

Credit EDA Assignment

BY – SHUBHAM SIDDHARTH



Data cleaning

BEFORE DOING ANYTHING WE NEED TO MAKE THAT RAW DATA INTO OPERATIONAL DATA.

FOR THAT WE NEED TO FOLLOW SOME STEPS ACCORDING TO DATA SET

WHICH INVOLVES MANY THINGS LIKE , TAKING NULL VALUES, EXTRA COLUMN, UNWANTED DATA AND MANY OTHER THINGS

Data cleaning steps

▶ Counting Null Values

- ▶ Removing Null values
- ▶ Removing extra columns
- ▶ Removing unwanted data
- ▶ Perform analysis then

Jupyter EDAAssignment 01 (2) Last Checkpoint: 27 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [57]: 1 #importing libraries needed for this assignment
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 from mpl_toolkits.mplot3d import Axes3D
7 from scipy.stats import boxcox
8 pd.set_option('display.max_columns', 100)
```

```
In [12]: 1 import warnings
2
3 warnings.filterwarnings('ignore')
```

Data Loading for cleanup

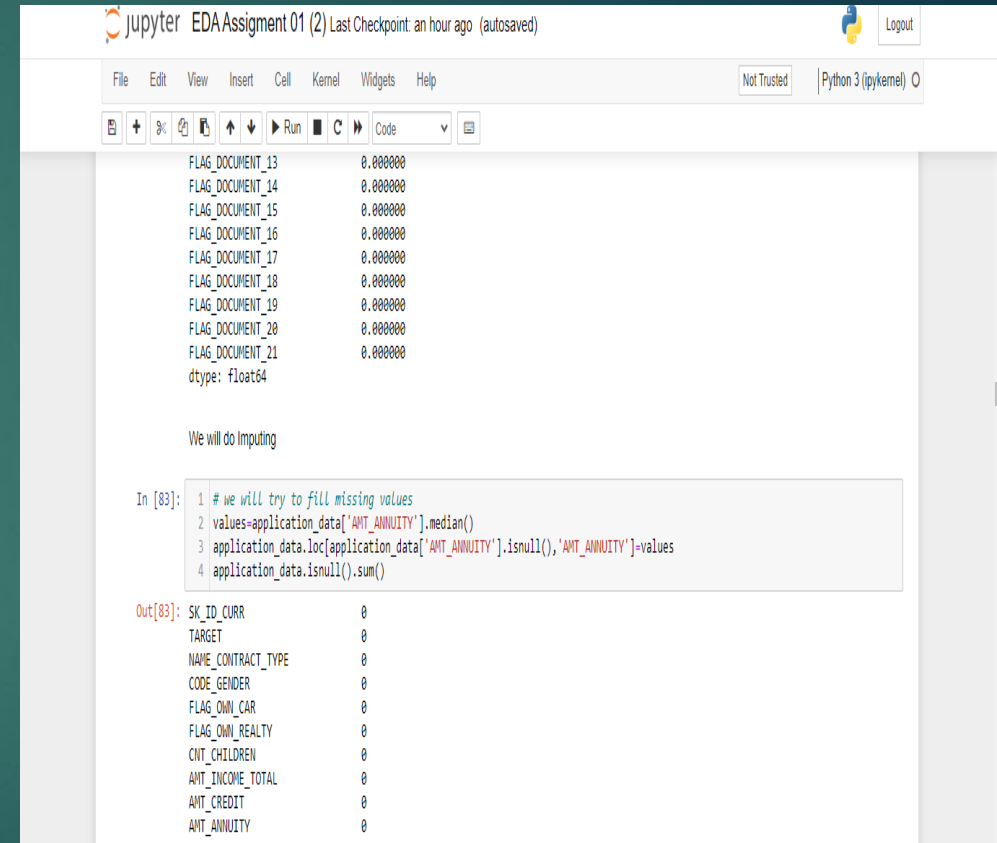
```
In [52]: 1 #Loading application_data file
2 application_data = pd.read_csv("application_data.csv")
3 application_data.head()
```

Out[52]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CRED
0	100002	1	Cash loans	M	N	Y	0	202500.0	40659
1	100003	0	Cash loans	F	N	N	0	270000.0	129350

Just couple of more steps mentioning

- ▶ Identifying the row with less data like less than 30%
- ▶ Remove all unwanted columns
- ▶ Rename columns wherever its needed
- ▶ Start Imputing where its needed



The screenshot shows a Jupyter Notebook titled "EDAAssignment 01 (2)" with a "Python 3 (pykernel)" environment. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running code, and viewing output. The notebook content displays a series of floating-point values (0.000000) for columns FLAG_DOCUMENT_13 through FLAG_DOCUMENT_21, with a dtype of float64. Below this, a text prompt reads "We will do Imputing". A code cell, labeled "In [83]:", contains the following Python code:

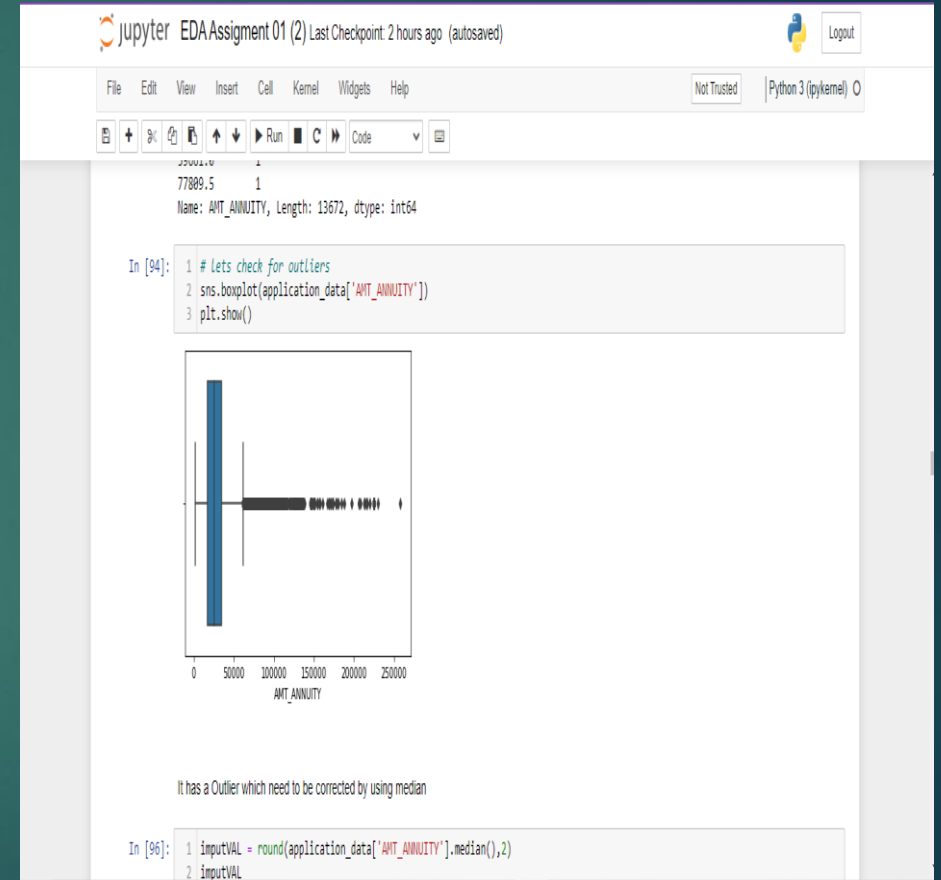
```
1 # we will try to fill missing values
2 values=application_data['AMT_ANNUITY'].median()
3 application_data.loc[application_data['AMT_ANNUITY'].isnull(), 'AMT_ANNUITY']=values
4 application_data.isnull().sum()
```

The output of this cell, labeled "Out[83]:", shows the sum of missing values for various columns, all of which are 0:

```
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
NAME_INCOME_TYPE  0
```

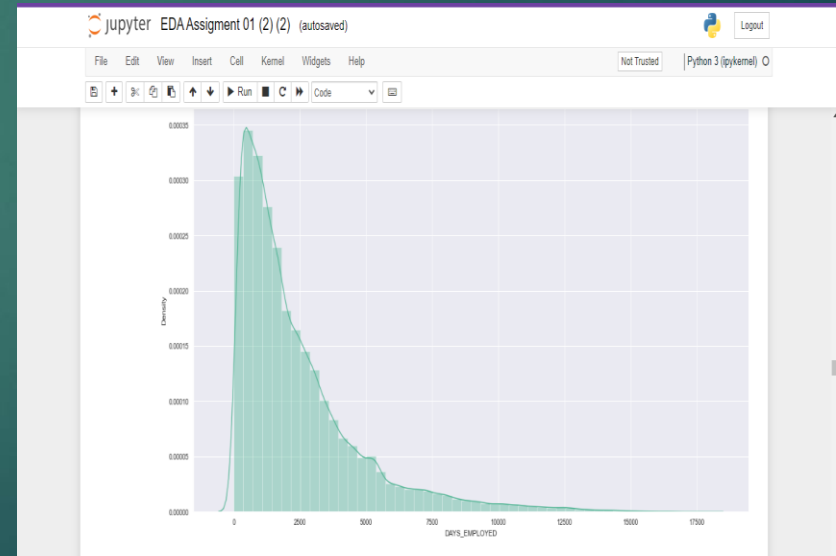
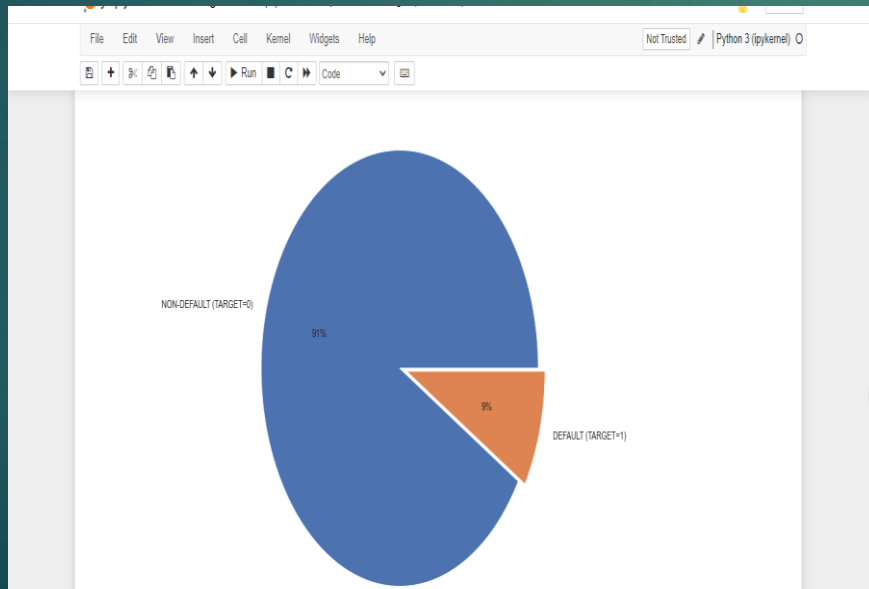
Data Analysis

- ▶ Finding Outliers through Graphs
- ▶ Dropping unwanted values
- ▶ Modifying different rows to make it operational.



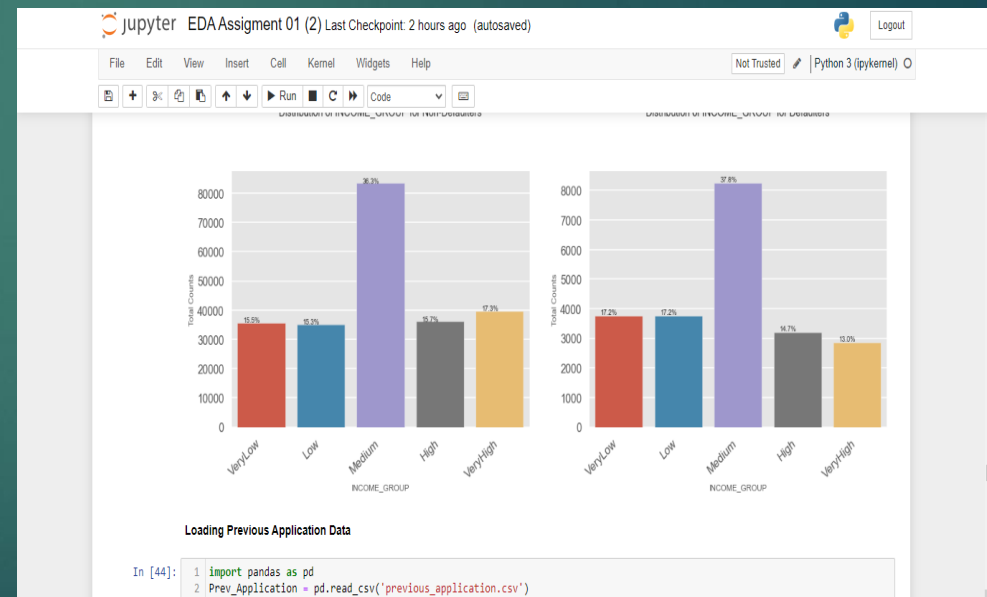
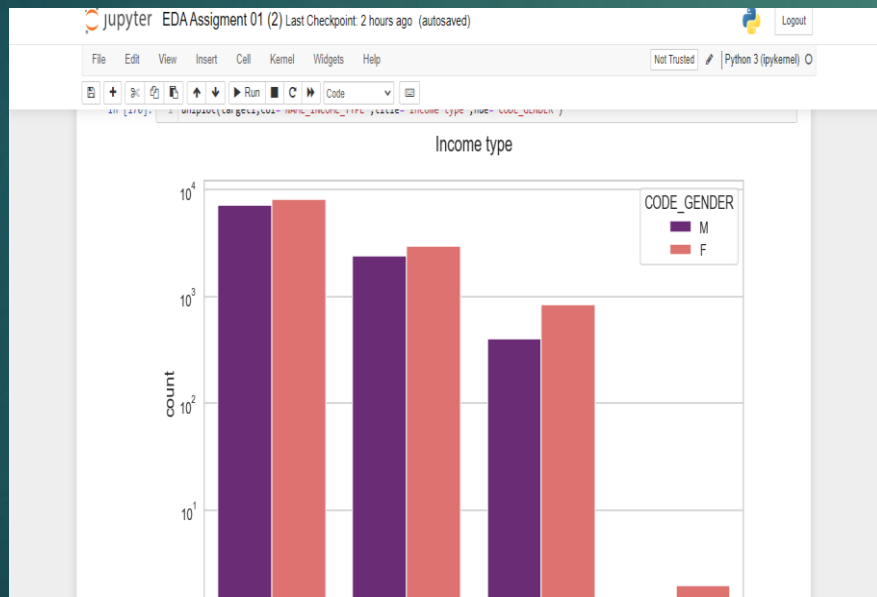
Data Understanding for Analysis

- ▶ Univariate and Bivariate
- ▶ Univariate analysis looks at one variable
- ▶ And Multi variate



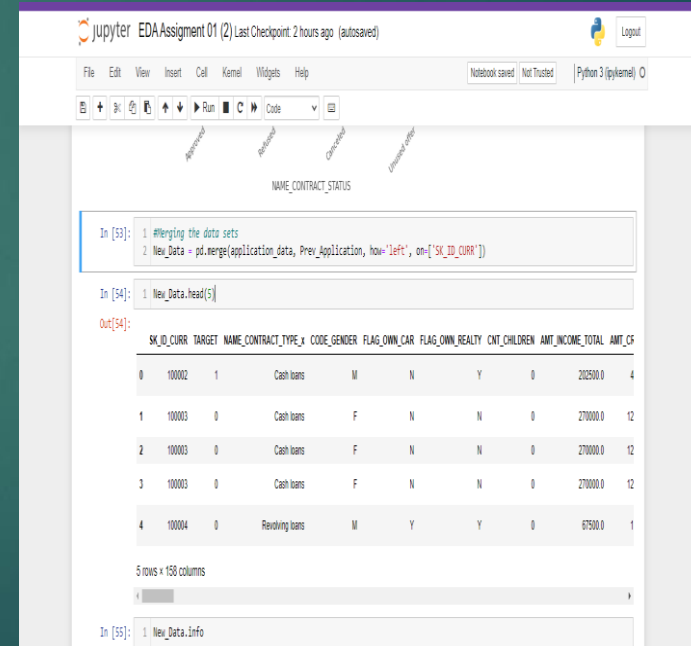
Data Analysis

- ▶ We need to know the condition income type
- ▶ Then also need to find the relation



Data Merge

- ▶ After analysing all the data we can perform the merge operation
- ▶ After joining then only we can develop the relation between in order to analyse further



The screenshot shows a Jupyter Notebook interface with the title "jupyter EDAAssignment 01 (2) Last Checkpoint: 2 hours ago (autosaved)". The notebook contains two code cells. The first cell, labeled "In [53]:", performs a merge operation: `1 #Merging the data sets` and `2 New_Data = pd.merge(application_data, Prev_Application, how="left", on=['SK_ID_CURR'])`. The second cell, labeled "In [54]:", displays the first five rows of the merged data: `1 New_Data.head(5)`. The output, labeled "Out[54]:", shows a table with 10 columns: SK_ID_CURR, TARGET, NAME_CONTRACT_TYPE_x, CODE_GENDER, FLAG_OWN_CAR, FLAG_OWN_REALTY, CNT_CHILDREN, AMT_INCOME_TOTAL, and AMT_SF. The table contains 5 rows of data. Below the table, it indicates "5 rows x 10 columns". The third cell, labeled "In [55]:", contains the command `1 New_Data.info`.

```
In [53]: 1 #Merging the data sets
         2 New_Data = pd.merge(application_data, Prev_Application, how="left", on=['SK_ID_CURR'])

In [54]: 1 New_Data.head(5)

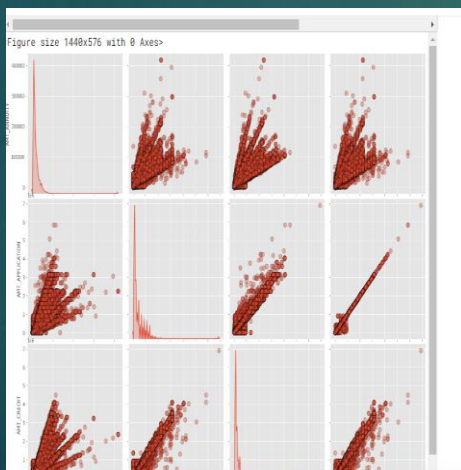
Out[54]:
   SK_ID_CURR  TARGET  NAME_CONTRACT_TYPE_x  CODE_GENDER  FLAG_OWN_CAR  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_SF
0      100002      1      Cash loans          M          N          Y          0      202500.0      4
1      100003      0      Cash loans          F          N          N          0      270000.0     12
2      100003      0      Cash loans          F          N          N          0      270000.0     12
3      100003      0      Cash loans          F          N          N          0      270000.0     12
4      100004      0      Revolving loans        M          Y          Y          0      67500.0      1

5 rows x 10 columns

In [55]: 1 New_Data.info
```


Univariate Analysis

- ▶ Some more analysis is needed to understand as we know after merge there are few departments having high defaulter who failed in repayment
- ▶ Data cleaning will still the first approach for second data set as it is possible that previous data can have some errors



```
jupyter EDAAssignment 01 (2) Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3 (ipykernel)

1438150 NaN 1 1 1
1438151 NaN 1 1 1
1438152 NaN 1 1 1
1438153 NaN 1 1 1
1438154 NaN 1 1 1

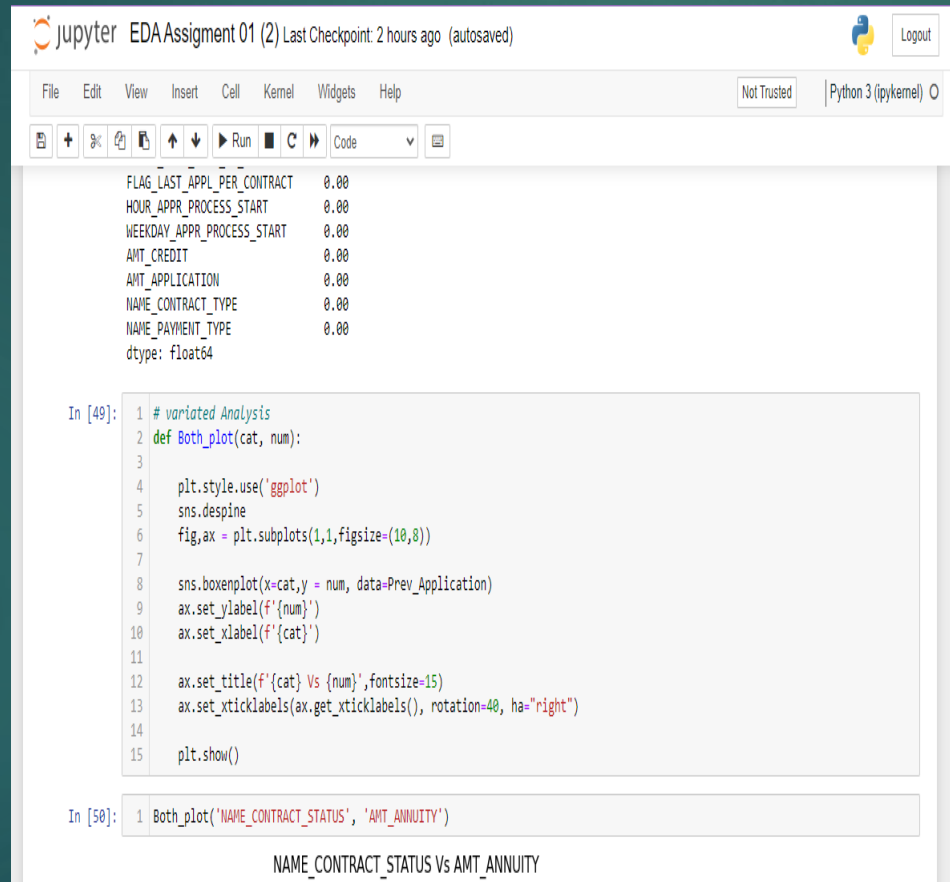
In [65]: 1 import seaborn as sns

In [71]: 1 New_Data1 = New_Data.rename({'NAME_CONTRACT_TYPE': 'NAME_CONTRACT_TYPE', 'AMT_CREDIT': 'AMT_CREDIT', 'AMT_ANNUITY': 'AMT_ANNUITY', 'WEEKDAY_APPR_PROCESS_START': 'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START': 'HOUR_APPR_PROCESS_START', 'NAME_CONTRACT_TYPE_PREV': 'NAME_CONTRACT_TYPE_PREV', 'AMT_CREDIT_PREV': 'AMT_CREDIT_PREV', 'AMT_ANNUITY_PREV': 'AMT_ANNUITY_PREV', 'WEEKDAY_APPR_PROCESS_STARTx': 'WEEKDAY_APPR_PROCESS_STARTx', 'HOUR_APPR_PROCESS_STARTx': 'HOUR_APPR_PROCESS_STARTx'}, axis=1)

In [75]: 1 # dropping unwanted data
2 New_Data1.drop(['SK_ID_CURR', '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', 'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY'], axis=1, inplace=True)

In [79]: 1 sns.set_style('whitegrid')
2 sns.set_context('talk')
3
4 plt.figure(figsize=(15,30))
```

Variate analysis on Previous data set



The image shows a Jupyter Notebook interface with the title "EDAAssigment 01 (2) Last Checkpoint: 2 hours ago (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar indicating "Not Trusted" and "Python 3 (ipykernel)".

The notebook contains two code cells. The first cell, labeled "In [49]:", defines a function `Both_plot` for creating a boxplot. The second cell, labeled "In [50]:", calls this function with the arguments `"NAME_CONTRACT_STATUS"` and `"AMT_ANNUITY"`.

Below the code cells, the output of the first cell is displayed as a text representation of a boxplot, showing the distribution of `AMT_ANNUITY` for each `NAME_CONTRACT_STATUS`. The output includes the variable names, their data types, and the `dtype: float64` label.

```
FLAG_LAST_APPL_PER_CONTRACT    0.00
HOUR_APPR_PROCESS_START         0.00
WEEKDAY_APPR_PROCESS_START      0.00
AMT_CREDIT                      0.00
AMT_APPLICATION                 0.00
NAME_CONTRACT_TYPE              0.00
NAME_PAYMENT_TYPE               0.00
dtype: float64
```

```
In [49]: 1 # variated Analysis
        2 def Both_plot(cat, num):
        3
        4     plt.style.use('ggplot')
        5     sns.despine
        6     fig,ax = plt.subplots(1,1,figsize=(10,8))
        7
        8     sns.boxenplot(x=cat,y = num, data=Prev_Application)
        9     ax.set_ylabel(f'{num}')
       10     ax.set_xlabel(f'{cat}')
       11
       12     ax.set_title(f'{cat} Vs {num}',fontsize=15)
       13     ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
       14
       15     plt.show()
```

```
In [50]: 1 Both_plot("NAME_CONTRACT_STATUS", "AMT_ANNUITY")
```

NAME_CONTRACT_STATUS Vs AMT_ANNUITY

Data Merge Analysis

- ▶ Try perform correlation on both data set in order to get the exact understanding and used different Graphs to understand the data impurities.
- ▶ Also Divided data set into two parts first so that we can perform different analysis simultaneously like univariate, multivariate

Now we need to divide our data set into two part payment difficulties and all other

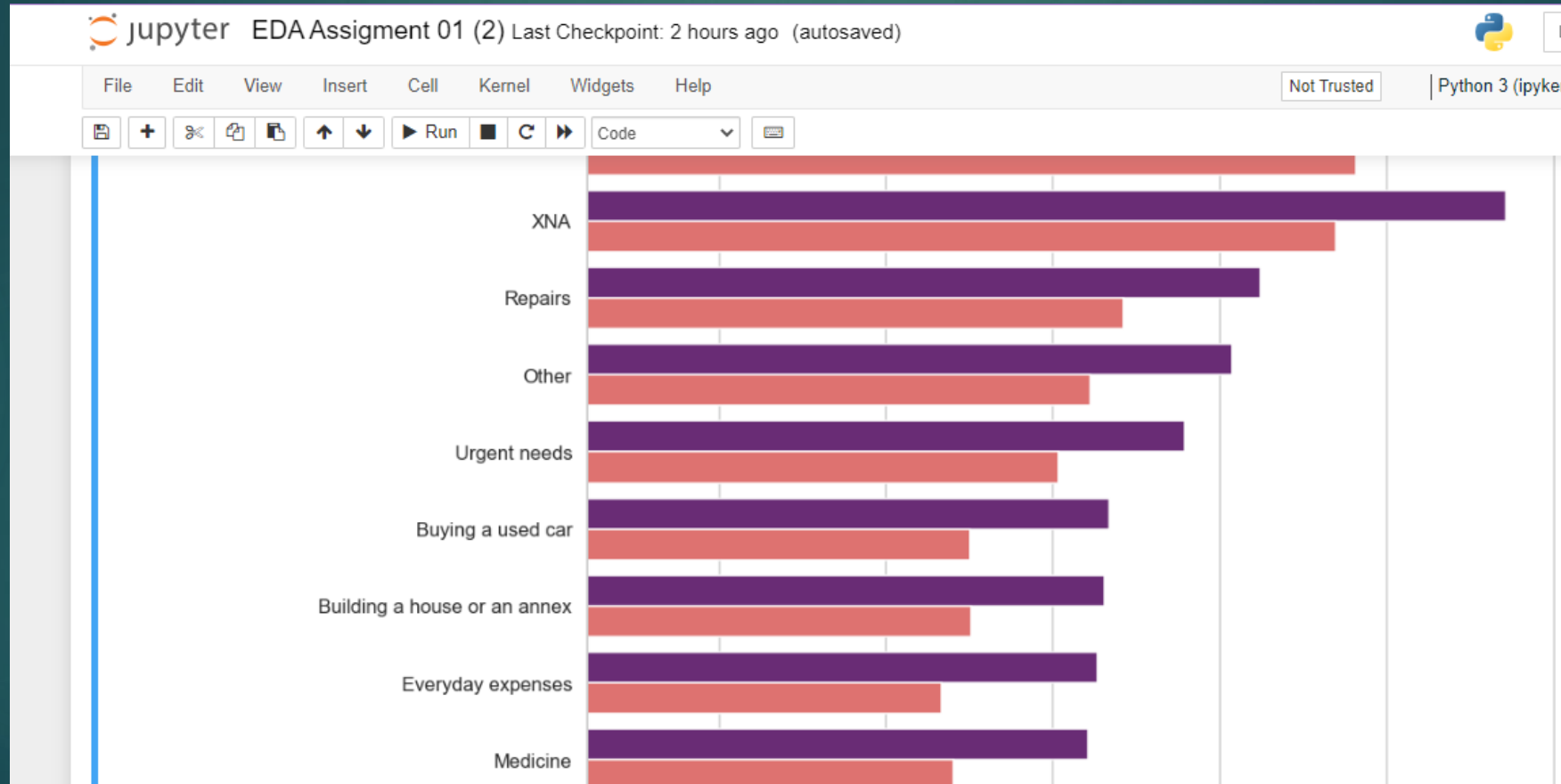
```
In [140]: 1 target0 = application_data.loc[application_data.TARGET == 0]
          2 target1 = application_data.loc[application_data.TARGET == 1]
```

```
In [144]: 1 target0.describe()
```

```
Out[144]:
```

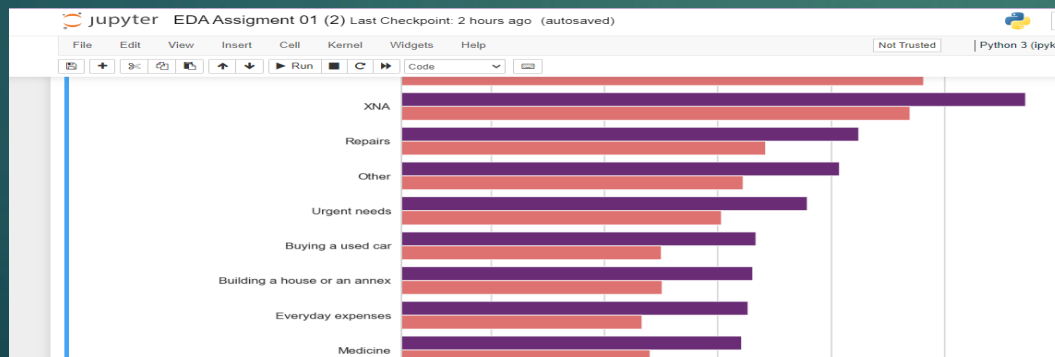
	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	REGION_POPULATION_RELATIVE	DAYS_BIRTH	DAYS_E
count	230302.000000	230302.0	230302.000000	2.303020e+05	2.303020e+05	230302.000000	230302.000000	230302.000000	230302.000000
mean	278159.213719	0.0	0.496696	1.764984e+05	6.164879e+05	27902.554759	0.021068	40.166386	24.000000
std	102858.253592	0.0	0.761316	1.154998e+05	4.114378e+05	14833.644504	0.014038	10.041735	23.000000
min	100003.000000	0.0	0.000000	2.565000e+04	4.500000e+04	1980.000000	0.000290	20.000000	0.000000
25%	188932.500000	0.0	0.000000	1.125000e+05	2.762775e+05	16969.500000	0.010006	32.000000	7.000000
50%	278200.500000	0.0	0.000000	1.575000e+05	5.212800e+05	25843.500000	0.018850	40.000000	16.000000
75%	367238.750000	0.0	1.000000	2.160000e+05	8.353800e+05	35743.500000	0.028663	48.000000	32.000000

Result Part compare to Business requirements



Conclusion Part

- ▶ Univariate analysis looks at one variable
- ▶ Banks Should consider and take care of loan request made by unemployed like Student loan and form average sector like small Business Sector
- ▶ Education Type has many difficulties in loan repayment.
- ▶ Default list is also greater for Repair group.



Assumptions on basis of Results

- ▶ We can see that the people who were approved for a loan earlier, defaulted less often where as people who were refused a loan earlier have higher chances of defaulting.
- ▶ We see that code gender doesn't have any effect on application approval or rejection.
- ▶ But we saw earlier that female have lesser chances of default compared to males. The bank can add more weightage to female while approving a loan amount

THE END

THANK YOU