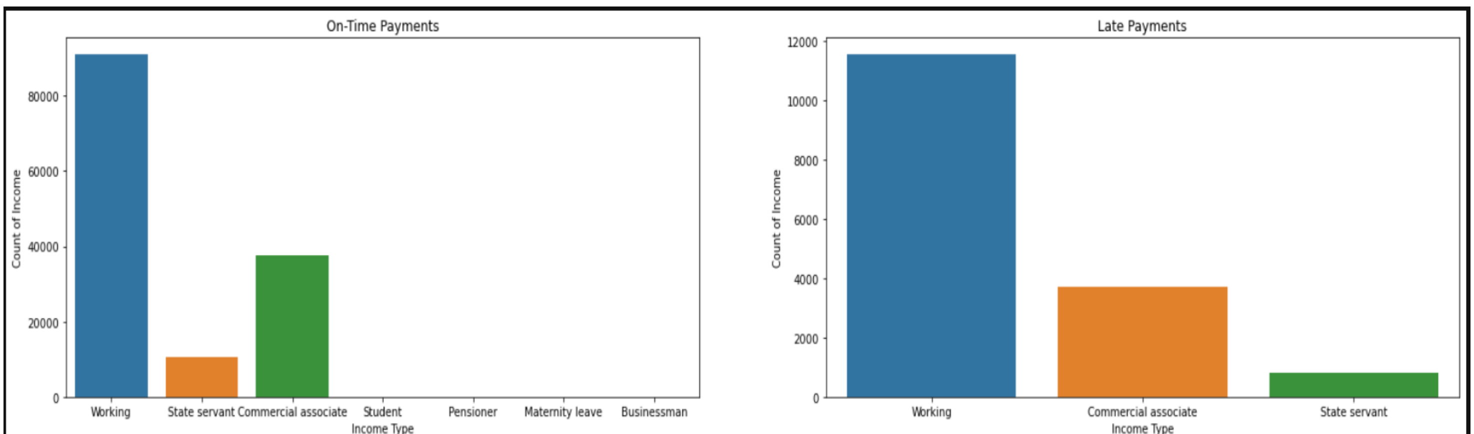
b. GENDER Variable



Observation: Females make more on-time payments than Male counterparts. But for late payments both genders are considerably similar.

*Ontime payments - Males are 50-60K, Females are > 80K
Late payments - Males are 7-8K, Female are >8K.

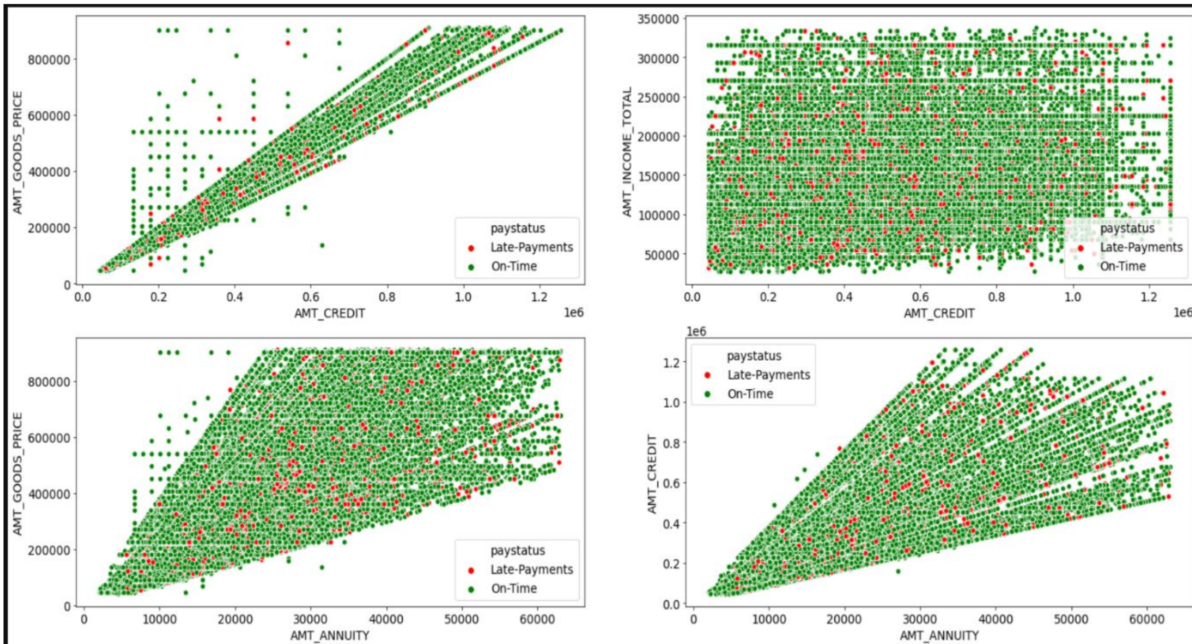
c. INCOME TYPE Variable



Observation: Student,Pensioner,Maternity Leave and Businessman are not present in the late payments data group. Also On-time paying customers are higher than late paying customers accross all income type groups.

Bivariate Analysis

a. Numeric Variable Analysis

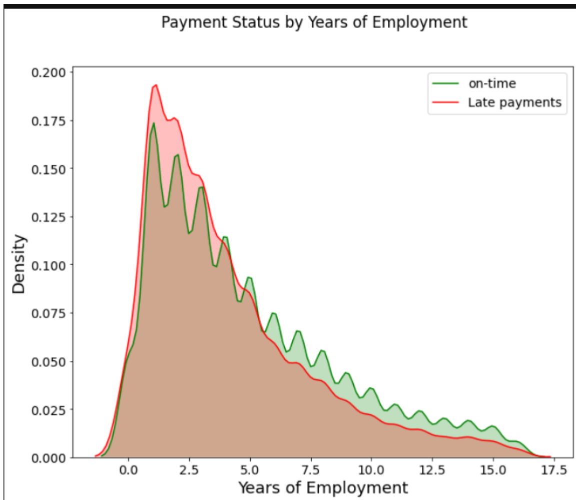


Observation:

1. We can see a linear progression between credit amount,annuity amount and goods price. Even customers with payment diffculties have linearly progressed accross all the above plot.

2. We didnt find strong linear corelation between income and credit amount. We were expecting to see people with higher income get more credit.

b. Years of Employment

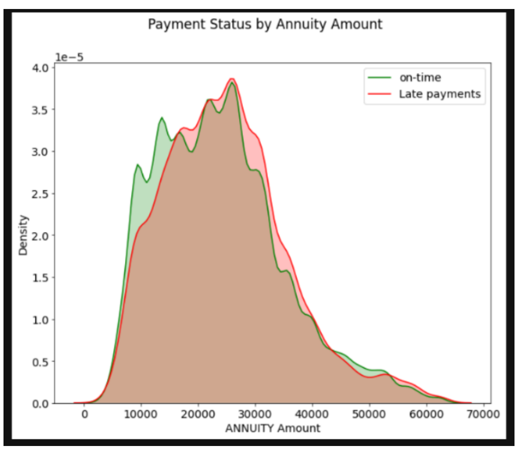


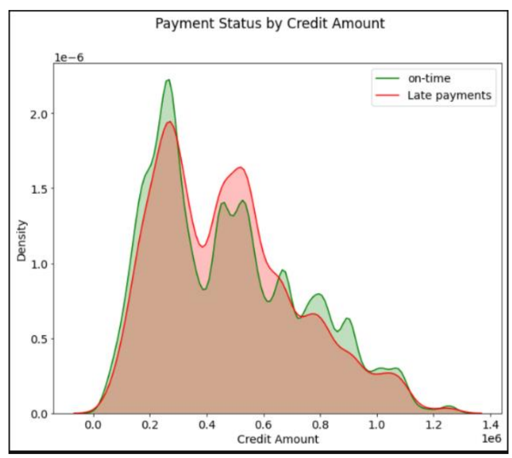
Observation:

1. We observe that customers with less than 5 years of employment have a higher possibility to default on their loans.

2. Customers with greater than 5 years of employment have less difficulties in paying their loans, hence they are a better group to attract on loan products.

c. ANNUITY Amount and CREDIT Amount





Observation:

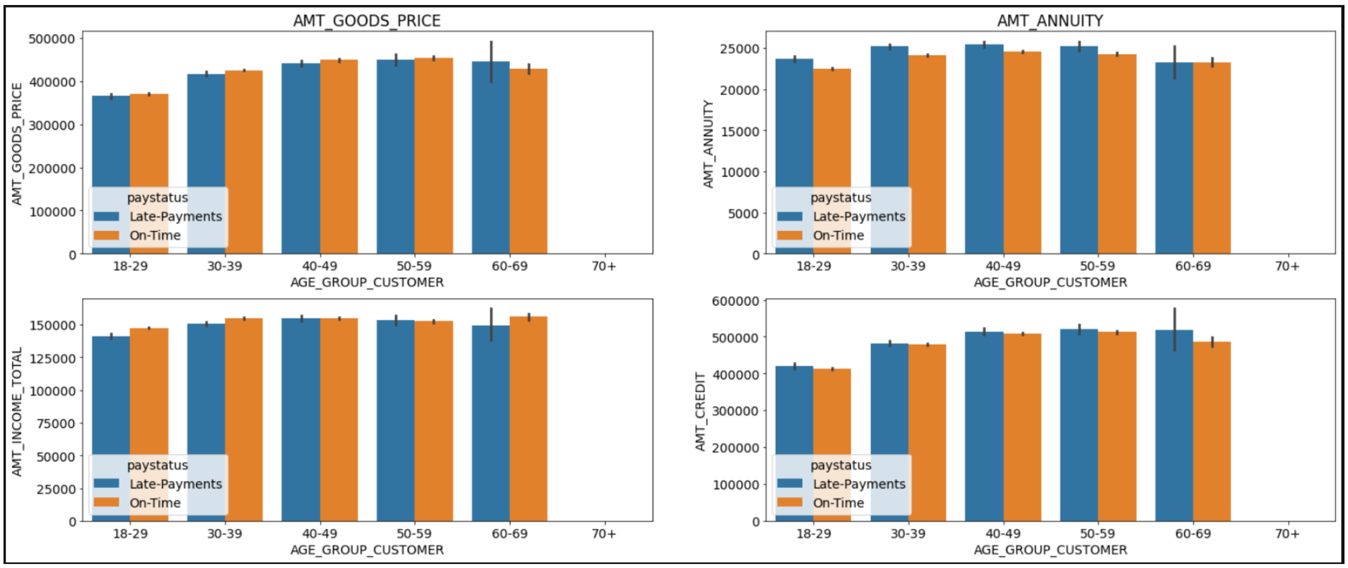
1. We observe that annuity amount below 20,000 have higher on-time payments rates.

2. Customer with loan credits of 2.5 lakhs have a higher on-time payment than rest of the credit ranges.

So we can conclude there is a low risk in giving out loans for 2.5 Lakhs or repayment annuity of below 20K.

Analysis between Numeric and Categorical variables

AGE_GROUP_CUSTOMER



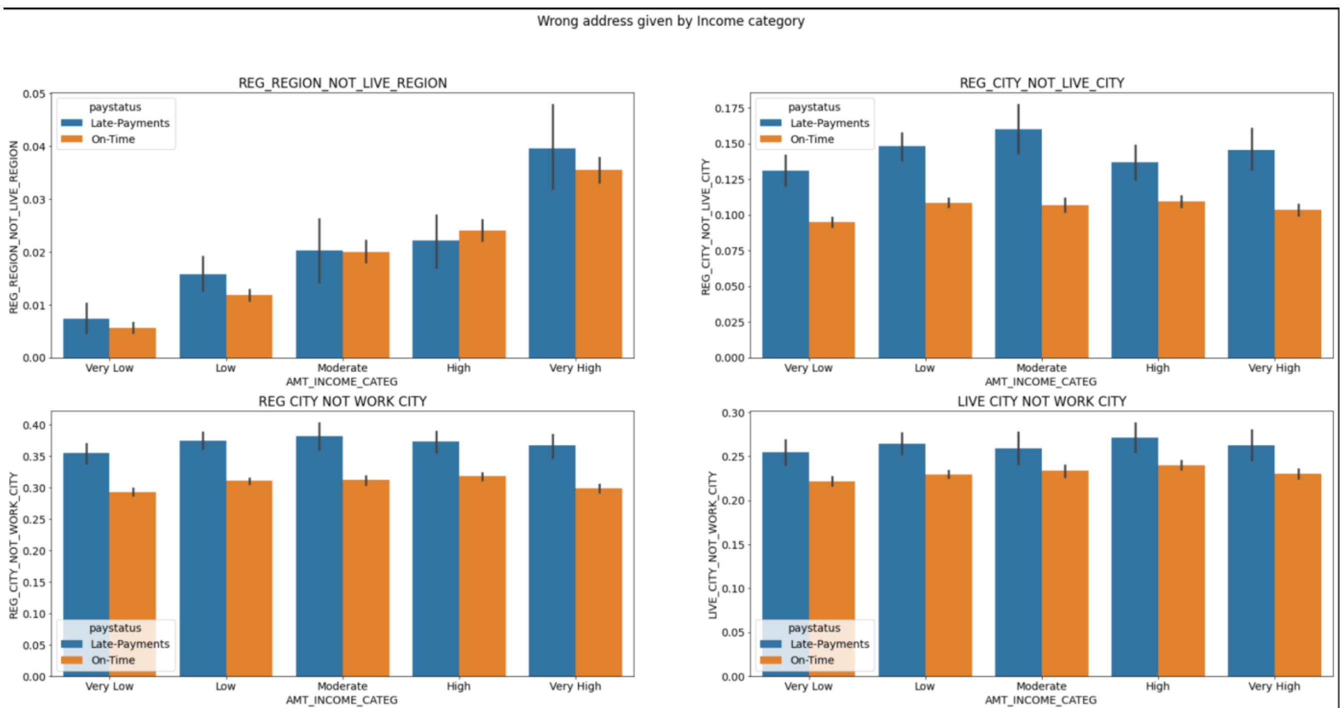
Observation:

1. We observe within age group 60-69 if their total income is more than 1.25 Lakhs and credited amount is more than 5 Lakhs, then there may be some difficulty to repay. So, we can say that if we offer those customers loans below 5 Lakhs then there is a higher probability to pay installments on time.

2. While considering annuity in range 20-25 thousand, all age groups apart from 60-69 have a higher possibility to default.

3. So we can conclude there is a low risk in giving out loans to age group 60-69 if the annuity is around 20-25 thousand and income is more than 1.25 Lakhs and credited amount is below 5 lakhs.

Wrong Address provided by Customers

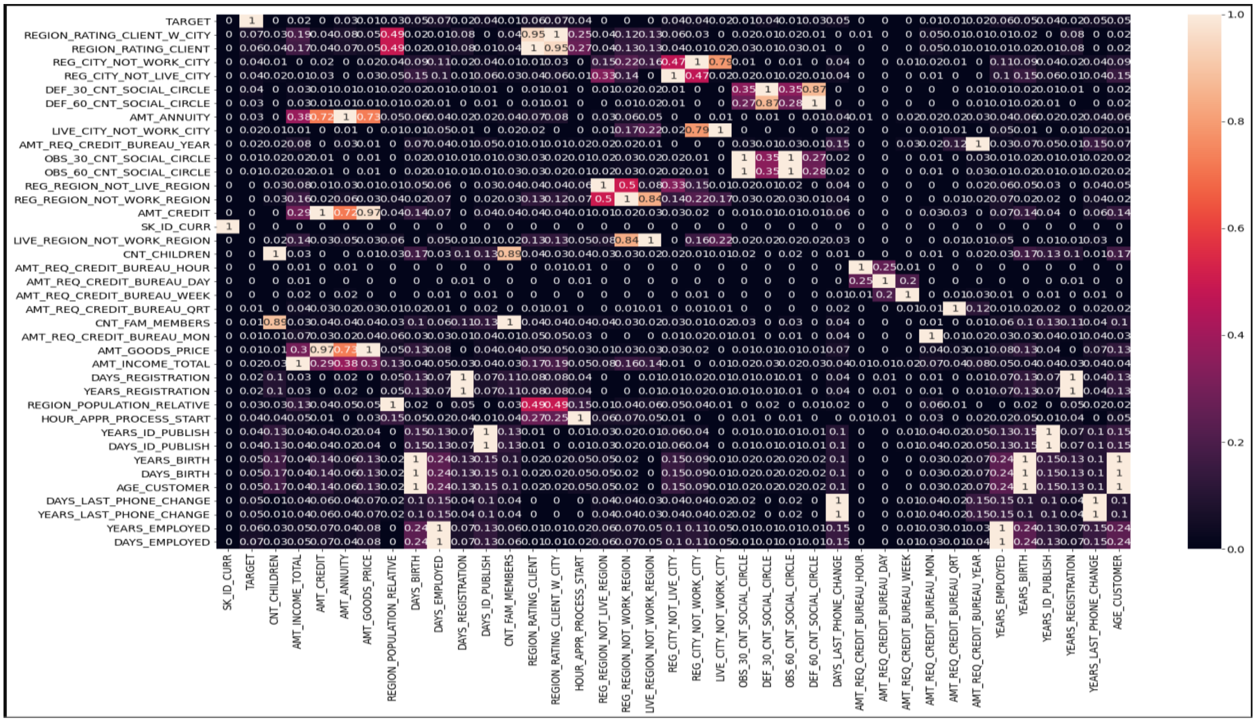


Observation:

We observed that no matter what income category the customer belongs to, if they provide wrong contact address then they have a higher probability to default.

As all the bars indicate higher late payment values when the address is not provided.

Multivariate Analysis



Based on the above heatmap, we have identified the highly corelated attributes:

```
[351]: AMT_GOODS_PRICE              AMT_CREDIT                   0.97
       AMT_CREDIT                   AMT_GOODS_PRICE              0.97
       REGION_RATING_CLIENT_W_CITY  REGION_RATING_CLIENT         0.95
       REGION_RATING_CLIENT         REGION_RATING_CLIENT_W_CITY  0.95
       CNT_CHILDREN                 CNT_FAM_MEMBERS              0.89
       CNT_FAM_MEMBERS              CNT_CHILDREN                 0.89
       DEF_60_CNT_SOCIAL_CIRCLE     DEF_30_CNT_SOCIAL_CIRCLE     0.87
       DEF_30_CNT_SOCIAL_CIRCLE     DEF_60_CNT_SOCIAL_CIRCLE     0.87
       REG_REGION_NOT_WORK_REGION   LIVE_REGION_NOT_WORK_REGION  0.84
       LIVE_REGION_NOT_WORK_REGION  REG_REGION_NOT_WORK_REGION   0.84
       LIVE_CITY_NOT_WORK_CITY      REG_CITY_NOT_WORK_CITY       0.79
       REG_CITY_NOT_WORK_CITY       LIVE_CITY_NOT_WORK_CITY      0.79
       AMT_GOODS_PRICE              AMT_ANNUITY                  0.73
       AMT_ANNUITY                  AMT_GOODS_PRICE              0.73
       AMT_CREDIT                   AMT_ANNUITY                  0.72
       AMT_ANNUITY                  AMT_CREDIT                   0.72
       dtype: float64
```

Processing Previous Application:

Data Exploration

```
[135]: p_appl_data.head()
```

[135]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AM
0	2030495	271877	Consumer loans	1730.430	
1	2802425	108129	Cash loans	25188.615	
2	2523466	122040	Cash loans	15060.735	
3	2819243	176158	Cash loans	47041.335	
4	1784265	202054	Cash loans	31924.395	

5 rows × 37 columns

```
[136]: p_appl_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_PREV                   1670214 non-null  int64
 1   SK_ID_CURR                   1670214 non-null  int64
 2   NAME_CONTRACT_TYPE           1670214 non-null  object
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64
```

```
[137]: p_appl_data.shape

[137]: (1670214, 37)
```

Handling NULL values

```
[500]: null_cols=((p_appl_data.isnull().sum()*100)/p_appl_data.shape[0]).round(2)
       null_cols[null_cols>0]

[500]: AMT_ANNUITY                  22.29
       AMT_DOWN_PAYMENT             53.64
       AMT_GOODS_PRICE              23.08
       RATE_DOWN_PAYMENT            53.64
       RATE_INTEREST_PRIMARY        99.64
       RATE_INTEREST_PRIVILEGED     99.64
       NAME_TYPE_SUITE              49.12
       CNT_PAYMENT                  22.29
       PRODUCT_COMBINATION           0.02
       DAYS_FIRST_DRAWING           40.30
       DAYS_FIRST_DUE               40.30
       DAYS_LAST_DUE_1ST_VERSION    40.30
       DAYS_LAST_DUE                40.30
       DAYS_TERMINATION             40.30
       NFLAG_INSURED_ON_APPROVAL    40.30
       dtype: float64
```

We see high percentage of null values in the above columns, dropping all columns having more than 20% of null values.

```
[502]: p_appl_data.drop(p_appl_data.loc[:,p_appl_data.isnull().mean()>=.20],axis=1,inplace=True)

[503]: ((p_appl_data.isnull().sum()*100)/p_appl_data.shape[0]).round(2)

[503]: SK_ID_PREV                     0.00
       SK_ID_CURR                     0.00
       NAME_CONTRACT_TYPE             0.00
       AMT_APPLICATION                0.00
       AMT_CREDIT                     0.00
       WEEKDAY_APPR_PROCESS_START     0.00
       HOUR_APPR_PROCESS_START        0.00
       FLAG_LAST_APPL_PER_CONTRACT    0.00
       NFLAG_LAST_APPL_IN_DAY         0.00
       NAME_CASH_LOAN_PURPOSE         0.00
       NAME_CONTRACT_STATUS           0.00
       DAYS_DECISION                  0.00
       NAME_PAYMENT_TYPE              0.00
       CODE_REJECT_REASON             0.00
       NAME_CLIENT_TYPE               0.00
       NAME_GOODS_CATEGORY            0.00
       NAME_PORTFOLIO                 0.00
       NAME_PRODUCT_TYPE              0.00
       CHANNEL_TYPE                   0.00
       SELLERPLACE_AREA               0.00
       NAME_SELLER_INDUSTRY           0.00
       NAME_YIELD_GROUP               0.00
       PRODUCT_COMBINATION            0.02
       dtype: float64

[504]: p_appl_data.shape

[504]: (1670214, 23)
```

After dropping high null value columns, Impute Product_combination with mode, it is the only column which has null values left.

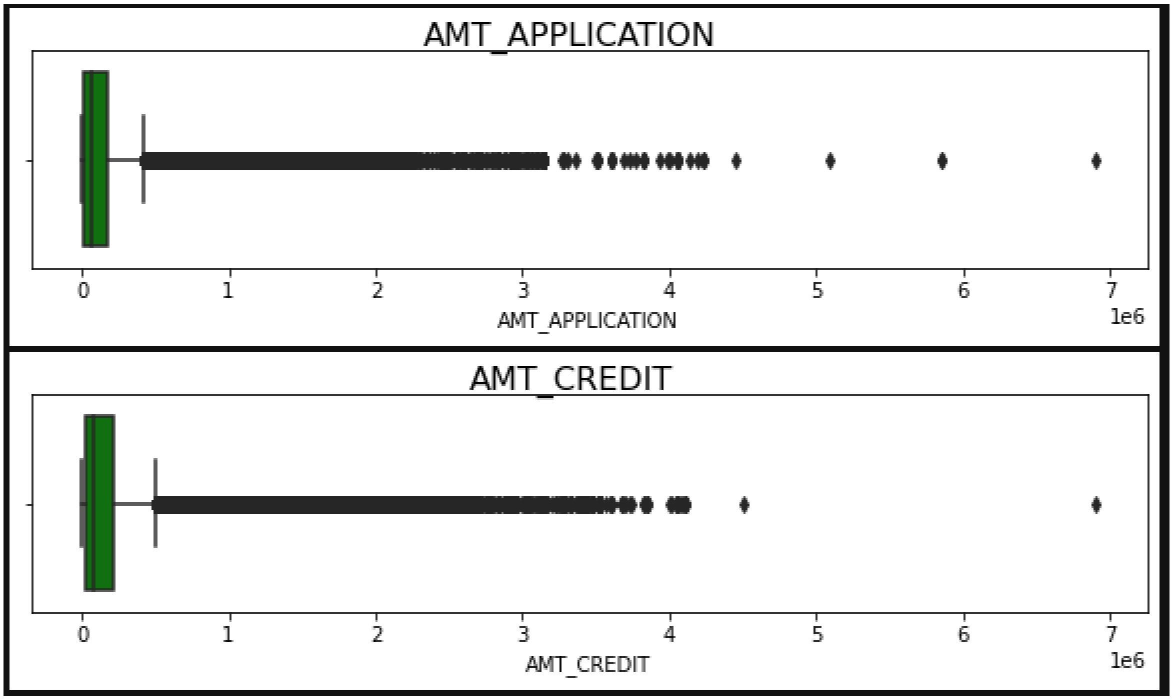
```
[505]: p_appl_data.PRODUCT_COMBINATION.mode()

[505]: 0    Cash
       dtype: object

[506]: p_appl_data['PRODUCT_COMBINATION'].fillna(p_appl_data.PRODUCT_COMBINATION.mode()[0], inplace=True)
       p_appl_data.PRODUCT_COMBINATION.isnull().sum()

[506]: 0
```

Identifying & Handling Outliers

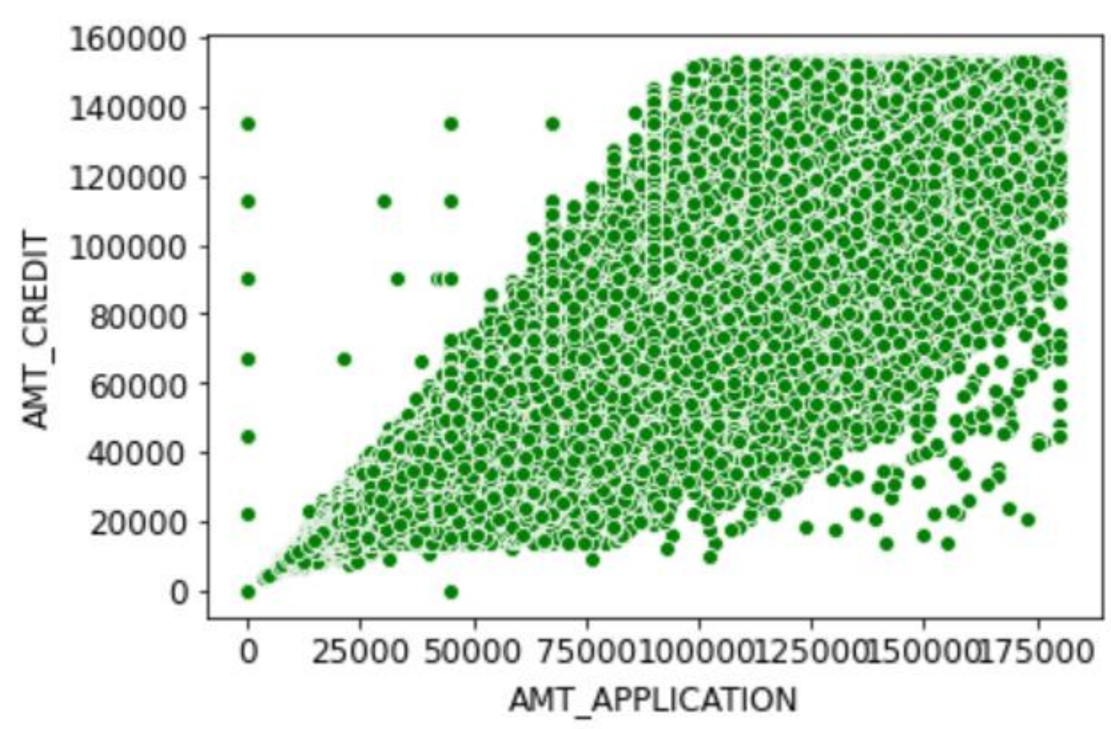


By looking at the data columns individually we decided on what percentile of data should be retained in the dataset. Which can make analysis accurate and does not result in skewed patterns. For determining the percentile (%) we looked at max values and retained only those outliers which are relatively closer to the 75% quantile values. We made sure all high value anomalies are completed removed.

Following is the list of columns and the corresponding quantiles below which we have retain the data.

Columns Name	Retain % data
AMT_APPLICATION	75%
AMT_CREDIT	90%

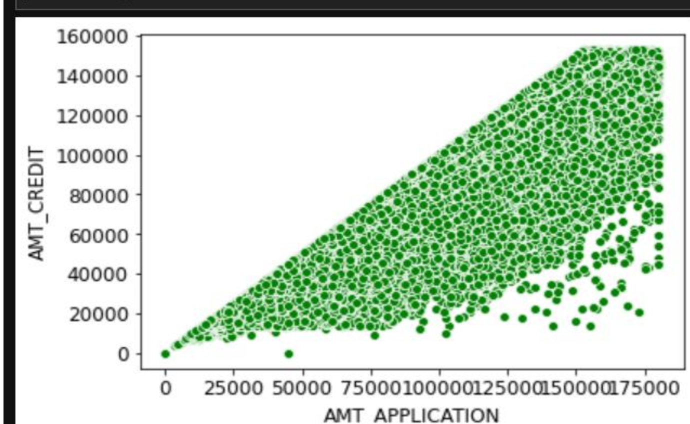
Fixing AMT_CREDIT and AMT_APPLICATION Anomalies



Ideally credit amount should not be greater than application amount. Hence, we should clean these values.

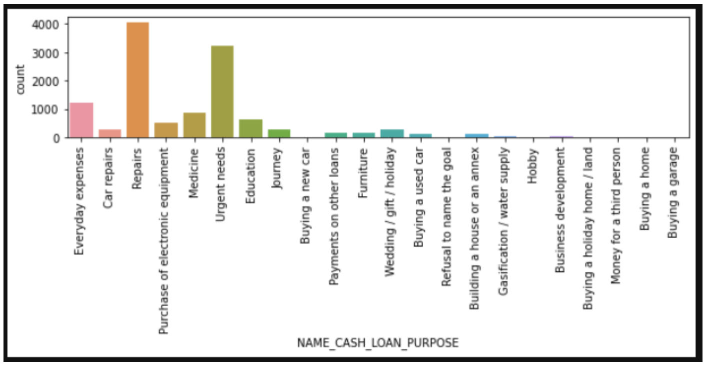
```
p_appl_data=p_appl_data[~(p_appl_data.AMT_CREDIT > p_appl_data.AMT_APPLICATION)]
```

```
sns.scatterplot(x = 'AMT_APPLICATION', y="AMT_CREDIT",data=p_appl_data,color='green')
plt.show()
```

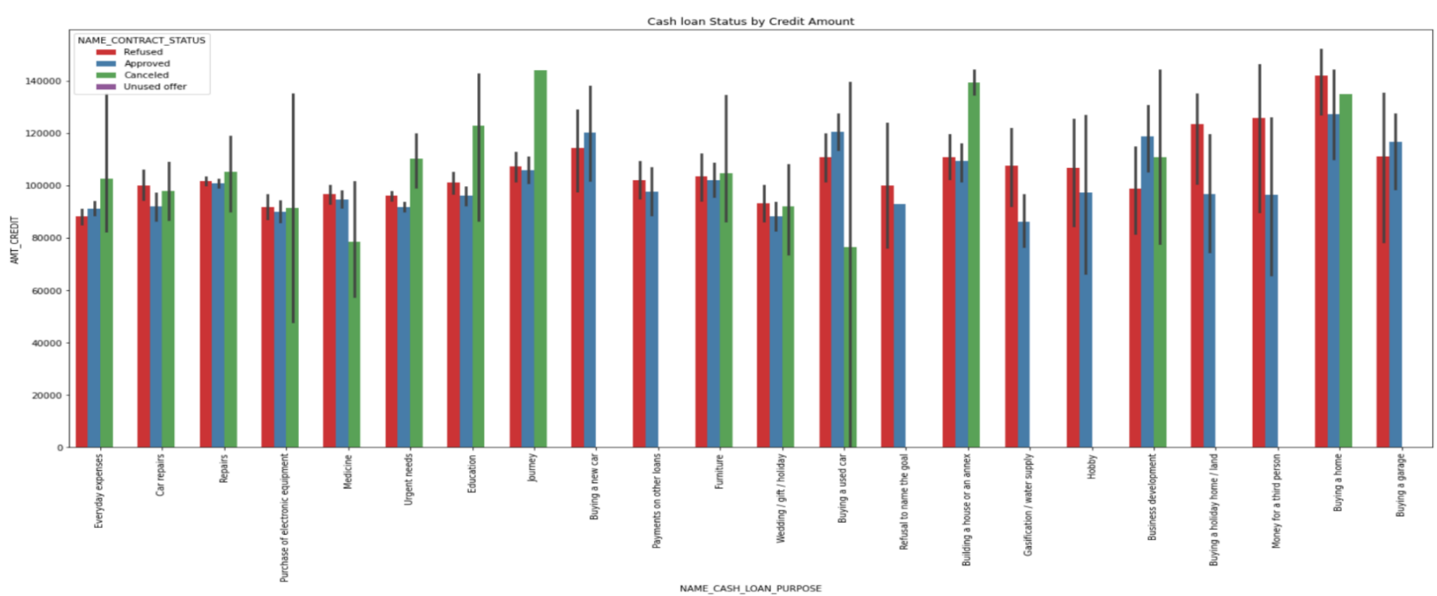


Performing Data Analysis and Identifying Insights

Cash Loans



Most number of cash loan applications are for Repairs.



Observation:

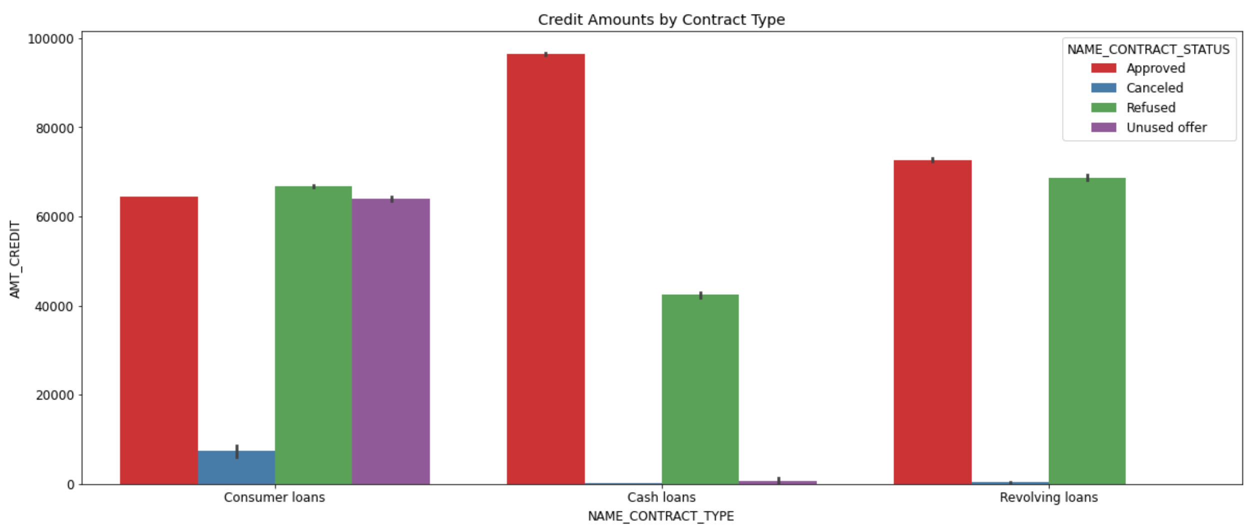
1. Buying a home has the most cash loan approvals in terms of credit amounts.

2. Journey and building a house has the greatest number of cash loan cancellations.

3. Business development has the highest ratio of credit amounts approved vs refused.

4. Hobby, Gasification, buying new car, buying a garage has no cancellations.

Contract Type

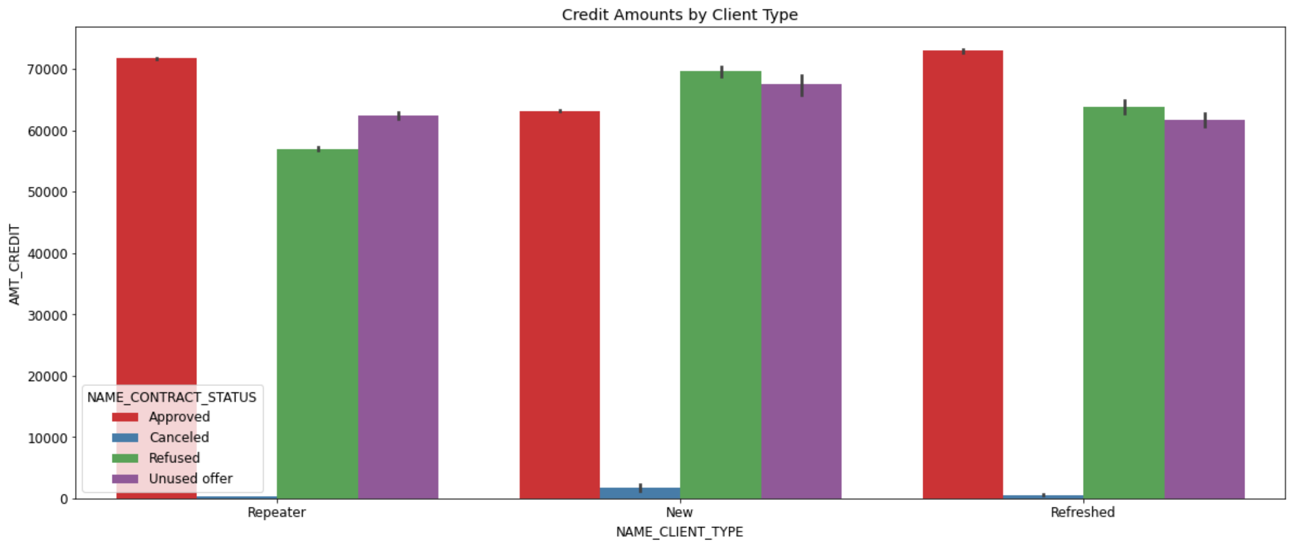


Observation:

1. Cash loans are approved for higher credit amounts than consumer and revolving loans and refused the least.

2. Consumer loans have higher cancellations as compared to cash loans and revolving loans.

Client Type:



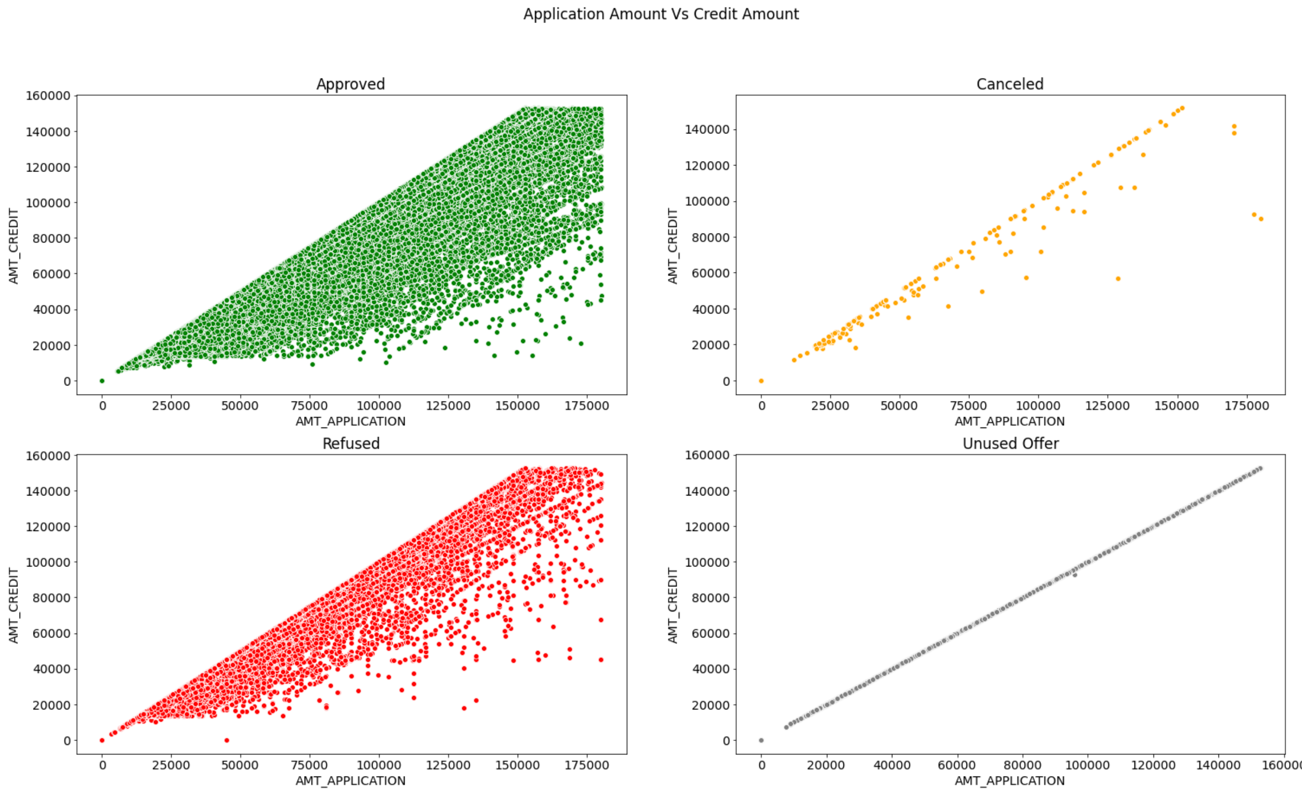
Observation:

1. Loans for greater than 70K+ are getting approved for Repeaters and Refreshed client as compared to New clients.

2. New clients have higher cancellations as compared to repeaters and refreshed clients.

3. New clients are refused on high credit loan applications.

Application Amount Vs Credit amount:



Observation:

1. We can see a linear progression between credit amount and application amount across all decision's types.

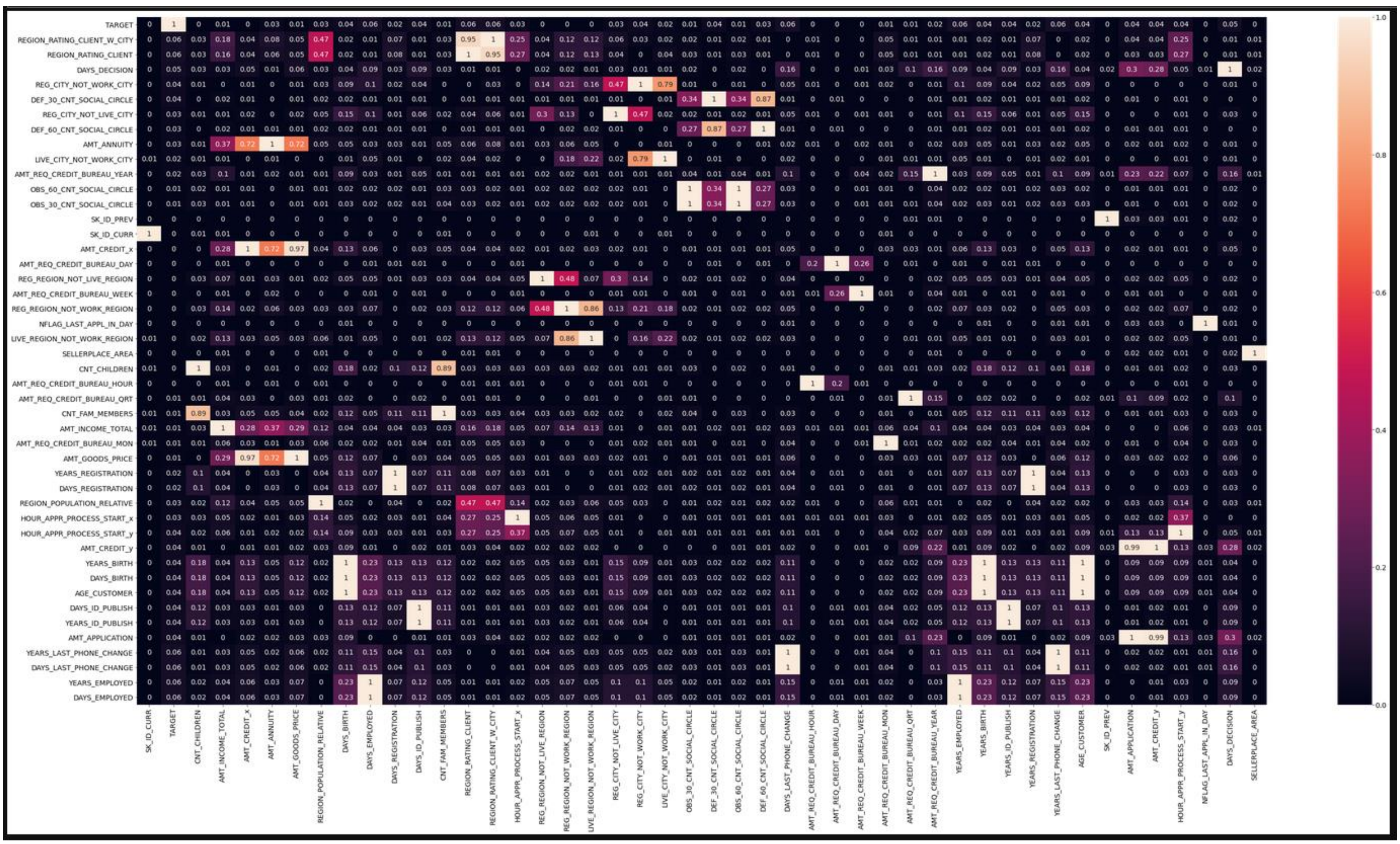
2. We observed on cancelled applications the amount credit and amount application differ by some degree. This can be a reason why applications may have been cancelled.

Merging Current and Previous Application:

9. Data Merging of Application data

```
combined_data = pd.merge(appl_data, p_appl_data, how='left', on=['SK_ID_CURR'])
combined_data
```

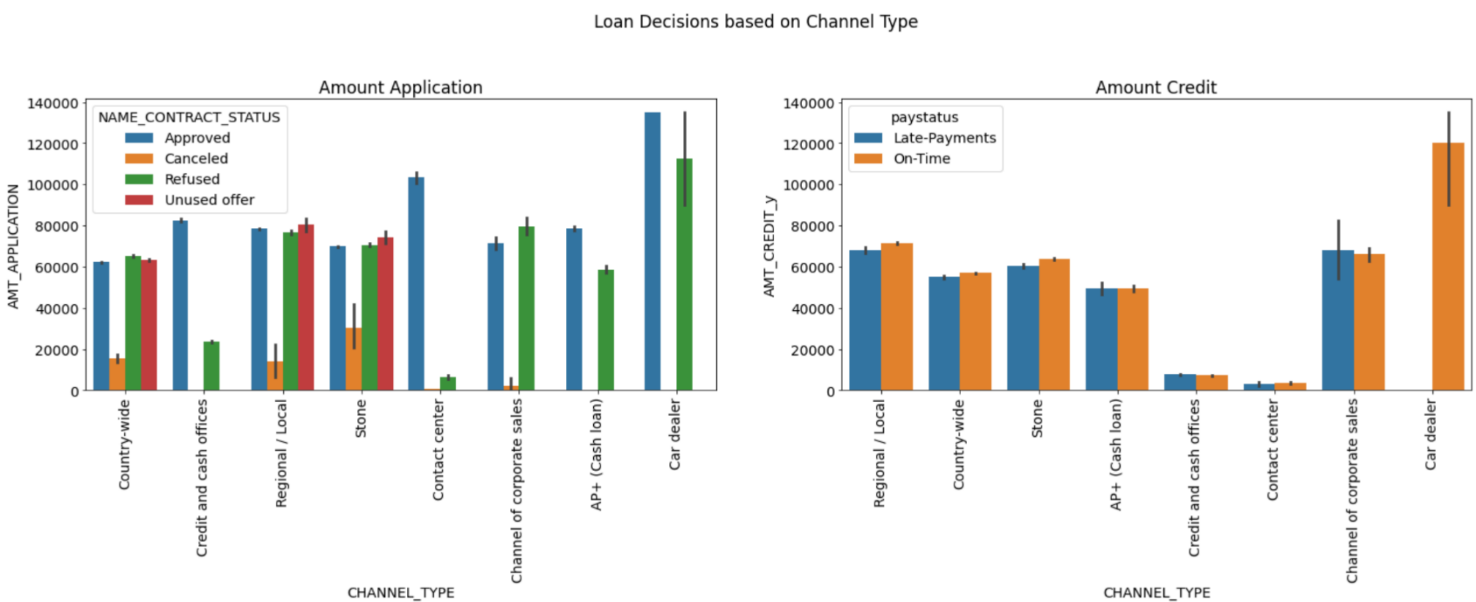
Identifying combined correlation of contributing factors:



Based on the above heatmap, we have identified the highly corelated attributes:

```
AMT_APPLICATION             AMT_CREDIT_y                 0.99
AMT_CREDIT_y                AMT_APPLICATION              0.99
AMT_GOODS_PRICE             AMT_CREDIT_x                 0.97
AMT_CREDIT_x                AMT_GOODS_PRICE              0.97
REGION_RATING_CLIENT        REGION_RATING_CLIENT_W_CITY  0.95
REGION_RATING_CLIENT_W_CITY REGION_RATING_CLIENT         0.95
CNT_CHILDREN                CNT_FAM_MEMBERS              0.89
CNT_FAM_MEMBERS             CNT_CHILDREN                 0.89
DEF_30_CNT_SOCIAL_CIRCLE    DEF_60_CNT_SOCIAL_CIRCLE     0.87
DEF_60_CNT_SOCIAL_CIRCLE    DEF_30_CNT_SOCIAL_CIRCLE     0.87
LIVE_REGION_NOT_WORK_REGION REG_REGION_NOT_WORK_REGION   0.86
REG_REGION_NOT_WORK_REGION  LIVE_REGION_NOT_WORK_REGION  0.86
REG_CITY_NOT_WORK_CITY      LIVE_CITY_NOT_WORK_CITY      0.79
LIVE_CITY_NOT_WORK_CITY     REG_CITY_NOT_WORK_CITY       0.79
AMT_ANNUITY                 AMT_GOODS_PRICE              0.72
                            AMT_CREDIT_x                 0.72
AMT_CREDIT_x                AMT_ANNUITY                  0.72
AMT_GOODS_PRICE             AMT_ANNUITY                  0.72
```

Loan Decisions based on Channel Type:



Observation:

1. Car dealers are good customers as they have less late payments and are approved more than they are refused.

2. Contact center are highly approved and have almost equal on-time vs late payment ratio.

3. Channel of Corporate sales are risky as they have higher late payment ratios and are refused loans more than approved.