EDA: Bank Loan Default Risk Analysis for Machine Learning

Kanishk Kumar

April 2, 2021

1. Introduction

In this report, I will detail the motivation and steps that led to the completion of this project. This is my final project for the Exploratory Data Analysis for Machine Learning course offered by IBM on Coursera. The codes used in this project can be found here.

2. Description of the Data

In this section, I will briefly explain the choice of the dataset and summarize its key attributes. I have always wanted to work with data related to finance. As a resident in India, my credit score determines a lot about my financial stability. Therefore, I have always had an interest in learning how a credit score is determined and what machine learning models are employed to provide a more accurate picture of one's financial position. However, the scope of this project is only to perform Exploratory Data Analysis (EDA).

2.1 Data Selection

While I am sure there is plenty of credit risk data available on the internet, I have decided to go with the Loan Defaulter dataset found on Kaggle by user Gaurav Dutta because of the number of records and features available. After downloading the data as a CSV and importing it in a Pandas DataFrame, I could get a view of it.

```
application_data.columns
```

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY',
       ...
       'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',
       'FLAG_DOCUMENT_21', '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', length=122)
```

We can see that the DataFrame has 122 columns. We will not be using all 122 features in this project and fortunately, the Kaggle repository also contains a key as to what each feature represents. I have made the decision to use only a subset of them and they are listed below.

SK_ID_CURR: ID of loan in our sample.

TARGET: Target variable (1 - client 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)

NAME_CONTRACT_TYPE: Identification if loan is cash or revolving.

CODE_GENDER: Gender of the client.

FLAG_OWN_CAR: Flag if the client owns a car.

FLAG_OWN_REALTY: Flag if the client owns property.

CNT_CHILDREN: Number of children the client has.

AMT_INCOME_TOTAL: Income of the client.

AMT_CREDIT: Credit amount of the loan.

AMT_ANNUITY: Loan annuity.

DAYS_BIRTH: Client's age in days at the time of application.

2.2 Summary of Attributes

I verify that the data types of each column is consistent with what they represent.

```
print('There are ' + str(len(data_subset)) + ' rows in the DataFrame.')
display(data_subset.dtypes)
```

```
There are 307511 rows in the DataFrame.
SK_ID_CURR              int64
TARGET                  int64
NAME_CONTRACT_TYPE     object
CODE_GENDER            object
FLAG_OWN_CAR           object
FLAG_OWN_REALTY        object
CNT_CHILDREN            int64
AMT_INCOME_TOTAL      float64
AMT_CREDIT            float64
AMT_ANNUITY           float64
DAYS_BIRTH              int64
dtype: object
```

Fortunately, it seems that no changes are needed. The first five rows of the DataFrame now looks as such:

```
print(data_subset.head())
```

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

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

   DAYS_BIRTH   AGE  CODE_GENDER_F  CODE_GENDER_M  CODE_GENDER_XNA  \
0       -9461  25.9            0.0            1.0              0.0
1      -16765  45.9            1.0            0.0              0.0
2      -19046  52.2            0.0            1.0              0.0
3      -19005  52.1            1.0            0.0              0.0
4      -19932  54.6            0.0            1.0              0.0

   FLAG_OWN_CAR_N  FLAG_OWN_CAR_Y  FLAG_OWN_REALTY_N  FLAG_OWN_REALTY_Y
0             1.0             0.0                0.0                1.0
1             1.0             0.0                1.0                0.0
2             0.0             1.0                0.0                1.0
3             1.0             0.0                0.0                1.0
4             1.0             0.0                0.0                1.0
```

Note: In order to show all the columns without the need to scroll, I used 'print' to show the DataFrame.

```
rows, col = data_subset.shape
print('There are ' + str(rows) + ' rows and ' + str(col) + ' columns in data_subset.')
```

```
There are 307511 rows and 11 columns in data_subset.
```

There are 307,511 rows in the DataFrame which makes it a large enough dataset to conduct analysis.

3. Initial Plan for Data Exploration

In this section, I will clean the data as per the findings of the previous section. Some features will have to be engineered to better fit future models. Before proceeding further in the data analysis, we need to make a plan on how to perform data exploration and how to identify outliers if any.

I propose that we follow the following steps:
1. Identify the number of unique values in each column.
2. Verify if null values are present.
3. Plot a Correlogram to visualize relationships between the continuous variables.

3.1 Identify Unique Values

In order to do so, we use the function nunique() provided by the DataFrame class.

```
data_subset.nunique(axis=0)
```

```
SK_ID_CURR              307511
TARGET                       2
NAME_CONTRACT_TYPE           2
CODE_GENDER                  3
FLAG_OWN_CAR                 2
FLAG_OWN_REALTY              2
CNT_CHILDREN                15
AMT_INCOME_TOTAL          2548
AMT_CREDIT                5603
AMT_ANNUITY              13672
DAYS_BIRTH               17460
dtype: int64
```

Observations:
• There are 3 unique values in the Gender Field which suggests wrong data.
• The Count of Children field has 15 unique values which suggests outliers.

3.2 Check for Null Values

To check for null values, we use the isnull() function provided by the DataFrame class and sum them to get a count of how many null values are present in each column.

```
data_subset.isnull().sum()
```

```
SK_ID_CURR                0
TARGET                    0
NAME_CONTRACT_TYPE        0
CODE_GENDER               0
FLAG_OWN_CAR              0
FLAG_OWN_REALTY           0
CNT_CHILDREN              0
AMT_INCOME_TOTAL          0
AMT_CREDIT                0
AMT_ANNUITY              12
DAYS_BIRTH                0
dtype: int64
```

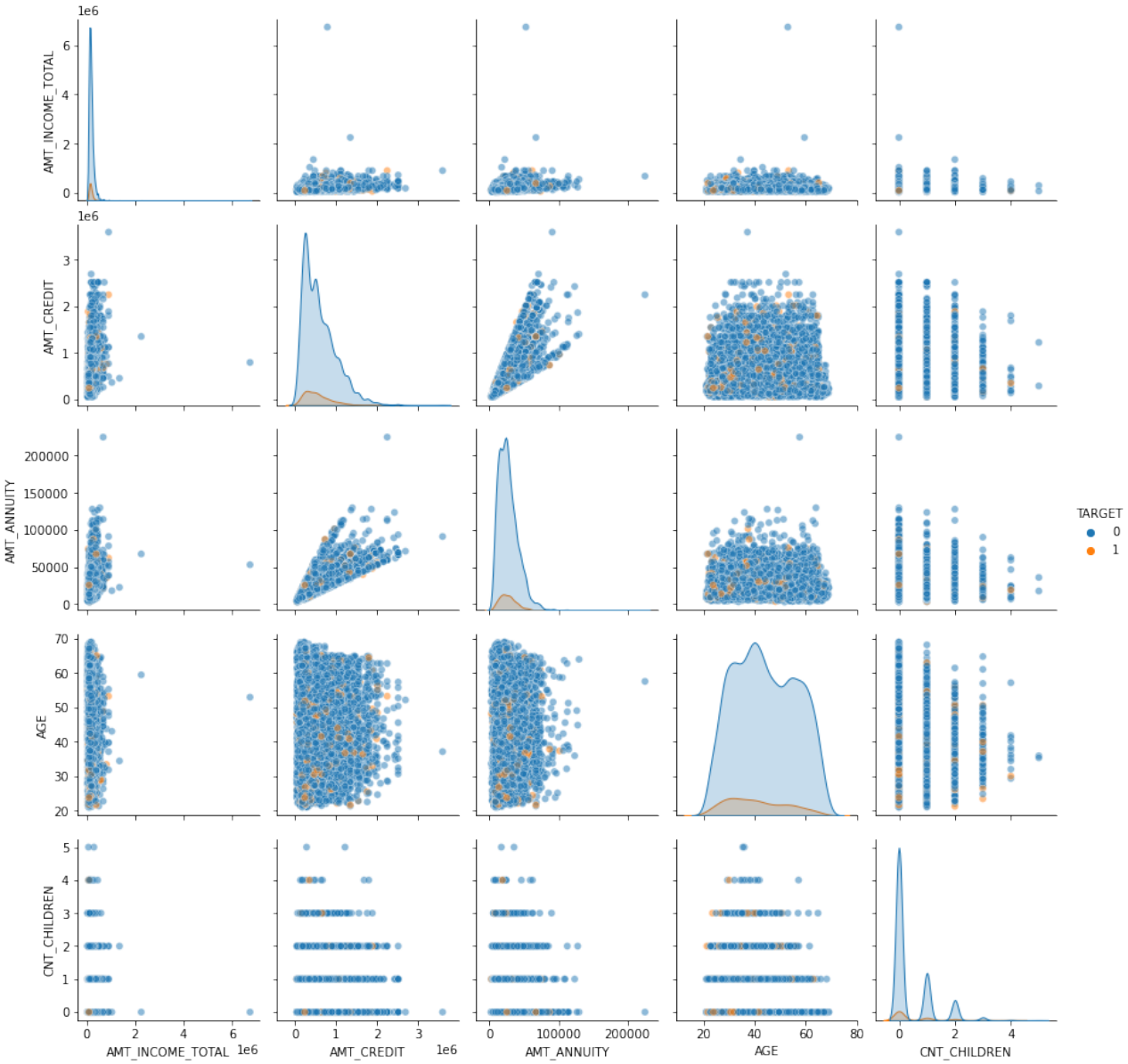
Observations:
• There are 12 null values in the AMT_ANNUITY column.

3.3 Correlogram

I plot a correlogram to identify the relationship if any among the continuous variables. I plot only a random sample of 10,000 records for two reasons: To make the graphs more readable and to make the plotting faster.

```
sns.pairplot(cleaned_data[['AMT_INCOME_TOTAL',
                           'AMT_CREDIT',
                           'AMT_ANNUITY',
                           'AGE',
                           'CNT_CHILDREN',
                           'TARGET']].sample(10000),
             hue='TARGET',
             plot_kws=dict(alpha=0.5))
```

I use the *pairplot()* function from the Seaborn python library. I also create a distinction between the two different groups of people, namely Target = 0 and Target = 1, representing people with no repayment difficulties and people with repayment difficulties respectively.



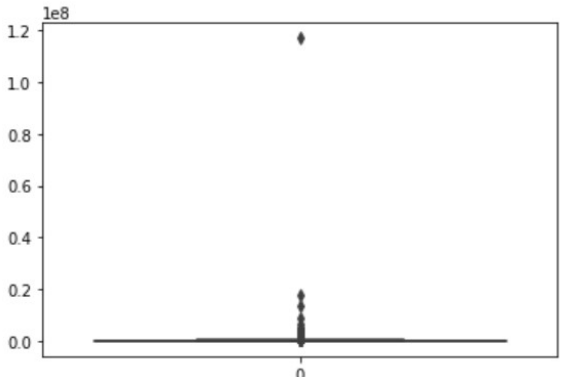
Observations:

- There are extreme outliers in the AMT_INCOME_TOTAL column.
- There are outliers in the CNT_CHILDREN column.

To better understand the outliers, I plot boxplots for the 2 columns.

```
sns.boxplot(data=cleaned_data['AMT_INCOME_TOTAL'])
```

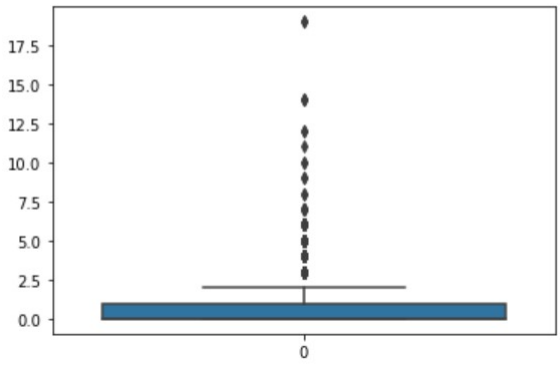
```
<matplotlib.axes._subplots.AxesSubplot at 0x1af144ed748>
```



Boxplot for AMT_TOTAL_INCOME

```
sns.boxplot(data=cleaned_data['CNT_CHILDREN'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1af14886c18>
```



Boxplot for CNT_CHILDREN

4. Data Cleaning and Feature Engineering

In this section, I address the issues that I found during the exploration phase. Some features will have to be engineered to better fit the future models.

4.1 Addressing Null Values

There are 12 null values in the AMT_ANNUITY column. We could delete them but since at this stage I do not know what model will be used, I will instead replace the null values with -1.

After replacing the null values in the problematic column, there are no null values in the data.

```
data_subset = data_subset.fillna(-1)
data_subset.isnull().sum() # cross-check
```

```
SK_ID_CURR            0
TARGET                0
NAME_CONTRACT_TYPE    0
CODE_GENDER           0
FLAG_OWN_CAR          0
FLAG_OWN_REALTY       0
CNT_CHILDREN          0
AMT_INCOME_TOTAL      0
AMT_CREDIT            0
AMT_ANNUITY           0
DAYS_BIRTH            0
dtype: int64
```

4.2 Addressing Third Gender

While exploring, it was found that there are 3 unique values in the CODE_GENDER field. To understand what is going on, I will display the unique genders.

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

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

There are 4 rows where the gender of the client was not specified. I decide not to remove them because it is possible that those clients did not want to fill the gender category or that they did not identify with the Gender Binary System. This could be of some value to future models.

4.3 One Hot Encode Categorical Values

Since I do not know what model will be used in the future, I will encode the categorical values that benefits most models. Columns to undergo One Hot Encoding are: *CODE_GENDER, FLAG_OWN_CAR, FLAG_OWN_REALTY.* The *sklearn.preprocessing* library has the *OneHotEncoder()* function that can handle this process.

```
enc = OneHotEncoder(sparse = False)
encoded_features = enc.fit_transform(data_subset[['CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY']])

encoded_featuresNames = enc.get_feature_names(['CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY'])

encoded_df = pd.DataFrame(data = encoded_features, columns = encoded_featuresNames)
encoded_df
```

And the result is as follows:

	CODE_GENDER_F	CODE_GENDER_M	CODE_GENDER_XNA	FLAG_OWN_CAR_N	FLAG_OWN_CAR_Y	FLAG_OWN_REALTY_N	FLAG_OWN_REALTY_Y
0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
1	1.0	0.0	0.0	1.0	0.0	1.0	0.0
2	0.0	1.0	0.0	0.0	1.0	0.0	1.0
3	1.0	0.0	0.0	1.0	0.0	0.0	1.0
4	0.0	1.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...
307506	0.0	1.0	0.0	1.0	0.0	1.0	0.0
307507	1.0	0.0	0.0	1.0	0.0	0.0	1.0
307508	1.0	0.0	0.0	1.0	0.0	0.0	1.0
307509	1.0	0.0	0.0	1.0	0.0	0.0	1.0
307510	1.0	0.0	0.0	1.0	0.0	1.0	0.0

We merged the new DataFrame with our dataset.

4.4 Creating a New Column to Calculate AGE

The dataset contains a column called DAYS_BIRTH which holds the number of days since birth of the client at the time of application. I will not drop this column but I will add another column called AGE which will calculate the age of the client and store it.

This will alleviate the workload of future data scientists who work on this data.

```
data_subset['AGE'] = data_subset['DAYS_BIRTH'] * -1 / 365
data_subset = data_subset.round(1)
data_subset[['DAYS_BIRTH', 'AGE']]
```

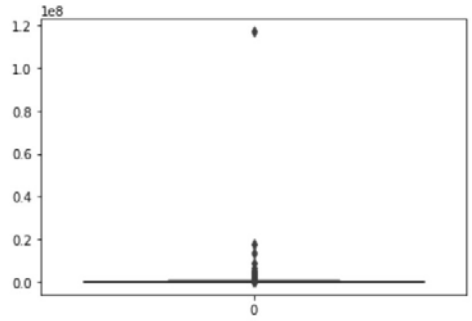
	DAYS_BIRTH	AGE
0	-9461	25.9
1	-16765	45.9
2	-19046	52.2
3	-19005	52.1
4	-19932	54.6

4.5 Addressing Outliers in the Income Column (Winsorization)

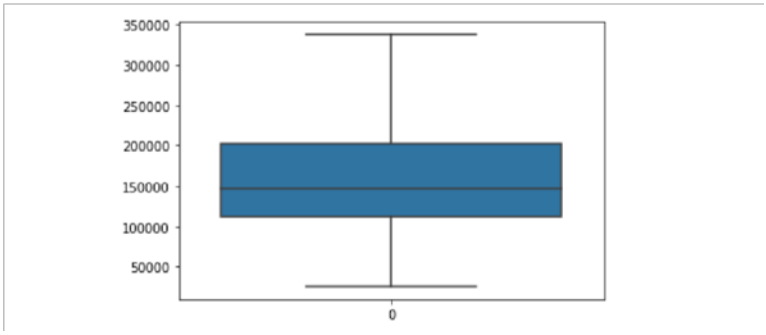
From the exploration phase, I noticed that there were extreme outliers in the *AMT_INCOME_TOTAL* column. While there are many ways to deal with them, I decided that Winsorization is the best approach here, more specifically, to transform all data above the 95 percentile to its closest acceptable value.

```
cleaned_data['AMT_INCOME_TOTAL'] = winsorize(cleaned_data['AMT_INCOME_TOTAL'], limits=[None, 0.05])
```

Comparing the data before and after Winsorization:



Before Winsorization



After Winsorization

4.6 Addressing Outliers in the CNT_CHILDREN Column

In this case, I removed records that are more than 3 standard deviations from the mean. This is because a lot of the data is erroneous and suggests that 20 year olds have over 10 children, an impossible feat.

```
z_scores = stats.zscore(cleaned_data['CNT_CHILDREN'])
z_scores
```

```
array([-0.57753784, -0.57753784, -0.57753784, ..., -0.57753784,
       -0.57753784, -0.57753784])
```

```
abs_z_scores = np.abs(z_scores)
abs_z_scores
```

```
array([0.57753784, 0.57753784, 0.57753784, ..., 0.57753784, 0.57753784,
       0.57753784])
```

```
filtered_entries = (abs_z_scores < 3)
filtered_entries
```

```
array([ True,  True,  True, ...,  True,  True,  True])
```

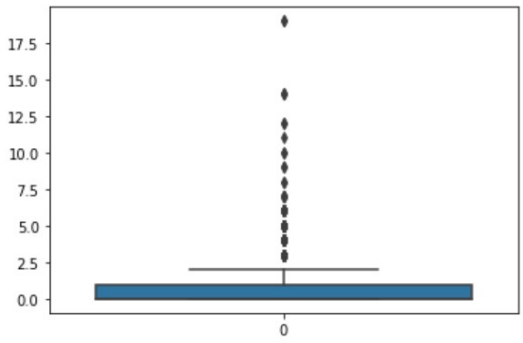
```
print('There are ' + str(len(cleaned_data.index)) + ' records before removing outliers.')
print('There are ' + str(len(cleaned_data[filtered_entries].index)) + ' records after removing outliers.')
print(str(len(cleaned_data.index) - len(cleaned_data[filtered_entries].index)) + ' records have been removed.')
sns.boxplot(data=cleaned_data[filtered_entries]['CNT_CHILDREN'])
```

```
There are 307511 records before removing outliers.
There are 303239 records after removing outliers.
4272 records have been removed.
```

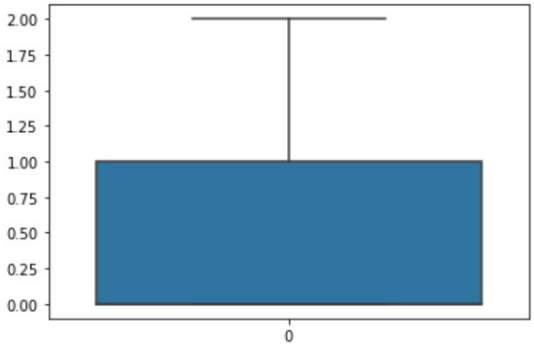
From this exercise, 4272 rows are dropped. This amount to a little over 1% of the total dataset.

Therefore, I deem it acceptable.

Results of Trimming outliers:



Before Trimming



After Trimming

This concludes the cleaning and feature engineering process. The data is now in a good shape for hypothesis statement and testing.

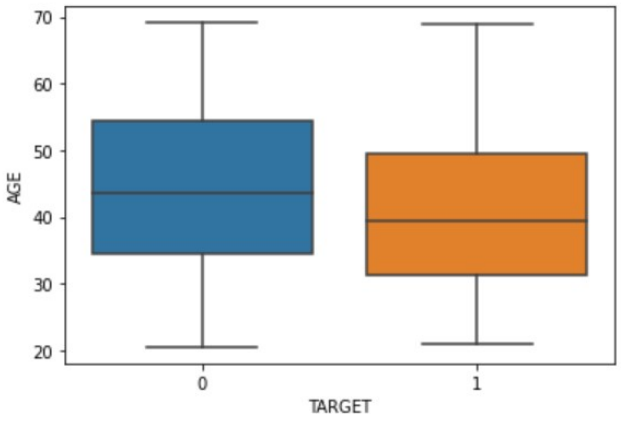
5. Findings and Insights

Finally, now I will analyze the cleaned data to gather insights and spot key findings about it. This will prove useful in formulating some good hypotheses about the data and will allow us to select tests to verify the veracity of these hypotheses.

I plot the boxplot to compare the age of Group 0 and Group 1.

```
sns.boxplot(data = cleaned_data[['TARGET', 'AGE']], y = 'AGE', x = 'TARGET')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1af1ccbeb38>
```



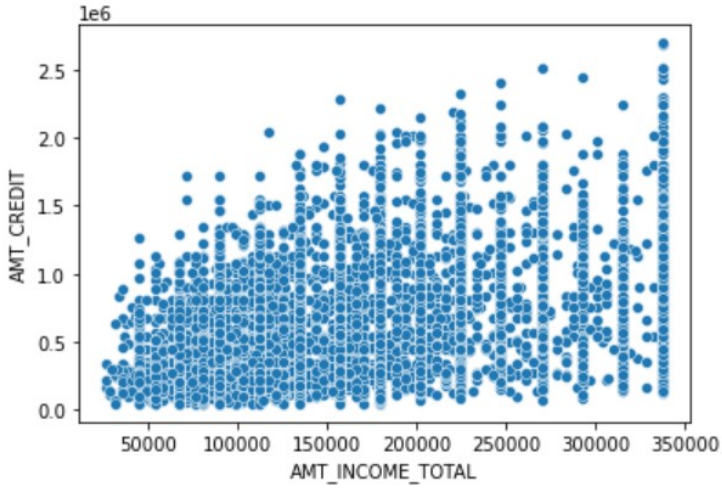
From this plot, we can see that the mean of Group 1 (people with repayment difficulties) is lower than that of Group 0 (people with no repayment difficulties).

Insight: This gives me reasonable doubts that younger people are less trustworthy with loans.

Other relevant findings from this dataset is that there seems to be a positive relationship between the income of clients and the line of credit that is advanced to them.

```
sns.scatterplot(data=cleaned_data[['AMT_INCOME_TOTAL',
                                   'AMT_CREDIT']].sample(10000),
                y = 'AMT_CREDIT',
                x='AMT_INCOME_TOTAL')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1af14088ac8>
```



The insights in this section closely match what I would expect to happen in real life. Therefore, I am equipped to start making hypotheses and test them.

6. Hypothesis Statement and Testing

This data offers an incredible opportunity to determine the most reliable groups to lend money to. From the correlated graph above and the previous section, we can state a few hypotheses and test them.

6.1 Comparing Age Groups

In the real world, and from this dataset, we can form the assumption that younger people are to be less trusted with loans and credit lines. Let's test if this is true.

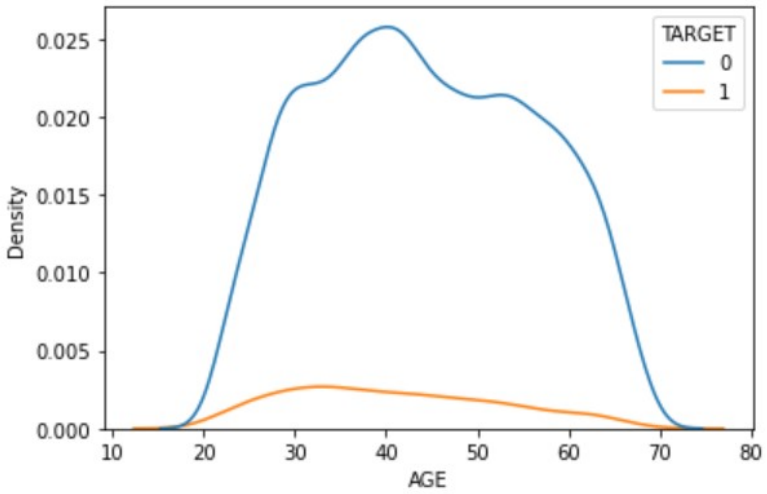
Hypothesis: h_0: $\mu_0 = \mu_1$; h_1: $\mu_0 \neq \mu_1$,
where,

μ_0: Mean age of Group 0
μ_1: Mean age of Group 1

I plot the distribution of the variables to determine what kind of test would be more appropriate.

```
sns.kdeplot(data=cleaned_data[['TARGET',
                               'AGE']].sample(10000),
            x = 'AGE',
            hue='TARGET')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1af1e151da0>
```



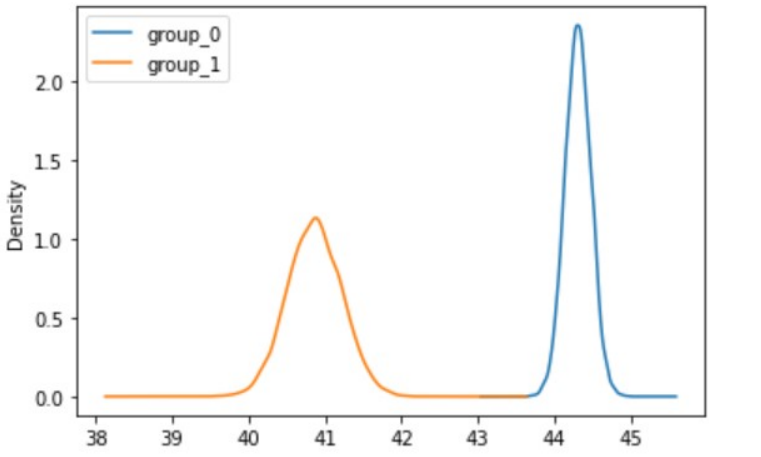
Since the age distributions in both groups do not follow a normal distribution, a non-parametric test could be used to determine if the difference in mean is significant.

However, I will take samples from each group and calculate their means to get a normal distribution for each group. This will allow us to perform the popular t-test instead of something like the Whitney Mann U Test.

After taking 10,000 sample means, the distributions are now Normal.

```
age_means.plot.kde()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1af1cb98e48>
```



From this plot, we can see a clear difference in mean from the two groups.

However, to be sure, we have to perform a t-test. Our significance level is 5%.

```
statistic, p_value = stats.ttest_ind(age_means['group_0'], age_means['group_1'])
significance_level = 0.05
if p_value < significance_level:
    print('The Null Hypothesis is rejected.')
else:
    print('The Null Hypothesis cannot be rejected')
```

```
The Null Hypothesis is rejected.
```

Conclusion: From the t-test, the Null Hypothesis is rejected and we have accepted the alternative hypothesis. We can therefore say that the difference in means of the two groups did not happen by chance and that truly young people are more likely to have loan repayment issues.

6.2 Comparing Income Groups

Similarly, I can compare the mean income of each group.

```
print('The mean income of Group 0 is ' + str(cleaned_data[cleaned_data['TARGET']==0]['AMT_INCOME_TOTAL'].mean()))
print('The mean income of Group 1 is ' + str(cleaned_data[cleaned_data['TARGET']==1]['AMT_INCOME_TOTAL'].mean()))
```

```
The mean income of Group 0 is 163078.8243509789
The mean income of Group 1 is 157155.14442941465
```

From the above data, I can form the following hypothesis:

h_0: $\mu_0 = \mu_1$; h_1: $\mu_0 \neq \mu_1$,

where,

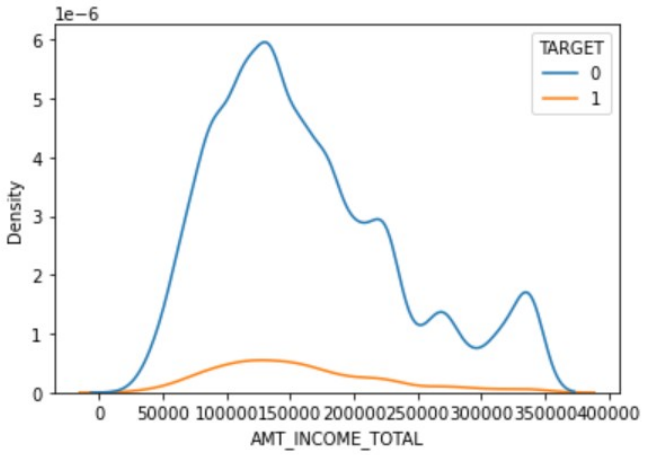
μ_0: Mean income of Group 0

μ_1: Mean income of Group 1

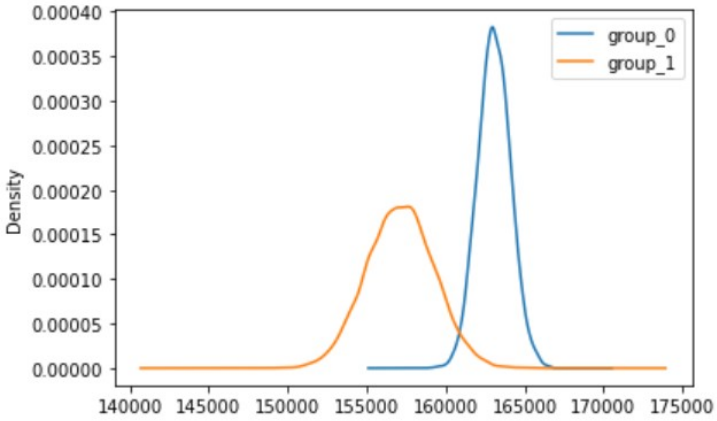
Plotting the distributions shows:

```
sns.kdeplot(data=cleaned_data[['TARGET',
                               'AMT_INCOME_TOTAL']].sample(10000),
            x = 'AMT_INCOME_TOTAL',
            hue='TARGET')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1af1ccd6cc0>
```



Again, as per the Central Limit Theorem, I will take the means of samples 10,000 times to normalize the data. The new distributions now look the figure on the right:



And performing the same t-test with a significance level of 5% yields the following results:

```
statistic, p_value = stats.ttest_ind(income_means['group_0'], income_means['group_1'])
significance_level = 0.05
if p_value < significance_level:
    print('The Null Hypothesis is rejected.')
else:
    print('The Null Hypothesis cannot be rejected')
```

```
The Null Hypothesis is rejected.
```

Conclusion: The null hypothesis is rejected in favor of the alternative Hypothesis. Therefore, as expected, people with higher incomes are less likely to struggle with repaying their loans compared to people with lower incomes.

6.3 Comparing Children Count

Similarly, we can compare the mean of children count in each group and determine if there is a significant difference between the mean of Group 0 and Group 1.

```
print('The mean children count of Group 0 is ' + str(cleaned_data[cleaned_data['TARGET']==0]['CNT_CHILDREN'].mean()))
print('The mean children count of Group 1 is ' + str(cleaned_data[cleaned_data['TARGET']==1]['CNT_CHILDREN'].mean()))
```

```
The mean children count of Group 0 is 0.3747521006444487
The mean children count of Group 1 is 0.41482210198393177
```

$\mu_0 = 0.37475$ $\mu_1 = 0.41482$

This time, the difference in mean seems tiny. But is it statistically different? I form the following hypothesis:

$h_0: \mu_0 = \mu_1$; $h_1: \mu_0 \neq \mu_1$,

where,

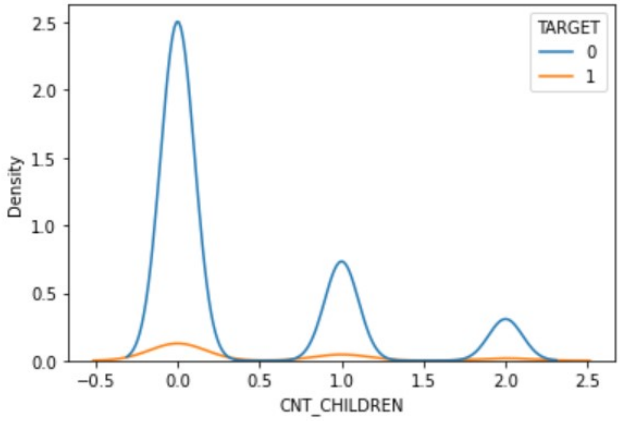
μ_0: Mean count of children of Group 0

μ_1: Mean count of children of Group 1

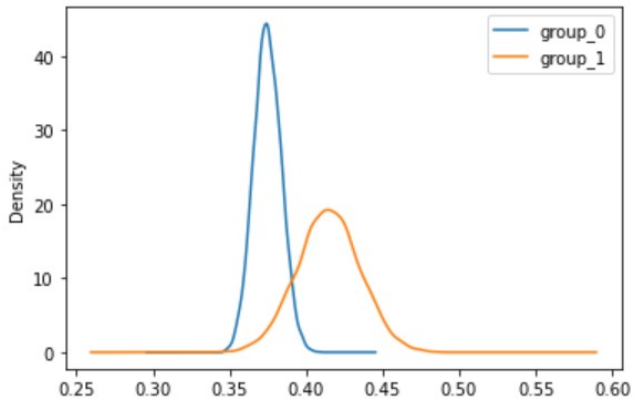
The distribution here cannot be normal. I will plot them to make sure.

```
sns.kdeplot(data=cleaned_data[['TARGET',
                               'CNT_CHILDREN']].sample(10000),
            x = 'CNT_CHILDREN',
            hue='TARGET')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1af1ce4f908>
```



Using the same technique as in the previous hypotheses, I will take the means of 10,000 samples to normalize the data so that I can perform a t-test with a significance level of 5%.



Performing the t-test at 5% significance level yields the following:

```
statistic, p_value = stats.ttest_ind(income_means['group_0'], income_means['group_1'], equal_var=False)
significance_level = 0.05
if p_value < significance_level:
    print('The Null Hypothesis is rejected.')
else:
    print('The Null Hypothesis cannot be rejected')
```

```
The Null Hypothesis is rejected.
```

Conclusion: Surprisingly, the null hypothesis is rejected. It seems that the difference in the number of children is significant and could be a factor that determines the ability of someone repaying their loans.

7. Conclusion and Next Steps

After performing an in-depth exploratory data analysis on this dataset, I can say that the data was in a good shape to begin with. There were however many extreme outliers that were dealt with by either trimming them or using a method called Winsorization.

Moreover, the hypotheses in the previous section hint that Age, Income and Number of Children are all factors that could potentially affect someone's ability to repay. Future data scientists working on this data can use that information to include in their models.

Finally, I would love if the author of this dataset could enlighten me on how the data was gathered in the first place. This would help me to understand the existence of some of the outliers and I would be able to make a better judgement on how to deal with them. The region where the data was collected would also be extremely useful so that future models can take the region into account as well.