Credit Applicant Fraud & Potential Detection

Detect fraudulent activity before allotment of loan.

-Bhavit Kanthalia
Under Prof. Dr. Soumya Dutta

Objective

In an era of increasing reliance on digital financial transactions and credit availability, the menace of credit fraud looms larger than ever.

"Credit Fraud Detection" project is aimed at fortifying financial systems against fraudulent activities by employing advanced data analysis techniques to prevent the allotment of loans in the future. This initiative aims to create a sophisticated defense mechanism that detects credit fraud swiftly.

By leveraging advanced analytics and verification of data, this initiative seeks to redefine the standards of credit fraud detection, ensuring a secure and reliable financial ecosystem for individuals and businesses alike.

Key contents of the Dataset

1. The dataset consists of mock PI Information about the applicants is comprised of many columns like gender, age, marital status, family member counts, children count, highest qualification achieved, contact details like numbers, and mail address.
2. Apart from PI, it contains address information that is bifurcated into city, state, locality, rental or own asset, apartment details, building type, and years of accommodation.
3. Financial details are also incorporated, like the total income of the applicant (yearly), asset details like car ownership, any real estate owned, and the Client's income type (businessman, working, maternity leave,…).
4. Miscellaneous details like purpose of the cash loan, sales channel (from what sources the applicant applied), new customer or old, previous application details (if present).

Initial Tasks

1. Dataset preparation and cleaning.
    a. Selection of key columns.
    b. Detection of columns and rows which contains empty/null values
    c. Handling of null or empty values (will remove the columns if empty values percentage is more than 30 %)
    d. Will try to impute the empty values eg: mean, mode, most frequent value, forward/backward fill
    e. Detection of outliers. By using Boxplot by matplotlib or by any other library.
2. Python set up for 3.11 version
3. Jupyter notebook or Pycharm setup (Done)
4. Find the basis of observations.
5. You can find dataset here

Libraries which can be useful

Libraries

1. NumPy
2. Pandas
3. Seaborn
4. Matplotlib
5. Plotly
6. CSV

Tools and softwares:

1. Jupyter notebook
2. Python 3
3. Github

Expected Outcome from the analysis

1. To detect the pattern, from which we can identify the applicant's risk to be a defaulter on the basis of:

   a. Employment details and PII/BII, Income details

   b. Education acquired

   c. Family count and marital status

   d. Address proof and assets shown

2. Identify payment difficulties of the applicants by using above mentioned categorical columns with uni and Bivariate analysis of the columns. And with the correlation we can achieve this.

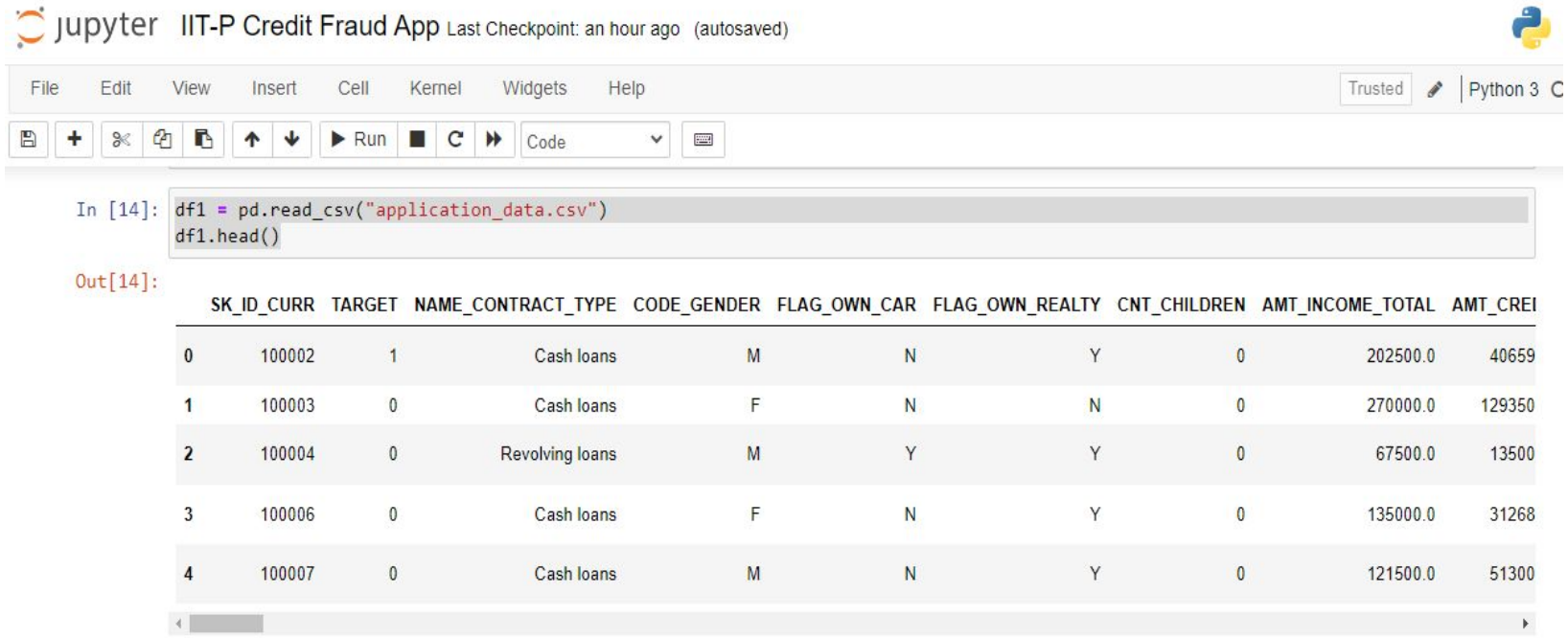
Example of the risk prediction in future scope

1. Male clients with Incomplete Education having very low salaries have a high risk of default.
2. **Senior Citizens**(60-100) and **Very young**(19-25) age group facing paying difficulties less as compared to other age groups.
3. The client's permanent address does not match the contact address are having higher risk to default.
4. Some of the clients with Higher Education, Incomplete Higher Education, Lower Secondary Education and Secondary/Secondary Special Education are more likely to take a high amount of credit loans
5. Businessman, students and Unemployed less likely to apply for loan .
6. Working category have high risk to default.
7. State Servant is at Minimal risk to default.

Data cleaning steps followed

1. Selection of important columns.
2. Detection of null values in the columns and handling the scenario
3. Detection of outliers and handling the scenarios
4. Software, libraries and tools used:
   Python 3.7.4 release version
   Jupyter notebook
   NumPy, Pandas, Seaborn, matplotlib

1. **Reading data: #Reading application data**
   df1 = pd.read_csv("application_data.csv")



	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

Finding columns with null values present

1. We will first try to find the columns which contains null values more than 35%

```
null_col = df1.isnull().sum().sort_values(ascending = False)

null_col = null_col[null_col.values >(0.35*len(df1))]
```

2. Plot these columns in descending order

```
plt.figure(figsize=(20,4))
null_col.plot(kind='bar', color="#4CB391")
plt.title('List of Columns & null counts where null values are more than 35%')
plt.xlabel("Null Columns",fontdict={"fontsize":12,"fontweight":5})            #Setting X-label and Y-label
plt.ylabel("Count of null values",fontdict={"fontsize":12,"fontweight":5})
plt.show()
```

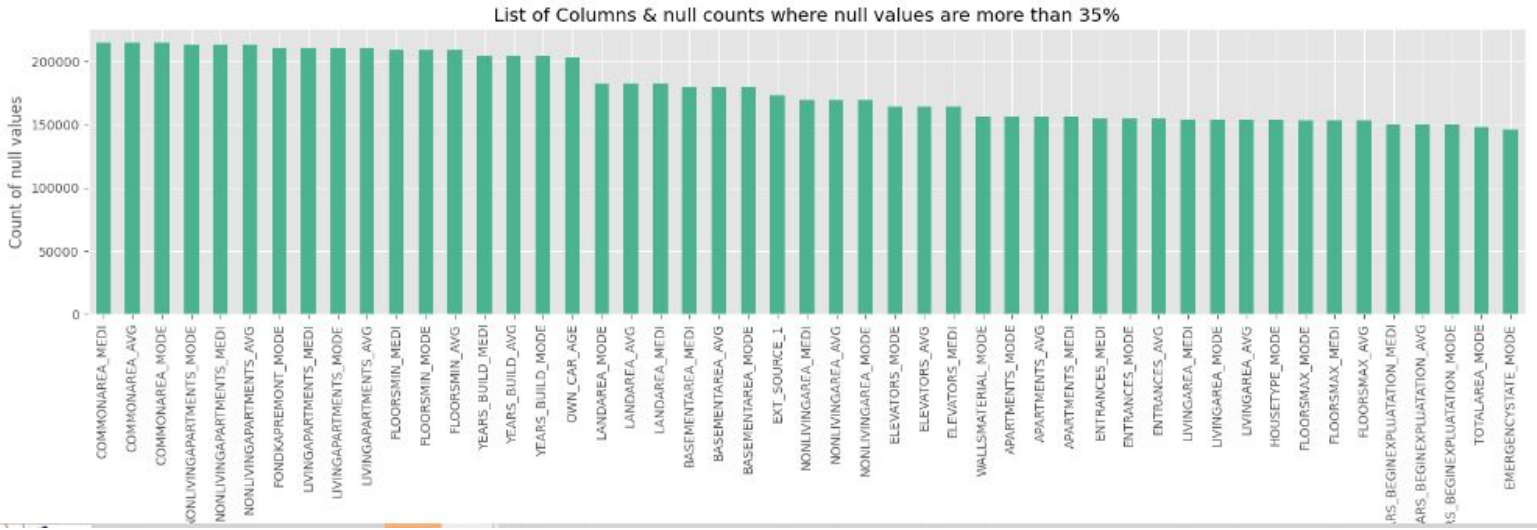
jupyter IIT-P Credit Fraud App Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [16]:
```
null_col = df1.isnull().sum().sort_values(ascending = False)
null_col = null_col[null_col.values >(0.35*len(df1))]
```

In [17]:
```
plt.figure(figsize=(20,4))
null_col.plot(kind='bar', color="#4CB391")
plt.title('List of Columns & null counts where null values are more than 35%')

plt.xlabel("Null Columns",fontdict={"fontsize":12,"fontweight":5})                    #Setting X-Label and Y-Label
plt.ylabel("Count of null values",fontdict={"fontsize":12,"fontweight":5})
plt.show()
```



Columns count after removal and before removal

In [46]:
```
#Total columns which contains null value more than 35%

len(null_col)
```

Out[46]: 49

In [47]:
```
#Total number columns present in the data set

df1.shape
```

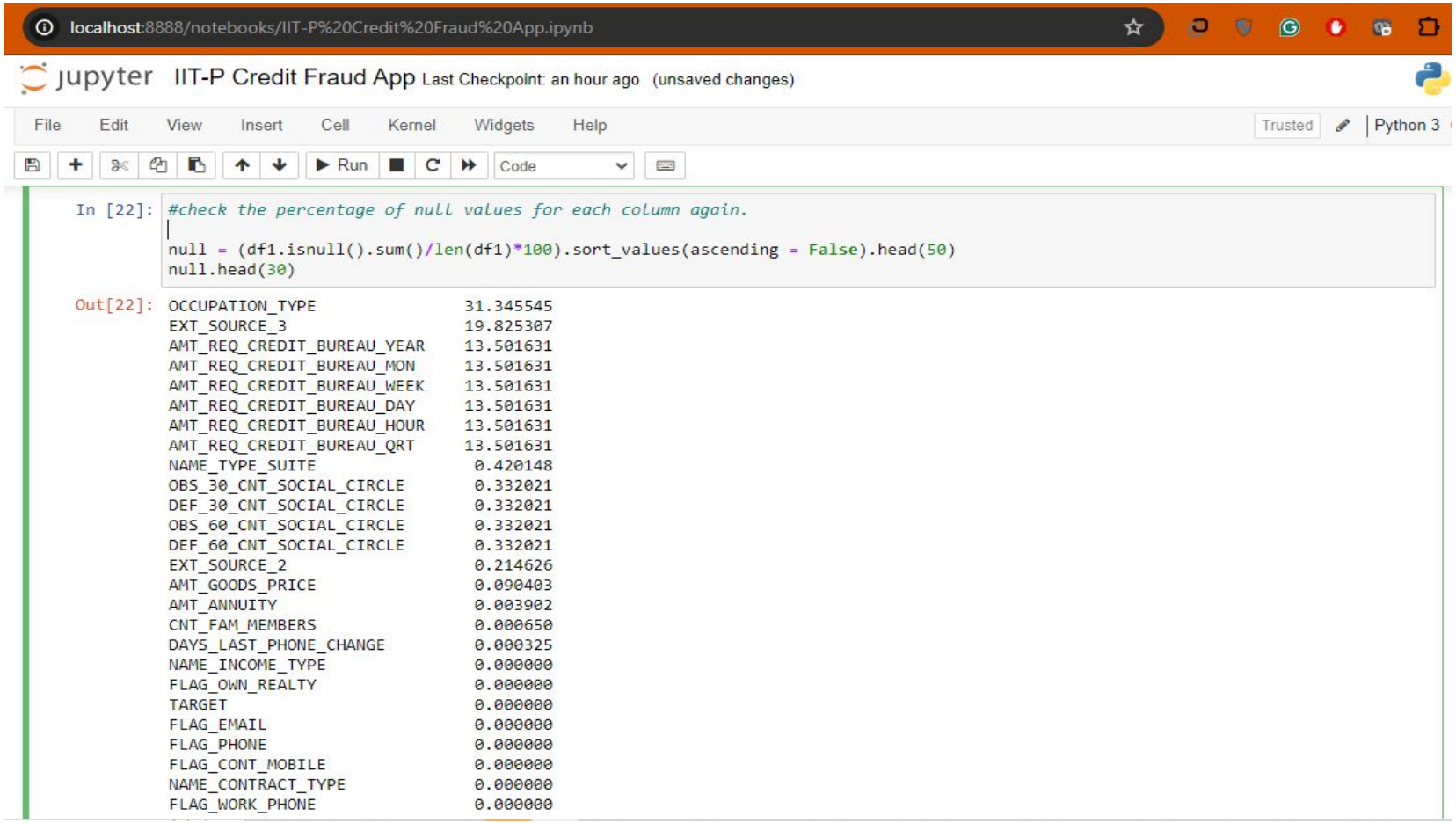
Out[47]: (307511, 122)

In [49]:
```
#Droping these columns from the data set

label = list(null_col.index.values)
df1.drop(labels = label,axis=1,inplace = True)
```

In [50]:
```
#After removal of columns total number of columns remaining
df1.shape
```

Out[50]: (307511, 73)

Check the null values again after removal of columns



Handling the null values using mode in place

In [23]:
```
#Handling the null values by placing mode
df1.AMT_REQ_CREDIT_BUREAU_YEAR.fillna(df1.AMT_REQ_CREDIT_BUREAU_YEAR.mode()[0],inplace = True) #AMT_REQ
df1.AMT_REQ_CREDIT_BUREAU_MON.fillna(df1.AMT_REQ_CREDIT_BUREAU_MON.mode()[0],inplace = True)   #AMT_REQ
df1.AMT_REQ_CREDIT_BUREAU_WEEK.fillna(df1.AMT_REQ_CREDIT_BUREAU_WEEK.mode()[0],inplace = True) #AMT_REQ
df1.AMT_REQ_CREDIT_BUREAU_DAY.fillna(df1.AMT_REQ_CREDIT_BUREAU_DAY.mode()[0],inplace = True)   #AMT_REQ
df1.AMT_REQ_CREDIT_BUREAU_HOUR.fillna(df1.AMT_REQ_CREDIT_BUREAU_HOUR.mode()[0],inplace = True) #AMT_REQ
df1.AMT_REQ_CREDIT_BUREAU_QRT.fillna(df1.AMT_REQ_CREDIT_BUREAU_QRT.mode()[0],inplace = True)   #AMT_REQ
df1.NAME_TYPE_SUITE.fillna(df1.NAME_TYPE_SUITE.mode()[0],inplace = True) #NAME_TYPE_SUITE
df1.OBS_30_CNT_SOCIAL_CIRCLE.fillna( df1.OBS_30_CNT_SOCIAL_CIRCLE.mode()[0],inplace = True) #OBS_30_CNT
df1.DEF_30_CNT_SOCIAL_CIRCLE.fillna( df1.DEF_30_CNT_SOCIAL_CIRCLE.mode()[0],inplace = True) #DEF_30_CNT
df1.OBS_60_CNT_SOCIAL_CIRCLE.fillna( df1.OBS_60_CNT_SOCIAL_CIRCLE.mode()[0],inplace = True) #OBS_60_CNT
df1.DEF_60_CNT_SOCIAL_CIRCLE.fillna( df1.DEF_60_CNT_SOCIAL_CIRCLE.mode()[0],inplace = True) #DEF_60_CNT
df1.CNT_FAM_MEMBERS.fillna(df1.CNT_FAM_MEMBERS.mode() , inplace = True) #CNT_FAM_MEMBERS
df1.DAYS_LAST_PHONE_CHANGE.fillna(df1.DAYS_LAST_PHONE_CHANGE.mode()[0],inplace = True) #DAYS_LAST_PHONE
```

Checking null values again after imputing

In [24]:
```
(df1.isnull().sum()/len(df1)*100).sort_values(ascending=False)
```

Out[24]:
```
OCCUPATION_TYPE               31.345545
AMT_GOODS_PRICE                0.090403
AMT_ANNUITY                    0.003902
CNT_FAM_MEMBERS                0.000650
AMT_REQ_CREDIT_BUREAU_YEAR     0.000000
                                  ...
FLAG_DOCUMENT_3                0.000000
FLAG_DOCUMENT_4                0.000000
FLAG_DOCUMENT_5                0.000000
FLAG_DOCUMENT_6                0.000000
SK_ID_CURR                     0.000000
Length: 73, dtype: float64
```

Replaced Y,N flags with 1,0

In [51]:
```
#Replaced Y,N flags with 1,0

df1['FLAG_OWN_CAR'] = np.where(df1['FLAG_OWN_CAR']=='Y', 1 , 0)
df1['FLAG_OWN_REALTY'] = np.where(df1['FLAG_OWN_REALTY']=='Y', 1 , 0)
```

Gender wise count

```
In [52]: #Gender wise count
         df1.CODE_GENDER.value_counts()

Out[52]: F      202448
         M      105059
         XNA         4
         Name: CODE_GENDER, dtype: int64
```

Terminologies:
F -> Female
M -> Male
XNA -> Null/ not available

Replacing the XNA with F because as we have more female count

```
In [32]: #Replacing the XNA with F because as we have more female count
         df1.loc[df1.CODE_GENDER == 'XNA','CODE_GENDER'] = 'F'
```

```
In [33]: #After replacing XNA with F
         df1.CODE_GENDER.value_counts()

Out[33]: F    202452
         M    105059
         Name: CODE_GENDER, dtype: int64
```

Handling Occupation_type column

```
In [34]: #Categories in organization type
         df1.ORGANIZATION_TYPE.value_counts().head()

Out[34]: Business Entity Type 3    67992
         XNA                       55374
         Self-employed             38412
         Other                     16683
         Medicine                  11193
         Name: ORGANIZATION_TYPE, dtype: int64
```

```
In [35]: #Categories in Income type column
         df1.NAME_INCOME_TYPE.value_counts()

Out[35]: 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
```

Replacing XNA with Pensioner

```
In [36]: df1['ORGANIZATION_TYPE'] = df1['ORGANIZATION_TYPE'].replace('XNA', 'Pensioner')
         df1['OCCUPATION_TYPE'].fillna('Pensioner' , inplace = True)
```

Here we replaced XNA values with Pensioner in the Occupation_type column
As we can see the values of both the entities lies approximately equal so we can deduce that XNA can be replaced by Pensioner

Removal of unwanted columns

In [53]:
```
#Removal of unwanted columns

unwanted=['FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE',
       'FLAG_PHONE', 'FLAG_EMAIL','REGION_RATING_CLIENT','REGION_RATING_CLIENT_W_CITY','FLAG_EMAIL', 'F
       'REGION_RATING_CLIENT_W_CITY', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3','FLAG_DOCUMENT_4', 'FLAG_DOC
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9','FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11',
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15','FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21']

df1.drop(labels=unwanted,axis=1,inplace=True)
```

In [56]:
```
#Final columns count
df1.shape
```

Out[56]: (307511, 45)

Final columns to be used and numeric columns

In [29]:
```
#Final columns count
df1.shape
```

Out[29]:
```
(307511, 47)
```

In [30]:
```
numerical_col = df1.select_dtypes(include='number').columns
len(numerical_col)
```

Out[30]:
```
35
```

In [32]:
```
numerical_col
```

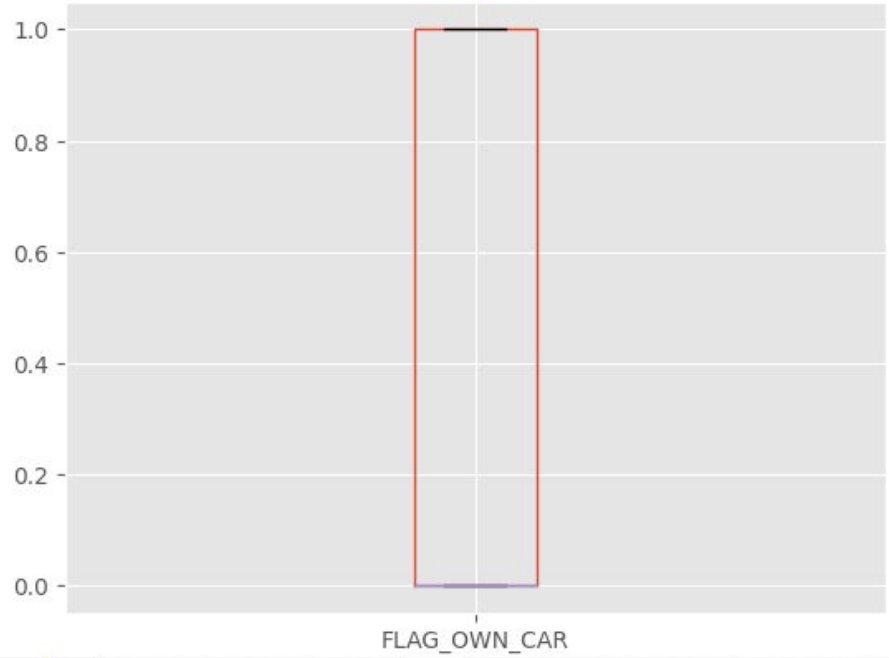
Out[32]:
```
Index(['SK_ID_CURR', 'TARGET', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
       'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY',
       'AMT_GOODS_PRICE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH',
       'CNT_FAM_MEMBERS', 'HOUR_APPR_PROCESS_START',
       '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_2',
       'EXT_SOURCE_3', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', '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'],
      dtype='object')
```

Outlier check in own_car column

In [47]:

Slide Type

```
df1.boxplot(column='FLAG_OWN_CAR', return_type='axes');
```

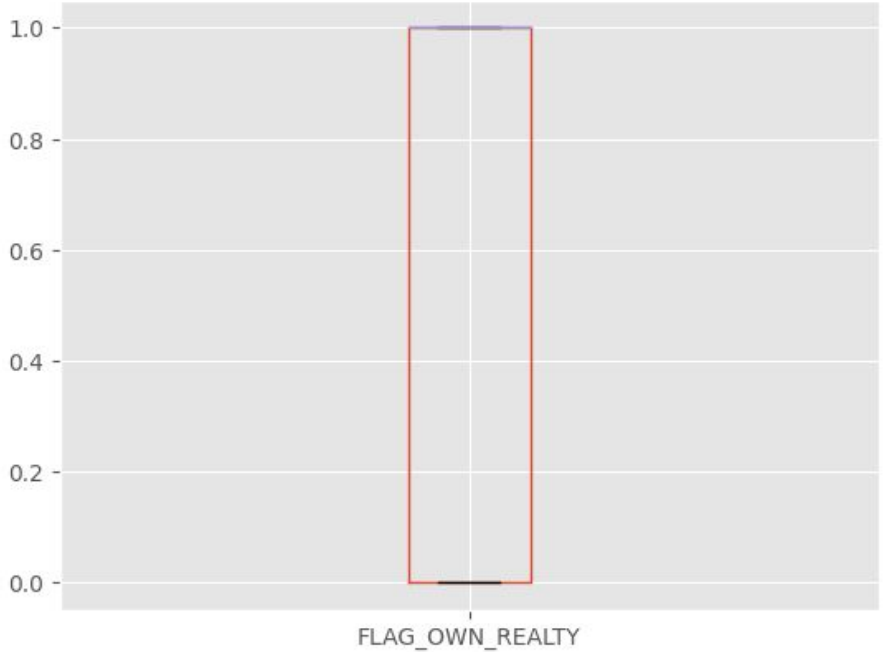


Outlier detection in own_realty column

In [48]:

Slide Type

```
df1.boxplot(column='FLAG_OWN_REALTY', return_type='axes');
```



Outlier check in family count column

In [45]:

Slide Type

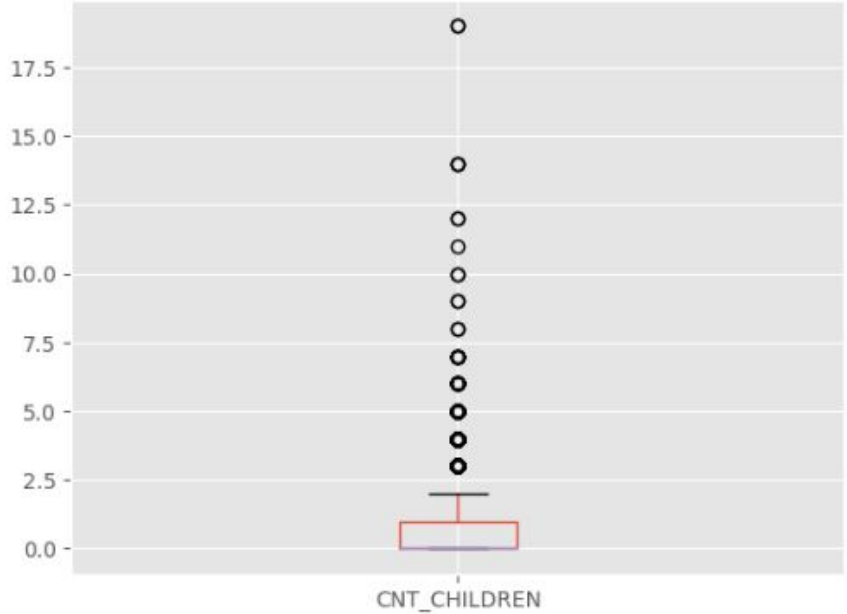
```
df1.boxplot(column='CNT_FAM_MEMBERS', return_type='axes');
```

Outlier detection in children count column

In [42]:

Slide Type

```
df1.boxplot(column='CNT_CHILDREN', return_type='axes');
```

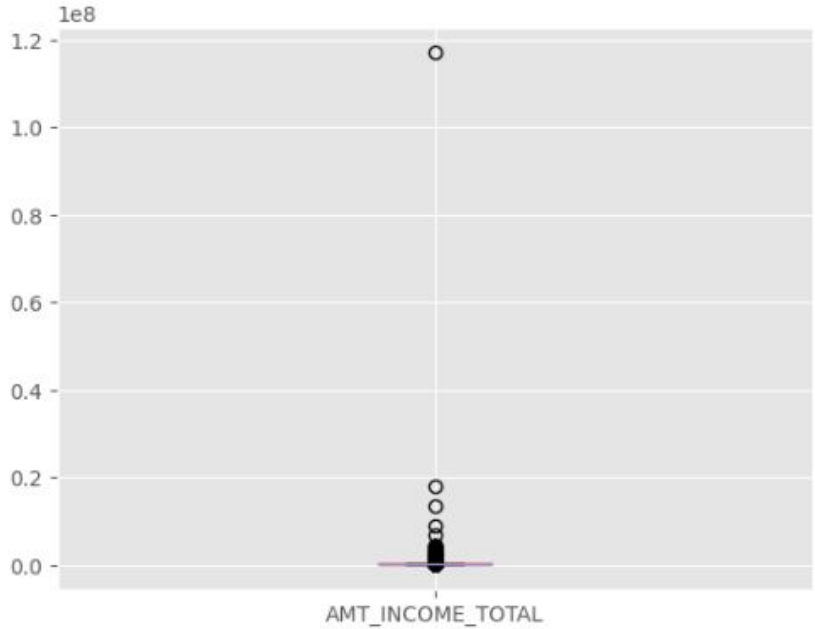


Outlier detection in Income total column

In [43]:

Slide Type

```
df1.boxplot(column='AMT_INCOME_TOTAL', return_type='axes');
```



Observations

1. For FLAG_OWN_CAR values lies within IQR, So we can conclude that most of the clients own a car.
2. For FLAG_OWN_REALTY values lies within IQR, So we can conclude that most of the clients own a car.
3. For CNT_FAM_MEMBERS , we can say that most of the clients have 4 family members.
4. CNT_CHILDREN have outlier values having children more than 4-5.

5. IQR for AMT_INCOME_TOTAL is very slim and it has a large number of outliers.

TARGET - Target variable (1 - applicants with payment difficulties: he/she had late payment more than X days on at least one of the first Y installments of the loan in our sample, 0 - all other cases) in past transaction with the organization

It contains binary values (0,1)

In [109]:

Slide Type

```
#Group by the defaulters and non defulaters
#Non-Defaulters
Target0 = df1.loc[df1["TARGET"]==0]

#Defaulters
Target1 = df1.loc[df1["TARGET"]==1]
```

Graphical representation of the Target column

In [55]:

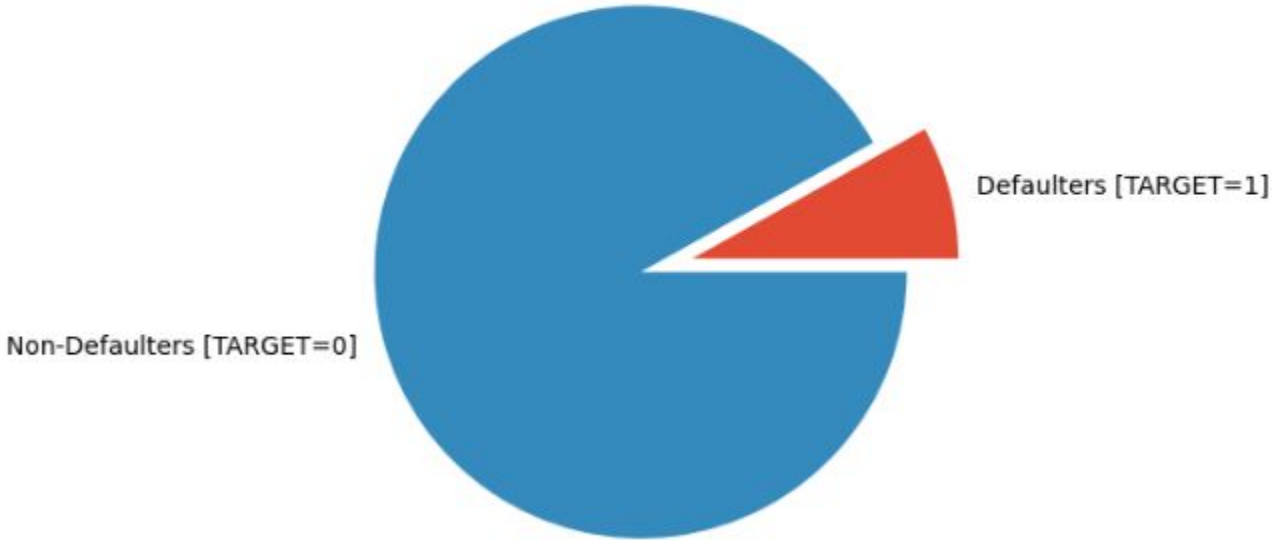
Slide Type

```
count1 = 0
count0 = 0
for i in df1['TARGET'].values:
    if i == 1:
        count1 += 1
    else:
        count0 += 1

count1 = (count1/len(df1['TARGET']))*100
count0 = (count0/len(df1['TARGET']))*100

print('Defaulters : ',count1)
print('Non-Defaulters : ',count0)

y = [count1, count0]

mylabels = ["Defaulters [TARGET=1]", "Non-Defaulters [TARGET=0]"]
myexplode = [0.2, 0,]

plt.pie(y, labels = mylabels, explode = myexplode)
plt.show()

#Reference - https://www.w3schools.com/python/matplotlib_pie_charts.asp
```

```
Defaulters :  8.072881945686495
Non-Defaulters :  91.92711805431351
```

Pie Chart distribution between Target values

```
Defaulters :  8.072881945686495
Non-Defaulters :  91.92711805431351
```



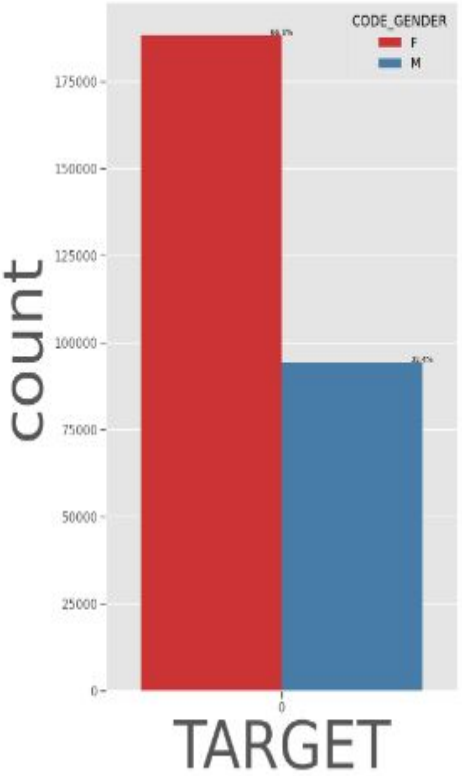
Gender wise distribution for Non-Defaulters

In [89]:

Slide Type

```
#Here the graph will be in 15 x 8 inches
plt.figure(figsize=(10,10))
#121 => 1 column, 2 rows and plot number
plt.subplot(121)
#x variable in data, hue takes column name, data is data frame for which we are plotting
ax = sns.countplot(x='TARGET',hue='CODE_GENDER',data=Target1, palette = 'Set1')
plt.title("Gender Distribution in Target0 [Non-defaulters]")

#Calculating the percetage on the bars
for p in ax.patches:
    percentage = '{:.1f}%'.format(100 * p.get_height()/len(Target1))
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')
plt.show()
```

Gender Distribution in Target0 [Non-defaulters]



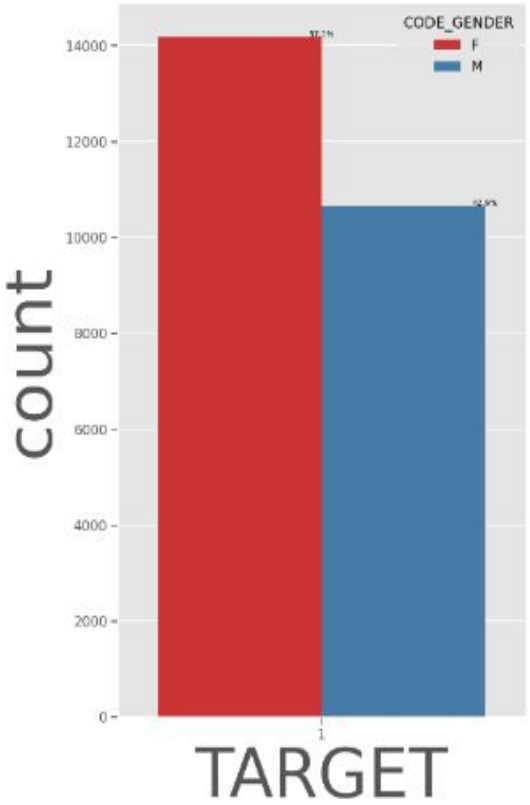
Gender wise distribution for Defaulters

In [166]:

Slide Type -

```
#Here the graph will be in 15 x 8 inches
plt.figure(figsize=(20,15))
#121 => 1 column, 2 rows and plot number
plt.subplot(121)
#x variable in data, hue takes column name, data is data frame for which we are plotting
ax = sns.countplot(x='TARGET',hue='CODE_GENDER',data=Target1, palette = 'Set1')
plt.title("Gender Distribution in Target1 [Defaulters]")

#Calculating the percetage on the bars
for p in ax.patches:
    percentage = '{:.1f}%'.format(100 * p.get_height()/len(Target1))
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')
plt.show()
```

Gender Distribution in Target1 [Defaulters]



Observation for gender wise distribution

1. It seems like Female clients applied higher than male clients for loan.
2. 66.6% Female clients are non-defaulters while 33.4% male clients are non-defaulters.
3. 57% Female clients are defaulters while 42% male clients are defaulters.

Converting days from birth into age and categorizing them

In [136]:

Slide Type

```
df1['DAYS_BIRTH']= (df1['DAYS_BIRTH']/365).astype(int)    # Converting
df1['DAYS_BIRTH'].unique()
df1['AGE_GROUP']=pd.cut(df1['DAYS_BIRTH'],
                        bins=[19,25,35,60,100], labels=['Very_Young','Young', 'Middle_Age', 'Senior_Citizen'])
```

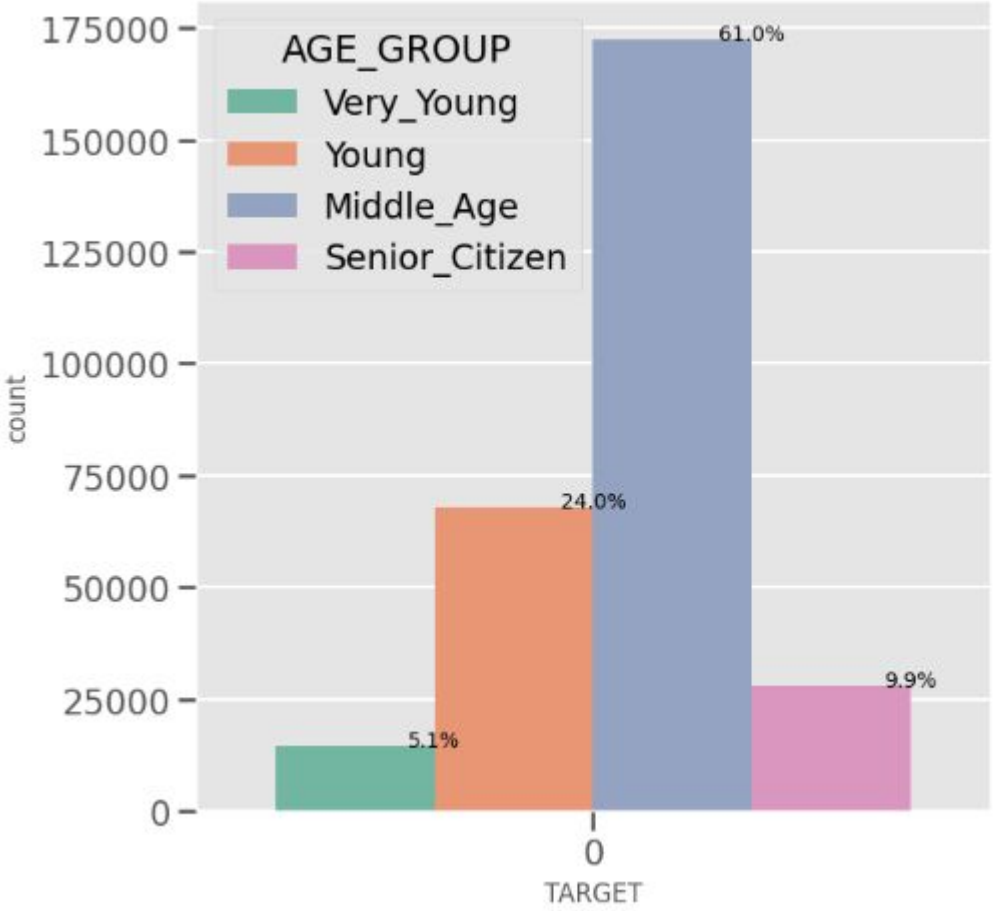
Age wise distribution for Non-defaulters

In [157]:

Slide Type

```
plt.figure(figsize=(15,7))
plt.subplot(121)
ax=sns.countplot(x='TARGET',hue='AGE_GROUP',data=Target0,palette='Set2')
for p in ax.patches:
    percentage = '{:.1f}%'.format(100 * p.get_height()/len(Target0))
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')


plt.show()
```

Graphical representation of Age wise distribution for Non-defaulters



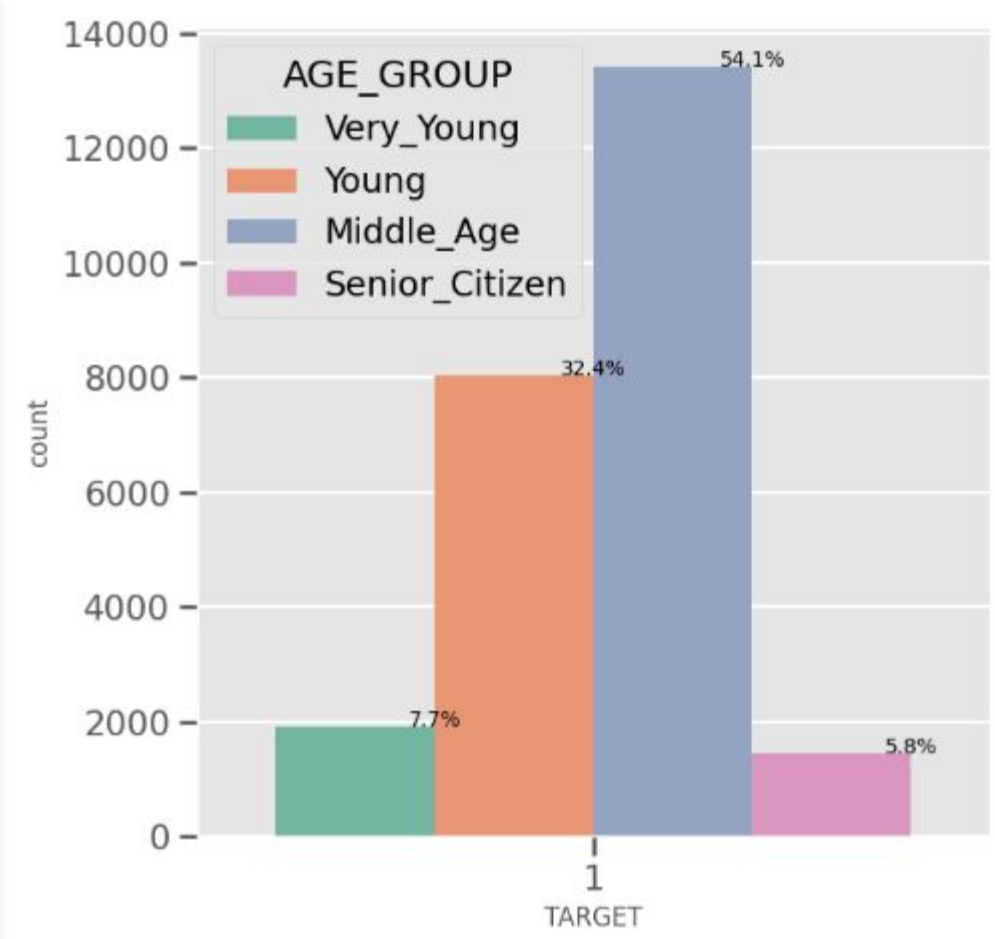
Age wise distribution for Defaulters

In [158]:

Slide Type -

```
plt.figure(figsize=(15,7))
plt.subplot(121)
ax=sns.countplot(x='TARGET',hue='AGE_GROUP',data=Target1,palette='Set2')
for p in ax.patches:
    percentage = '{:.1f}%'.format(100 * p.get_height()/len(Target1))
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')


plt.show()
```

Graphical representation of Age wise distribution for defaulters



Observations

1. Middle Age the group seems to applied higher than any other age group for loans in the case of Defaulters as well as Non-defaulters.

2. Also, Middle Age group facing paying difficulties the most.

3. While Senior Citizens and Very young age group facing paying difficulties less as compared to other age groups.

Parallel coordinate plotting

Columns used :

1. Target: it shows the applicant has defaulter history or not.
   Non-defaulters (Target0->0), Defaulters (Target1 ->1)
2. Education Type : "Incomplete higher"(0), "Higher education"(1), "Academic degree"(2), "Secondary / secondary special"(3), "Lower Secondary"(4)
3. Gender: Female F->0, Male M->1
4. Own_car: N->0, Y->1
5. Own_rrealty; N->0, Y->1

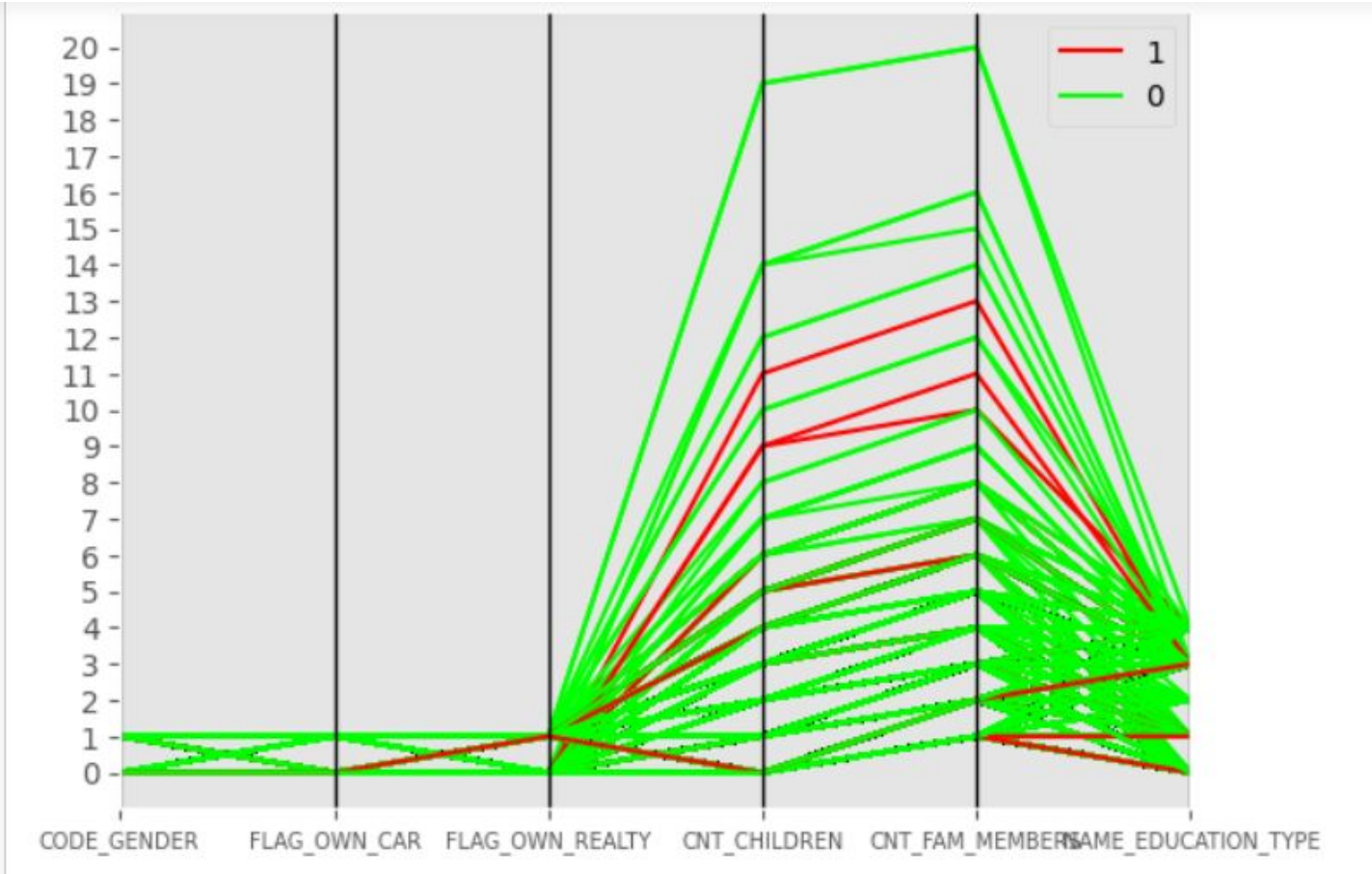
Simple Parallel coordinate plot between variables

In [*]:

Slide Type

```
# import packages
import matplotlib.pyplot as plt
import pandas as pd
from pandas.plotting import parallel_coordinates
import numpy as np

#Defaulters dataframe
defaulters = df1

#Changing education values to int
defaulters["NAME_EDUCATION_TYPE"] = np.where(defaulters["NAME_EDUCATION_TYPE"] == "Incomplete higher", 0, defaulters["NAME_EDUCA
defaulters["NAME_EDUCATION_TYPE"] = np.where(defaulters["NAME_EDUCATION_TYPE"] == "Lower secondary", 1, defaulters["NAME_EDUCATI
defaulters["NAME_EDUCATION_TYPE"] = np.where(defaulters["NAME_EDUCATION_TYPE"] == "Academic degree", 2, defaulters["NAME_EDUCATI
defaulters["NAME_EDUCATION_TYPE"] = np.where(defaulters["NAME_EDUCATION_TYPE"] == "Secondary / secondary special", 3, defaulters
defaulters["NAME_EDUCATION_TYPE"] = np.where(defaulters["NAME_EDUCATION_TYPE"] == "Higher education", 4, defaulters["NAME_EDUCAT

defaulters["CODE_GENDER"] = np.where(defaulters["CODE_GENDER"] == "F", 0, defaulters["CODE_GENDER"])
defaulters["CODE_GENDER"] = np.where(defaulters["CODE_GENDER"] == "M", 1, defaulters["CODE_GENDER"])

defaulters["FLAG_OWN_CAR"] = np.where(defaulters["FLAG_OWN_CAR"] == "N", 0, defaulters["FLAG_OWN_CAR"])
defaulters["FLAG_OWN_CAR"] = np.where(defaulters["FLAG_OWN_CAR"] == "Y", 1, defaulters["FLAG_OWN_CAR"])

defaulters["FLAG_OWN_REALTY"] = np.where(defaulters["FLAG_OWN_REALTY"] == "N", 0, defaulters["FLAG_OWN_REALTY"])
defaulters["FLAG_OWN_REALTY"] = np.where(defaulters["FLAG_OWN_REALTY"] == "y", 1, defaulters["FLAG_OWN_REALTY"])

selective_defaulters = defaulters.filter(['CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','CNT_CHILDREN','CNT_FAM_MEMBERS','TARGE

selective_defaulters

pd.plotting.parallel_coordinates(selective_defaulters,'TARGET',color=('#FF0000', '#00FF00'))

ticks = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
plt.yticks(ticks, ticks)
plt.tick_params(axis='x', which='major', labelsize=3)
plt.show()
```

Parallel coordinate plot for selected variables



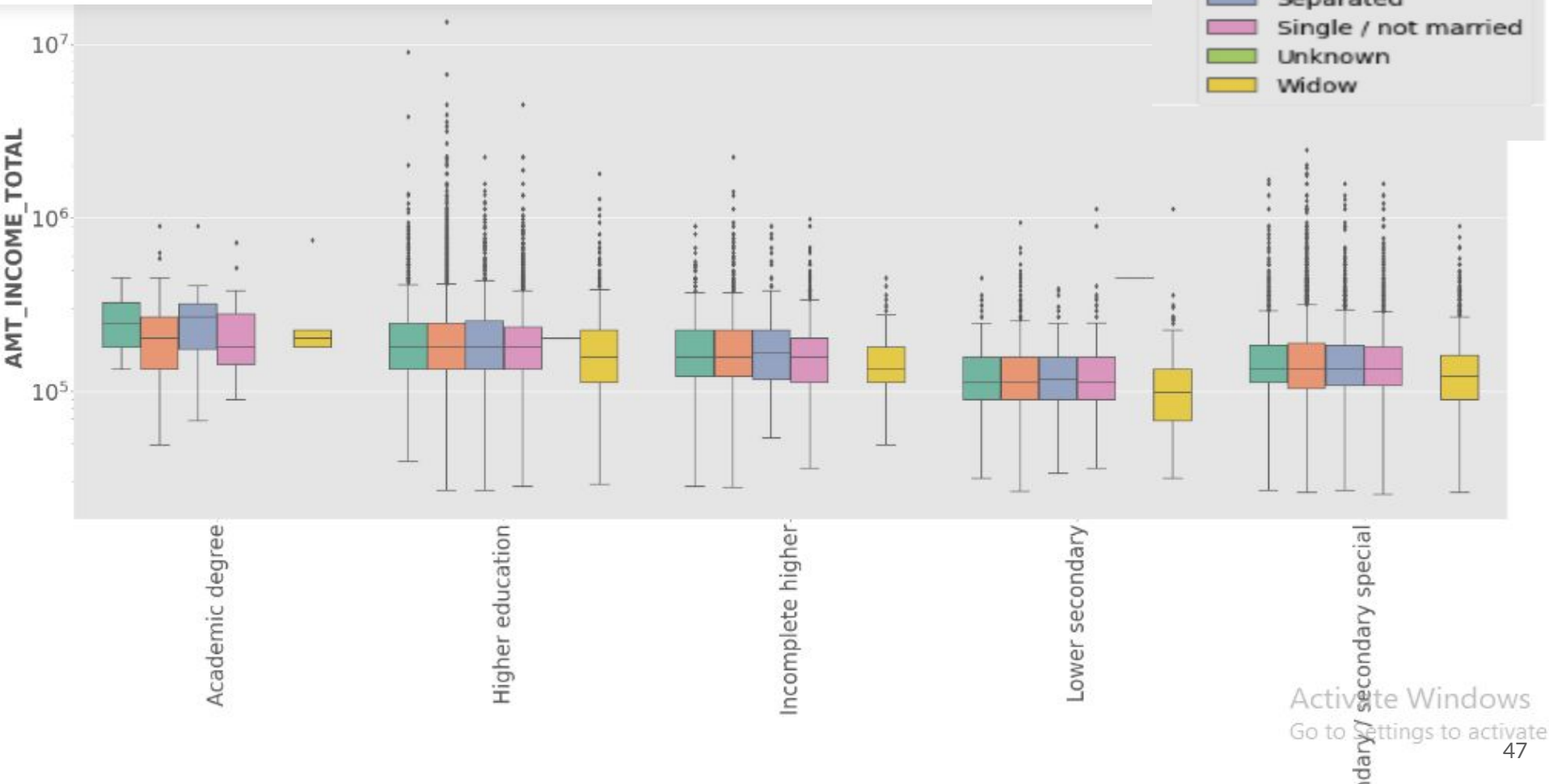
Bivariate analysis with income, education and family status (Non-Defaulters)

In [54]:

Slide Type

```
plt.figure(figsize=(35,14))
plt.yscale('log')                    #As the values are too large, it is convinient to use log for better analysis
plt.xticks(rotation = 90)


sns.boxplot(data =Target0, x='NAME_EDUCATION_TYPE',y='AMT_INCOME_TOTAL',   #Boxplot w.r.t Data Target 0
            hue ='NAME_FAMILY_STATUS',orient='v',palette='Set2')


plt.legend( loc = 'upper right')                                          #Adjusting legend position
plt.title('Income amount vs Education Status',fontsize=35 )
plt.xlabel("NAME_EDUCATION_TYPE",fontsize= 30, fontweight="bold")
plt.ylabel("AMT_INCOME_TOTAL",fontsize= 30, fontweight="bold")
plt.xticks(rotation=90, fontsize=30)
plt.yticks(rotation=360, fontsize=30)

plt.show()
```

Plot



Observation:

1. Applicants with the higher education and incomplete higher education and secondary education have more outliers.
2. In all the education criteria married applicants are having higher income
3. The applicant who are married and having higher and secondary education have higher income and outliers

Bivariate analysis with income, education and family status (Defaulters)

NAME_EDUCATION_TYPE

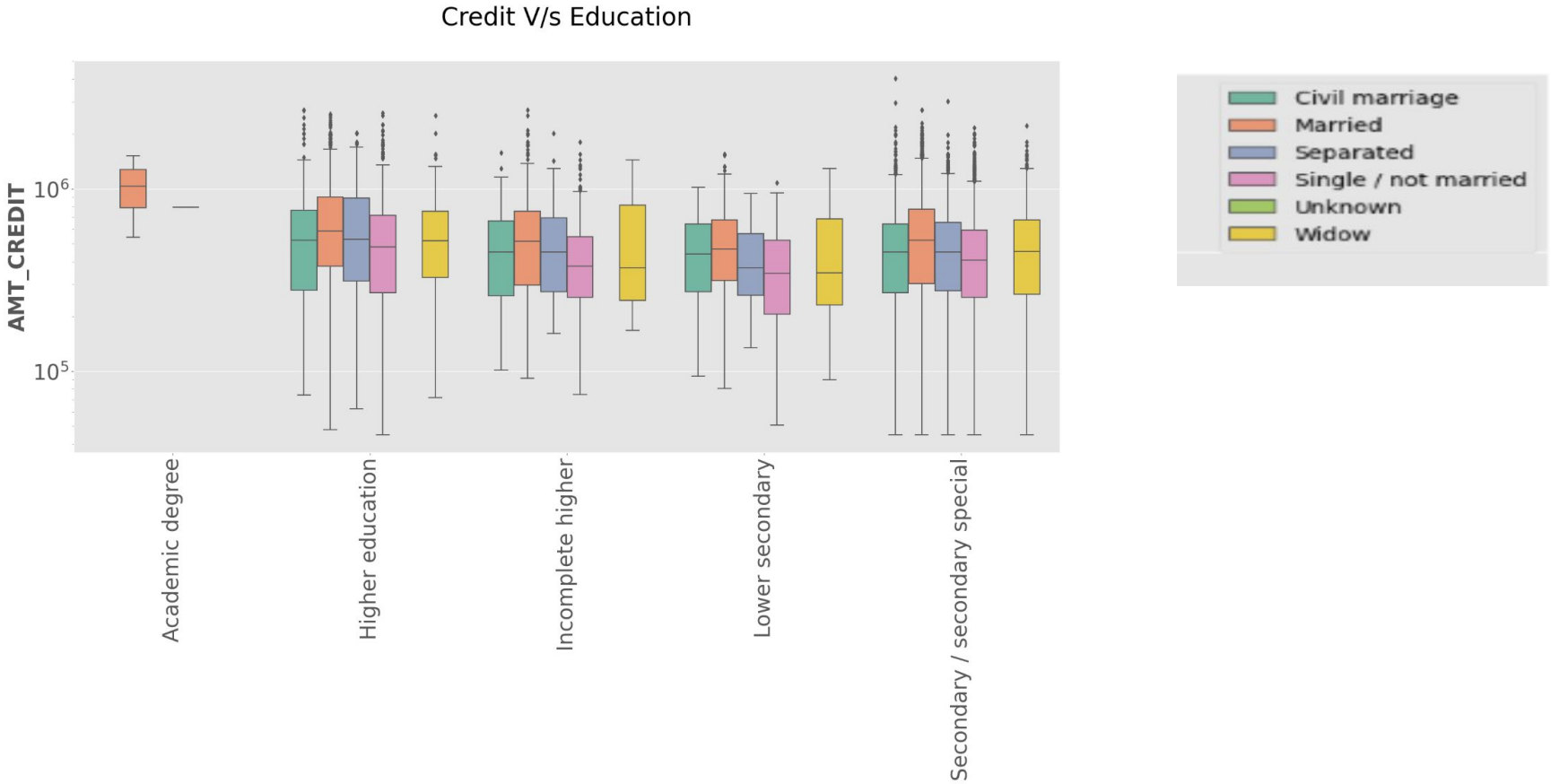
In [62]:

Slide Type

```
plt.figure(figsize=(25,10))
plt.yscale('log')                       #As the values are too large, it is convinient to use log for better analysis
plt.xticks(rotation = 90)


sns.boxplot(data =Target1, x='NAME_EDUCATION_TYPE',y='AMT_CREDIT',          #Boxplot w.r.t Data Target 0
            hue ='NAME_FAMILY_STATUS',orient='v',palette='Set2')


plt.legend( bbox_to_anchor=(1.5, 1),loc = 'upper right')            #Adjusting legend position
plt.title('Credit V/s Education',fontsize=35 )
plt.xlabel("NAME_EDUCATION_TYPE",fontsize= 30, fontweight="bold")
plt.ylabel("AMT_CREDIT",fontsize= 30, fontweight="bold")
plt.xticks(rotation=90, fontsize=30)
plt.yticks(rotation=360, fontsize=30)

plt.show()
```

Credit V/s Education

Box plot


Credit V/s Education
AMT_CREDIT
10^6
10^5
Academic degree
Higher education
Incomplete higher
Lower secondary
Secondary / secondary special
NAME_EDUCATION_TYPE
Civil marriage
Married
Separated
Single / not married
Unknown
Widow

Observations:

1. Clients with different education types except Academic degree have higher outliers so they are more tend to have higher credit amount.
2. Clients who are widow and have academic degree tend to apply with higher credit amount.
3. The client who are married and have higher education and secondary education tends to apply for higher loan amount.
4. The income amount for Married clients with an academic degree is much lesser as compared to others.

5. (Defaulter) Clients have relatively less income as compared to Non-defaulters.

Correlation between variables using pair plots

In [59]:

Slide Type

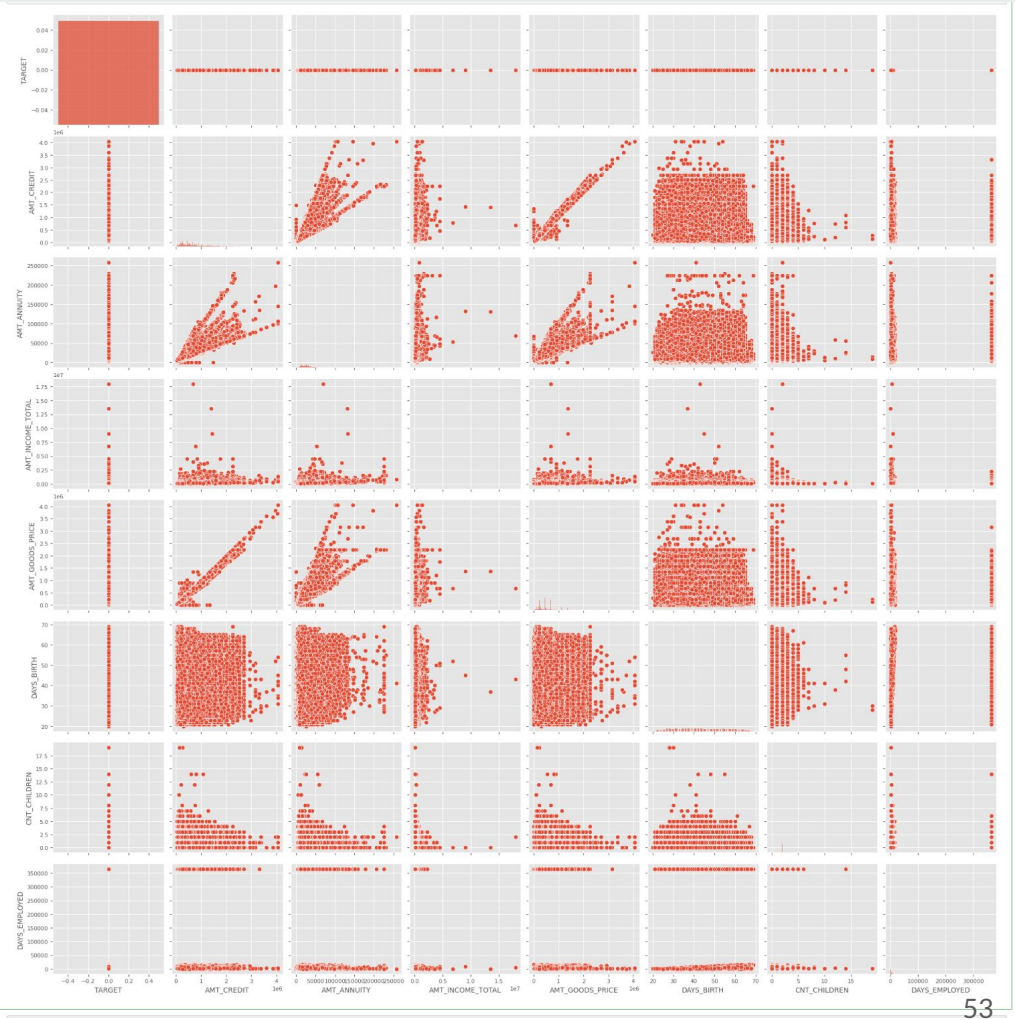
```
pair = Target0[['TARGET','AMT_CREDIT', 'AMT_ANNUITY', 'AMT_INCOME_TOTAL', 'AMT_GOODS_PRICE', 'DAYS_BIRTH',
                'CNT_CHILDREN','DAYS_EMPLOYED']].fillna(0)
sns.pairplot(pair)
plt.xticks(rotation=90, fontsize=10)
plt.yticks(rotation=360, fontsize=10)

plt.show()
```

In [55]:

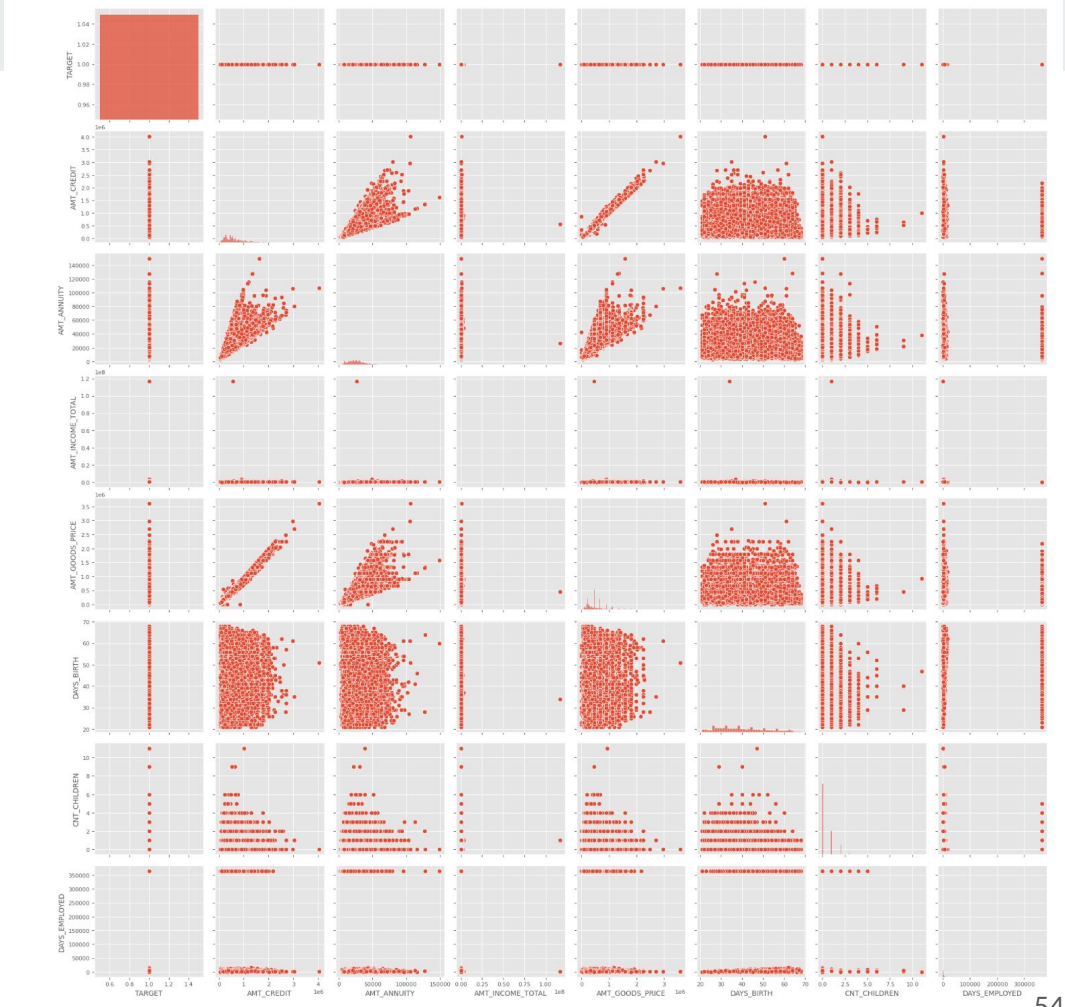
Slide Type

```
pair = Target1[['TARGET','AMT_CREDIT', 'AMT_ANNUITY', 'AMT_INCOME_TOTAL', 'AMT_GOODS_PRICE', 'DAYS_BIRTH','CNT_CHILDREN','DAYS_EM
sns.pairplot(pair)
plt.xticks(rotation=90, fontsize=10)
plt.yticks(rotation=360, fontsize=10)
plt.show()
```

Pair Plots for Defaulters



Pair Plots for Non-Defaulters



Observations

- AMT_CREDIT(Loan Amount) and AMT_GOODS_PRICE(Asset Price) are highly correlated variables for both defaulters and non – defaulters. So as the home price increases the loan amount also increases
- AMT_CREDIT(Loan Amount) and AMT_ANNUITY (EMI) are highly correlated variables for both defaulters and non – defaulters. So as the home price increases the EMI amount also increases which is logical
- All three variables AMT_CREDIT(Loan Amount), AMT_GOODS_PRICE(Asset Price) and AMT_ANNUITY(EMI) are highly correlated for both defaulters and non-defaulters, which might not give a good indicator for defaulter detection

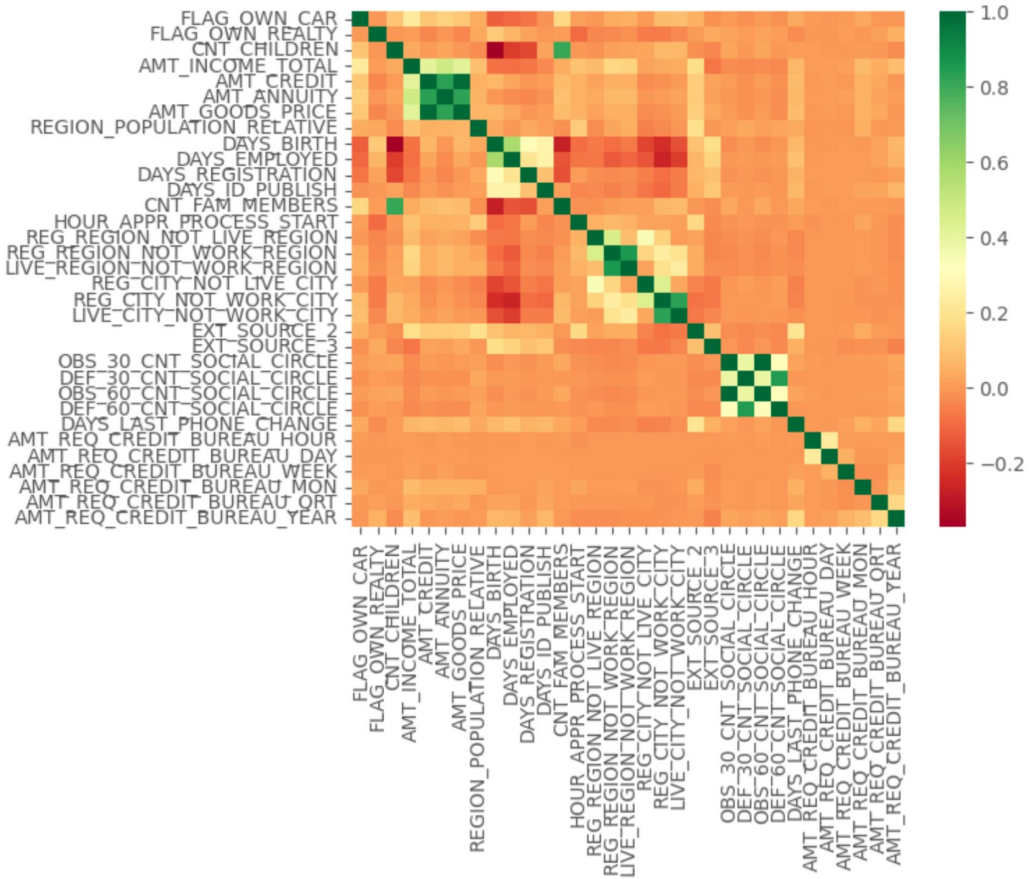
Correlation between variables under non-defaulters dataframe

In [122]:

Slide Type

```
#Correlation between variables under Target0(Defaulters)
corr0=df1.iloc[0:,2:]

t0=corr0.corr(method='spearman')   # t0 - Corelations distibuted according rank wise for target 0

dataplot = sns.heatmap(t0, cmap="RdYlGn")

plt.show()
```

Heat Map for the Non-Defaulters (Target0)



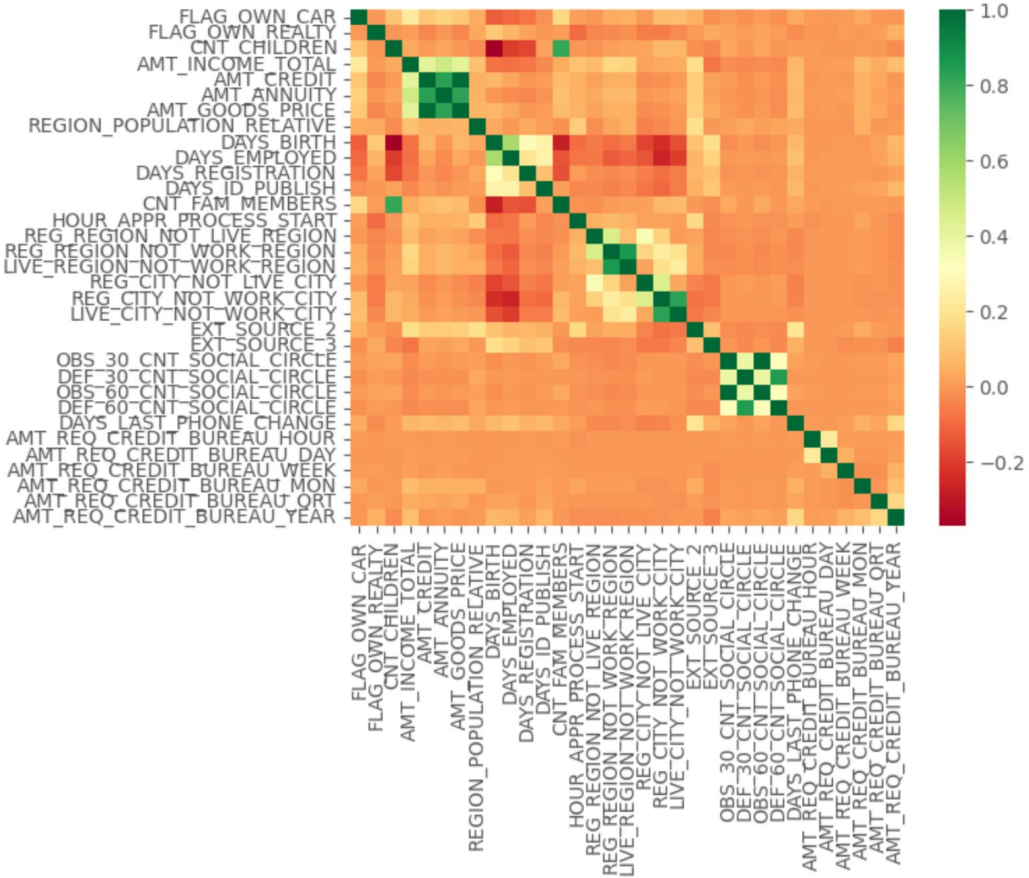
Correlation between variables under defaulters dataframe

In [124]:

Slide Type

```
corr1=df1.iloc[0:,2:]

t1=corr1.corr(method='spearman')   # t1 - Corelations distibuted according rank wise for target 1

dataplot1 = sns.heatmap(t1, cmap="RdYlGn")

plt.show()
```

Heat Map for the Defaulters (Target1)

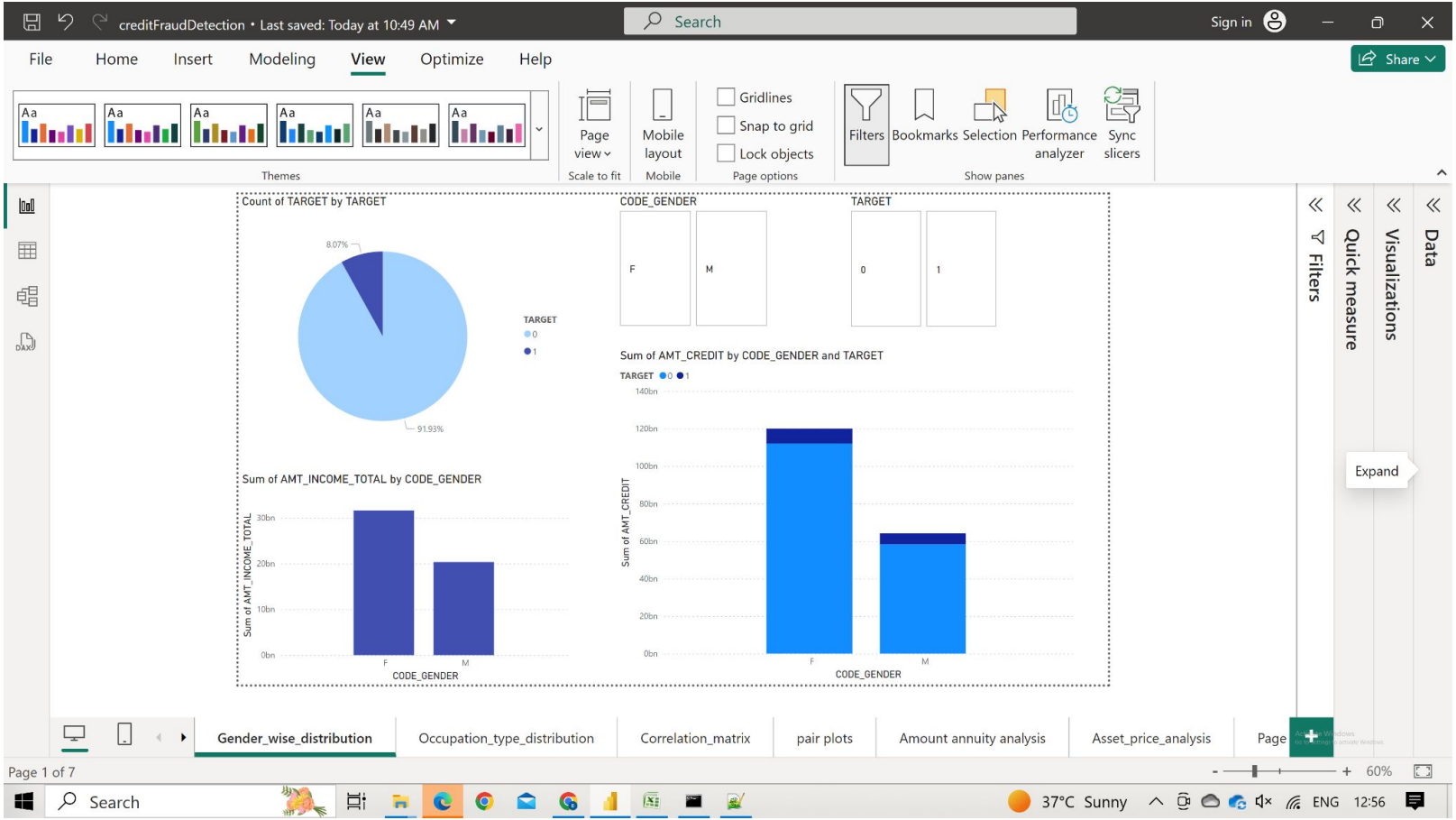


Observations

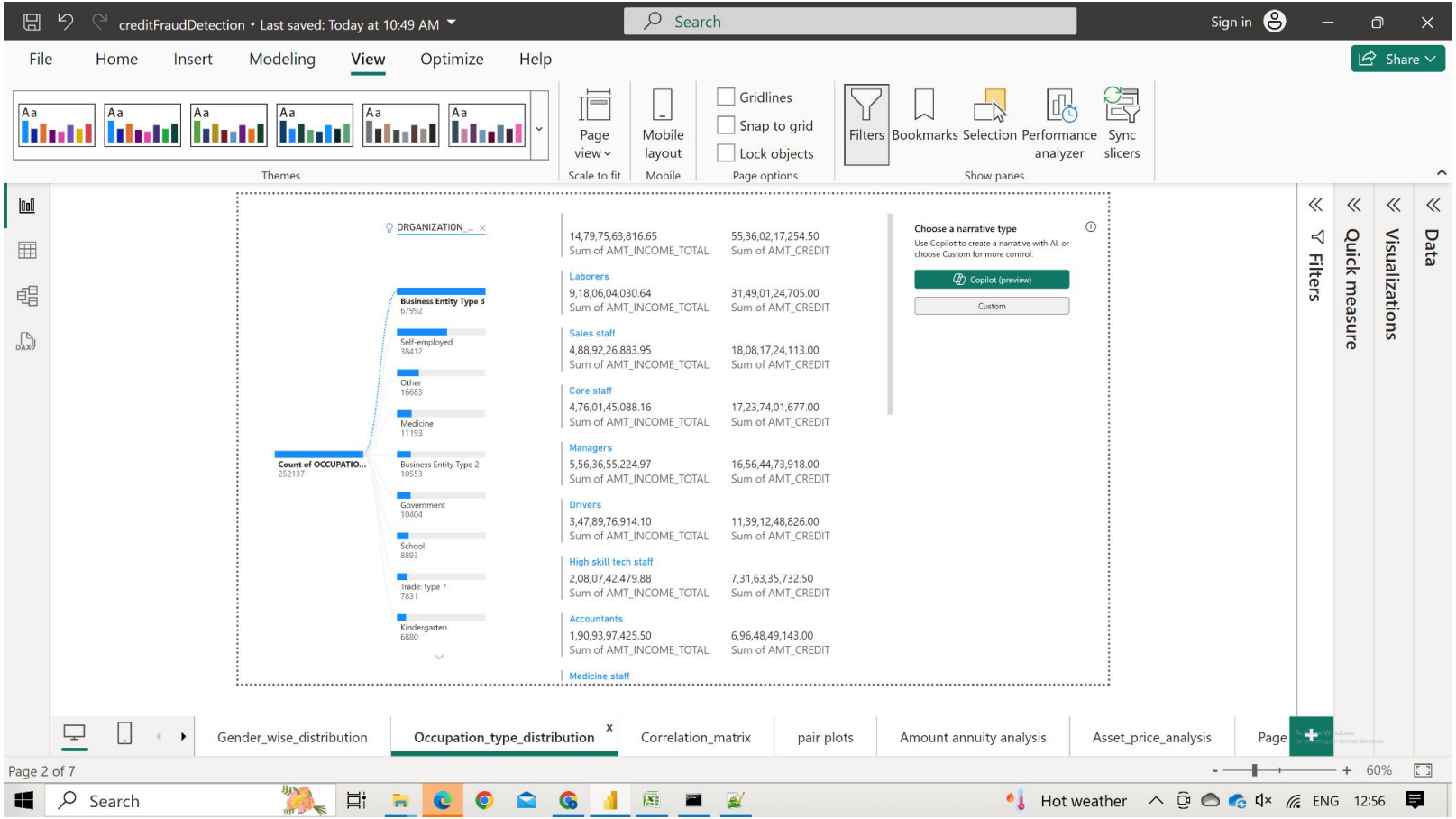
- The client’s permanent address does not match the contact address are having fewer children.
- The client’s permanent address does not match the work address are having fewer children.
- The above two points shows the discrepancy provided by applicants in the defaulters heat map
- fewer children clients have in a densely populated area.
- `AMT_CREDIT` is higher in a densely populated area.
- `AMT_INCOME_TOTAL` is also higher in a densely populated area.
- `AMT_CREDIT` is inversely proportional to the `DAYS_BIRTH` , peoples belong to the low-age group taking high Credit amount and vice-versa
- `AMT_CREDIT` is inversely proportional to the `CNT_CHILDREN`, means the Credit amount is higher for fewer children count clients have and vice-versa.
- `AMT_INCOME_TOTAL` is inversely proportional to the `CNT_CHILDREN`, means more income for fewer children clients have and vice-versa.

Note : For Taget0 and Target1 observations are very much similar

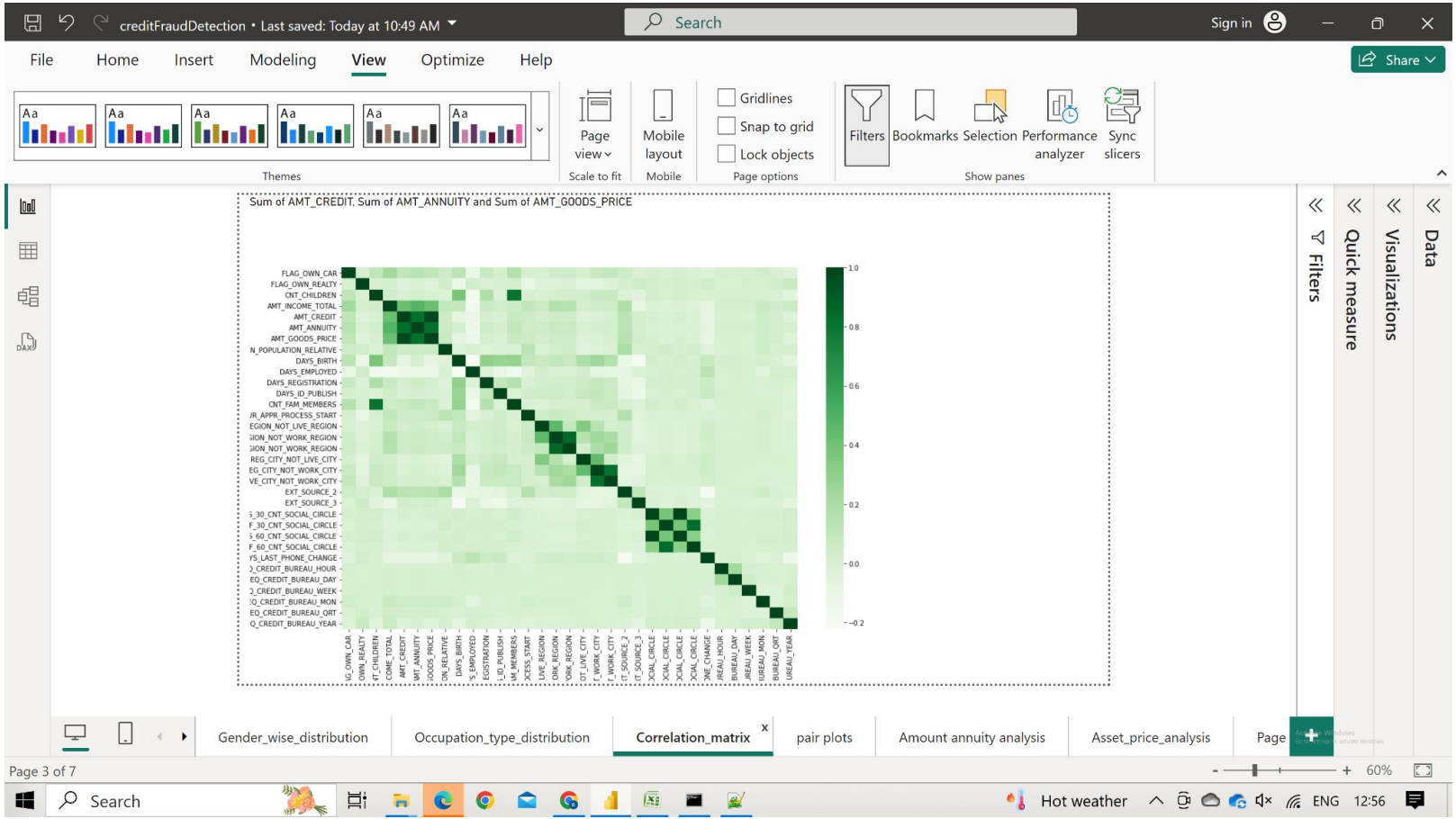
Power Bi Visualization - Gender wise distribution



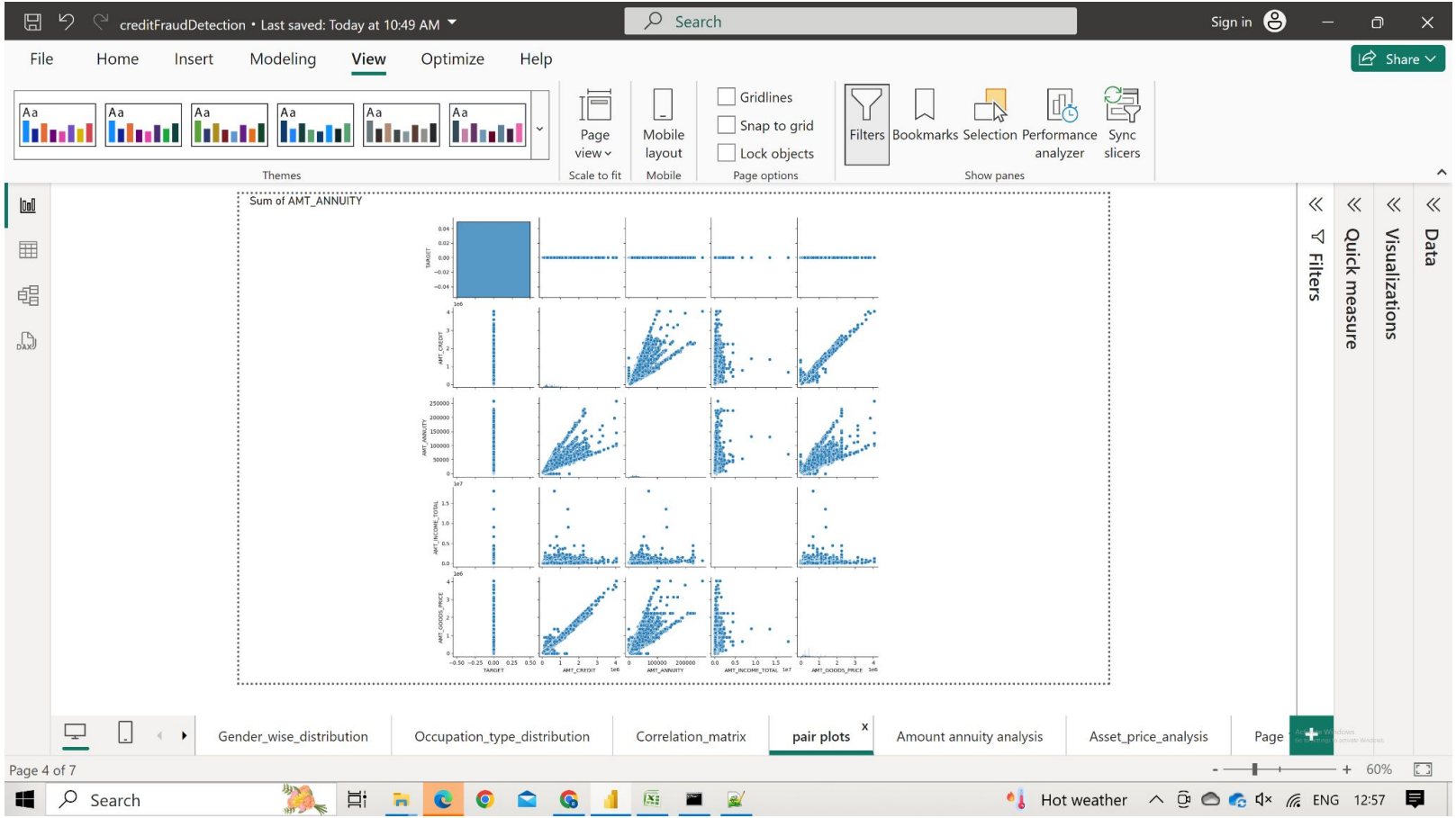
Power Bi Visualization - Occupation type distribution



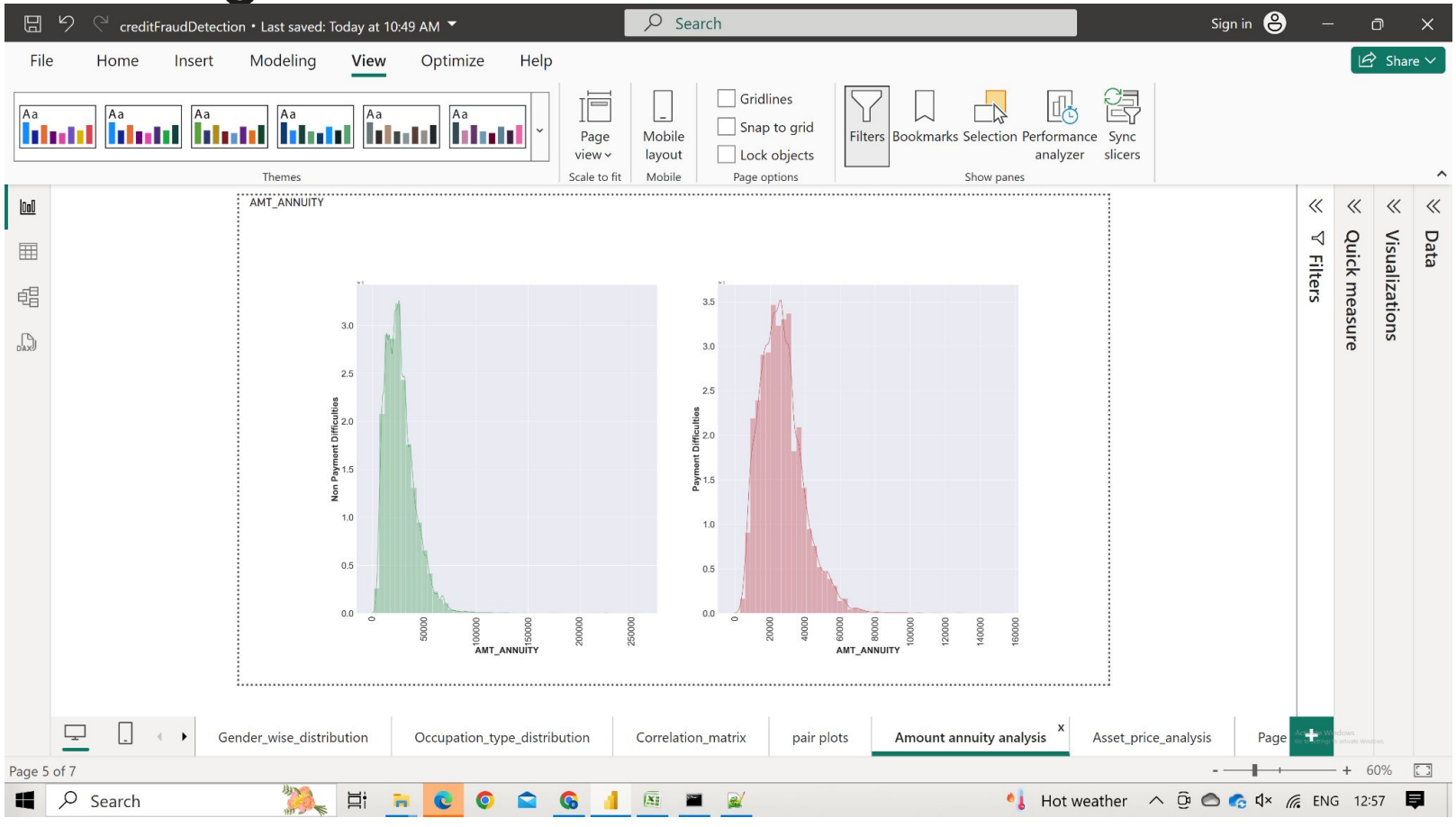
Power Bi Visualization - Correlation matrix



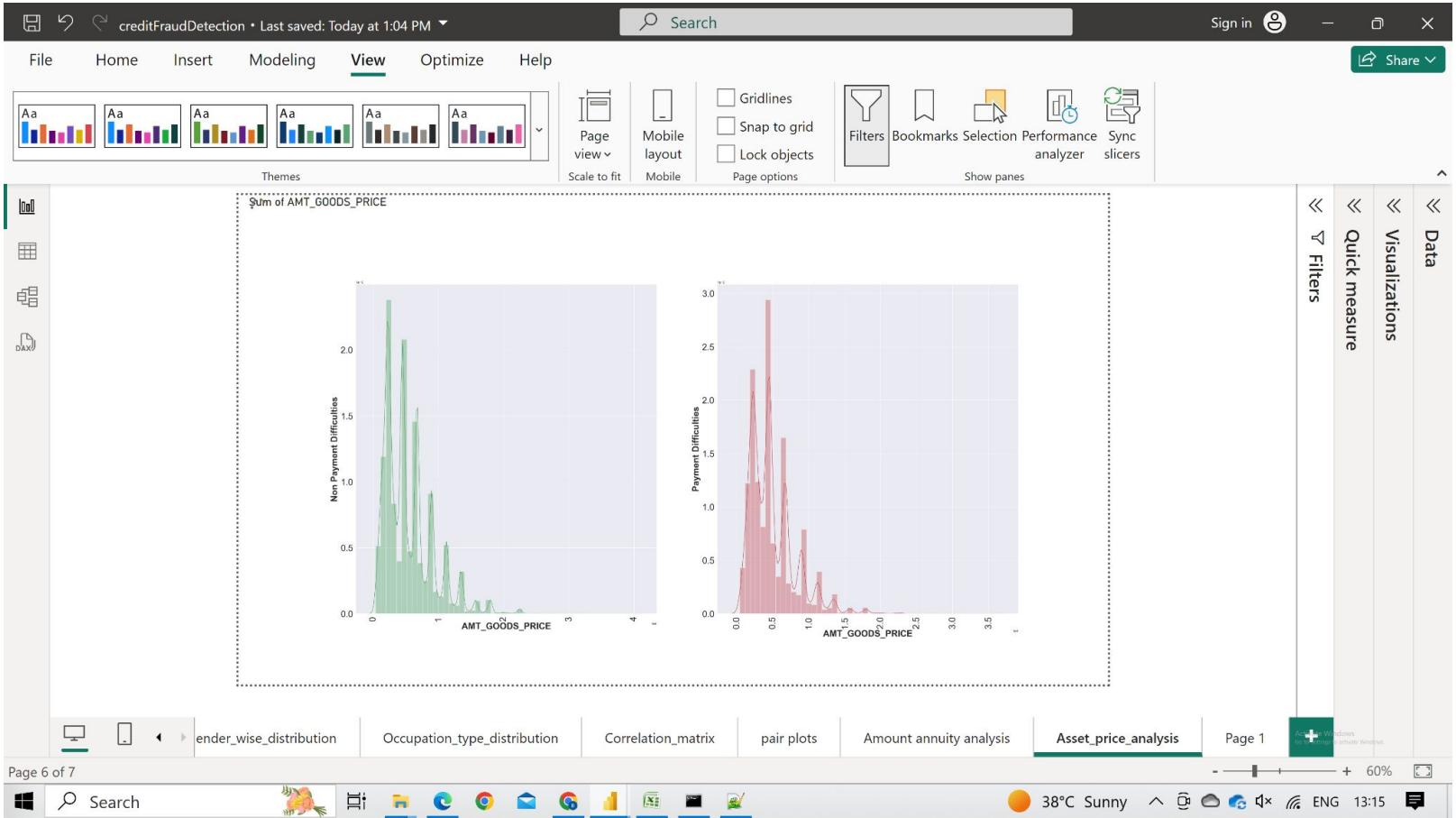
Power Bi Visualization - Pair plot analysis



Power Bi Visualization - Amount annuity distribution with target



Power Bi Visualization - Amount goods price distribution with target



Final Conclusion and observations:

1. For CNT_FAM_MEMBERS , we can say that most of the clients have 4 family members.
2. CNT_CHILDREN have outlier values having children more than 4-5. And on an average 2 childrems are there for applicants.
3. According to the outlier detection most of the clients are having own car(FLAG_OWN_CAR) and realstate(FLAG_OWN_REALTY).
4. In the dataset we have one column known as Target which describes according to the past history clients defaulter's record its boolean in type and the fact is according to this data arround 8.07% clients are defaulters and 91.92% are non-defaulters.
5. According to the analysis it is found that females have applied more as compared to the male clients.
6. In the range of Defaulters there are 57% females who defaulted and 42% males who defaulted.
7. According to the analysis Middle Age(35-60 years) the group seems to applied higher than any other age group for loans in the case of Defaulters as well as Non-defaulters.
8. Majority of defaulters are having high family and children count. As children and family count are way proportional.
9. As per the analysis it is found that female clients are more educated as compared to male clients and having majorityof higher education and secondary education.
10. Clients with higher education and secondary education are more often to take loan.
11. Clients with incomplete education or academic degree are more pron to default.
12. The applicant who are married and having higher and secondary education have higher income and outliers

Observations continued..

1. Clients who are widow and have academic degree tend to apply with higher credit amount.
2. The client who are married and have higher education and secondary education tends to apply for higher loan amount.
3. Defaulters income is less as compared to non-defaulters.
4. AMT_CREDIT(Loan Amount), AMT_GOODS_PRICE(Asset Price) and AMT_ANUITY(EMI) are highly correlated variables for both defaulters and non-defaulters.
5. AMT_INCOME_TOTAL is inversely proportional to the CNT_CHILDREN, means more income for fewer children clients have and vice-versa.
6. AMT_CREDIT is inversely proportional to the CNT_CHILDREN, means the Credit amount is higher for fewer children count clients have and vice-versa.
7. AMT_CREDIT is inversely proportional to the DAYS_BIRTH , peoples belong to the low-age group taking high Credit amount and vice-versa
8. AMT_INCOME_TOTAL is also higher in a densely populated area.
9. AMT_CREDIT is higher in a densely populated area.
10. fewer children clients have in a densely populated area.
11. The client’s permanent address does not match the contact address are having fewer children.
12. The client’s permanent address does not match the work address are having fewer children.

Final checkpoints to verify a client's potential

There should be a filtering process which should govern some set of checkpoints to examine the potential of a client:

1. Check education level and the income status of the client. Clients having higher or secondary education tends to take high loan amount.
2. The targeted age group which is taking high loan amount is middle age i.e; 35-60 years. So they can be a good lead.
3. The clients who are having high family member or children counts are more pron to default. So we can take account of income and education at the time of loan sanction.
4. According to the analysis the Clients who are highly educated and are married can be a potential customer.
5. Clients with higher income tends to take high loan amount and are having matching emi with high asset price.
6. Clients with business and private sector tends to apply for high amount of loan.
7. Clients with lower education status, low income, high family and children count, low availability of own assets like car or realty and are in senior age category are more prone to be in defaulters list.

Thank You