Home Credit Default Risk (HCDR)

In [1]:
```
# #Mount Google Drive
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
drive.mount("/content/gdrive", force_remount=True).
```

The course project is based on the Home Credit Default Risk (HCDR) Kaggle Competition. The goal of this project is to predict whether or not a client will repay a loan. In order to make sure that people who struggle to get loans due to insufficient or non-existent credit histories have a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

Some of the challenges

1. Dataset size
   - (688 meg uncompressed) with millions of rows of data
   - 2.71 Gig of data uncompressed

- Dealing with missing data
- Imbalanced datasets
- Summarizing transaction data

Kaggle API setup

Kaggle is a Data Science Competition Platform which shares a lot of datasets. In the past, it was troublesome to submit your result as your have to go through the console in your browser and drag your files there. Now you can interact with Kaggle via the command line. E.g.,

```
! kaggle competitions files home-credit-default-risk
```

It is quite easy to setup, it takes me less than 15 minutes to finish a submission.

1. Install library

- Create a API Token (edit your profile on Kaggle.com); this produces `kaggle.json` file
- Put your JSON `kaggle.json` in the right place
- Access competition files; make submissions via the command (see examples below)
- Submit result

For more detailed information on setting the Kaggle API see here and here.

In [2]:
```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-package
s (1.5.12)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-pack
ages (from kaggle) (1.15.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packag
es (from kaggle) (2021.10.8)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packag
es (from kaggle) (1.24.3)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packa
ges (from kaggle) (2.23.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dis
t-packages (from kaggle) (2.8.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
(from kaggle) (4.64.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist
-packages (from kaggle) (6.1.2)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.
7/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-p
ackages (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/d
ist-packages (from requests->kaggle) (3.0.4)
```

In [3]:
```
pwd
```

Out[3]:
```
'/content'
```

In [4]:
```
!mkdir ~/.kaggle
!cp /root/shared/Downloads/kaggle.json ~/.kaggle
!chmod 600 ~/.kaggle/kaggle.json
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
cp: cannot stat '/root/shared/Downloads/kaggle.json': No such file or director
y
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
```

In [5]:
```
! kaggle competitions files home-credit-default-risk
```

```
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 5, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python3.7/dist-packages/kaggle/__init__.py", line 23, i
n <module>
    api.authenticate()
  File "/usr/local/lib/python3.7/dist-packages/kaggle/api/kaggle_api_extended.
py", line 166, in authenticate
    self.config_file, self.config_dir))
OSError: Could not find kaggle.json. Make sure it's located in /root/.kaggle.
Or use the environment method.
```

Team and project meta information.

Project Title - Home Credit Default Risk

Group Number - 16

Team Members:

- Naga Jahnavi Dhulipalla – ndhulipa@iu.edu
- Shivani Chennoju – schennoj@iu.edu
- Shalini Kothuru – Skothuru@iu.edu
- Bharath Chowdary Anumolu – banumolu@iu.edu

Credit Assessment Plan

Phase - 4		
Task	**Description**	**Contributor**
Abstract	Brief summary about the project Home Credit Default Risk	Bharath
Data Description	Brief summary about the data set files which are used in this project	Bharath
Metrics Updates	Updation of metrics as per MLP	Jahnavi
Neural Network	implementing different neural networks	Shivani
Loss Functions	Loss Functions	Shivani
Validation and Results	Accuracies and AOC values obtained after performing Multi Layer perceptron	Shivani
Phase Leader Plan	Assignment of all the taks for this phase	Shalini
PPT	Presentation of Phase-4	Shalini
Video	Video	Group

Dataset and how to download

Back ground Home Credit Group

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders.

Home Credit Group

Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Background on the dataset

Home Credit is a non-banking financial institution, founded in 1997 in the Czech Republic.

The company operates in 14 countries (including United States, Russia, Kazahstan, Belarus, China, India) and focuses on lending primarily to people with little or no credit history which will either not obtain loans or became victims of untrustworthly lenders.

Home Credit group has over 29 million customers, total assests of 21 billions Euro, over 160 millions loans, with the majority in Asia and and almost half of them in China (as of 19-05-2018).

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Data files overview

There are 7 different sources of data:

- **application_train/application_test:** the main training and testing data with information about each loan application at Home Credit. Every loan has its own row and is identified by the feature SK_ID_CURR. The training application data comes with the TARGET indicating **0: the loan was repaid** or **1: the loan was not repaid**. The target variable defines if the client had payment difficulties meaning he/she had late payment more

than X days on at least one of the first Y installments of the loan. Such case is marked as 1 while other all other cases as 0.

- **bureau:** data concerning client's previous credits from other financial institutions. Each previous credit has its own row in bureau, but one loan in the application data can have multiple previous credits.
- **bureau_balance:** monthly data about the previous credits in bureau. Each row is one month of a previous credit, and a single previous credit can have multiple rows, one for each month of the credit length.
- **previous_application:** previous applications for loans at Home Credit of clients who have loans in the application data. Each current loan in the application data can have multiple previous loans. Each previous application has one row and is identified by the feature SK_ID_PREV.
- **POS_CASH_BALANCE:** monthly data about previous point of sale or cash loans clients have had with Home Credit. Each row is one month of a previous point of sale or cash loan, and a single previous loan can have many rows.
- credit_card_balance: monthly data about previous credit cards clients have had with Home Credit. Each row is one month of a credit card balance, and a single credit card can have many rows.
- **installments_payment:** payment history for previous loans at Home Credit. There is one row for every made payment and one row for every missed payment.

In [6]:
```
DATA_DIR = "/content/gdrive/MyDrive/AML/home-credit-default-risk"   #same level
#DATA_DIR = os.path.join('./ddddd/')
!mkdir $DATA_DIR
```

```
mkdir: cannot create directory ‘/content/gdrive/My’: Operation not supported
mkdir: cannot create directory ‘Drive/AML_DATA’: No such file or directory
```

In [7]:
```
!ls -l $DATA_DIR
```

```
ls: cannot access '/content/gdrive/My': No such file or directory
ls: cannot access 'Drive/AML_DATA': No such file or directory
```

In [8]:
```
! kaggle competitions download home-credit-default-risk -p $DATA_DIR
```

```
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 5, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python3.7/dist-packages/kaggle/__init__.py", line 23, i
n <module>
    api.authenticate()
  File "/usr/local/lib/python3.7/dist-packages/kaggle/api/kaggle_api_extended.
py", line 166, in authenticate
    self.config_file, self.config_dir))
OSError: Could not find kaggle.json. Make sure it's located in /root/.kaggle.
Or use the environment method.
```

In [9]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import os
import zipfile
```

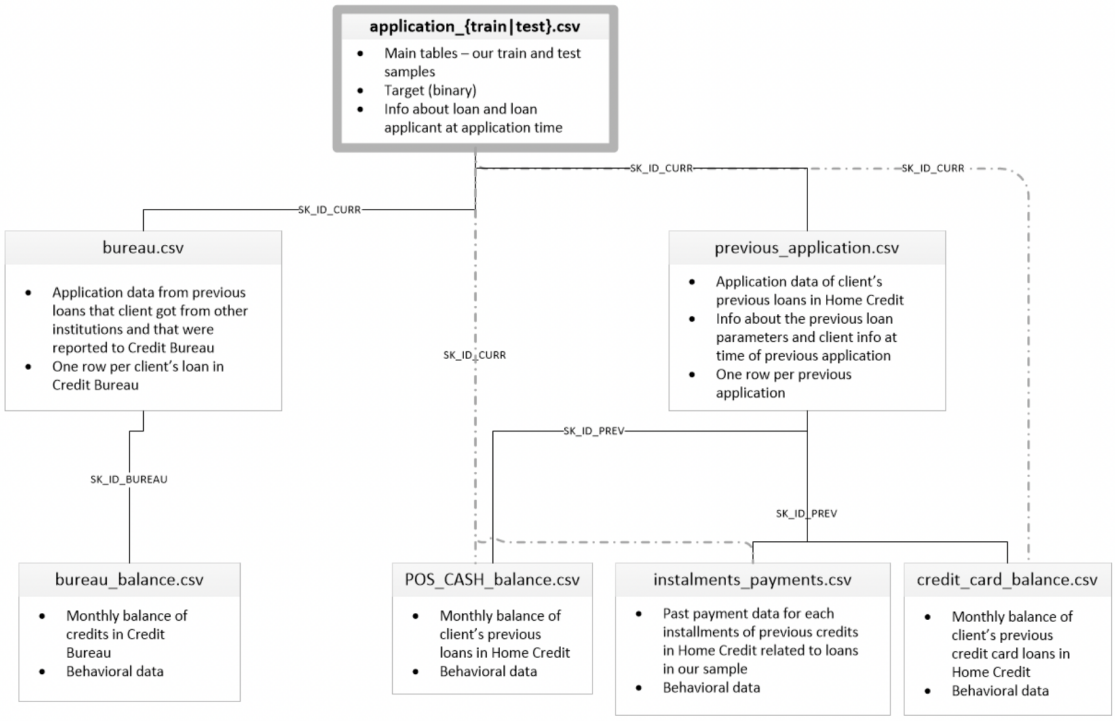
```
from sklearn.base import BaseEstimator, TransformerMixin
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline, FeatureUnion
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
import warnings
warnings.filterwarnings('ignore')
```

In [10]:
```
unzippingReq = False
if unzippingReq: #please modify this code
    zip_ref = zipfile.ZipFile('application_train.csv.zip', 'r')
    zip_ref.extractall('datasets')
    zip_ref.close()
    zip_ref = zipfile.ZipFile('application_test.csv.zip', 'r')
    zip_ref.extractall('datasets')
    zip_ref.close()
    zip_ref = zipfile.ZipFile('bureau_balance.csv.zip', 'r')
    zip_ref.extractall('datasets')
    zip_ref.close()
    zip_ref = zipfile.ZipFile('bureau.csv.zip', 'r')
    zip_ref.extractall('datasets')
    zip_ref.close()
    zip_ref = zipfile.ZipFile('credit_card_balance.csv.zip', 'r')
    zip_ref.extractall('datasets')
    zip_ref.close()
    zip_ref = zipfile.ZipFile('installments_payments.csv.zip', 'r')
    zip_ref.extractall('datasets')
    zip_ref.close()
    zip_ref = zipfile.ZipFile('POS_CASH_balance.csv.zip', 'r')
    zip_ref.extractall('datasets')
    zip_ref.close()
    zip_ref = zipfile.ZipFile('previous_application.csv.zip', 'r')
    zip_ref.extractall('datasets')
    zip_ref.close()
```

Data files overview

Data Dictionary

As part of the data download comes a Data Dictionary. It named `HomeCredit_columns_description.csv`



Application train

> Indented block

In [11]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import os
import zipfile
from sklearn.base import BaseEstimator, TransformerMixin
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline, FeatureUnion
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
import warnings
warnings.filterwarnings('ignore')

def load_data(in_path, name):
    df = pd.read_csv(in_path)
    print(f"{name}: shape is {df.shape}")
    print(df.info())
    display(df.head(5))
    return df
```

```
datasets={}  # lets store the datasets in a dictionary so we can keep track of
ds_name = 'application_train'
datasets[ds_name] = load_data(os.path.join(DATA_DIR, f'{ds_name}.csv'), ds_name

datasets['application_train'].shape
```

```
application_train: shape is (307511, 122)
<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
None
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWI
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	

5 rows × 122 columns

Out[11]:
```
(307511, 122)
```

Application test

- **application_train/application_test:** the main training and testing data with information about each loan application at Home Credit. Every loan has its own row and is identified by the feature SK_ID_CURR. The training application data comes with the TARGET indicating **0: the loan was repaid** or **1: the loan was not repaid**. The target variable defines if the client had payment difficulties meaning he/she had late payment more than X days on at least one of the first Y installments of the loan. Such case is marked as 1 while other all other cases as 0.

In [12]:
```
ds_name = 'application_test'
datasets[ds_name] = load_data(os.path.join(DATA_DIR, f'{ds_name}.csv'), ds_name
```

```
application_test: shape is (48744, 121)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48744 entries, 0 to 48743
Columns: 121 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(40), object(16)
memory usage: 45.0+ MB
None
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
0	100001	Cash loans	F	N	Y
1	100005	Cash loans	M	N	Y
2	100013	Cash loans	M	Y	Y
3	100028	Cash loans	F	N	Y
4	100038	Cash loans	M	Y	N

5 rows × 121 columns

The Other datasets

- **bureau:** data concerning client's previous credits from other financial institutions. Each previous credit has its own row in bureau, but one loan in the application data can have multiple previous credits.
- **bureau_balance:** monthly data about the previous credits in bureau. Each row is one month of a previous credit, and a single previous credit can have multiple rows, one for each month of the credit length.
- **previous_application:** previous applications for loans at Home Credit of clients who have loans in the application data. Each current loan in the application data can have multiple previous loans. Each previous application has one row and is identified by the feature SK_ID_PREV.
- **POS_CASH_BALANCE:** monthly data about previous point of sale or cash loans clients have had with Home Credit. Each row is one month of a previous point of sale or cash loan, and a single previous loan can have many rows.
- credit_card_balance: monthly data about previous credit cards clients have had with Home Credit. Each row is one month of a credit card balance, and a single credit card can have many rows.
- **installments_payment:** payment history for previous loans at Home Credit. There is one row for every made payment and one row for every missed payment.

In [13]:
```
%%time
ds_names = ("application_train", "application_test", "bureau","bureau_balance",
            "previous_application","POS_CASH_balance")

for ds_name in ds_names:
    datasets[ds_name] = load_data(os.path.join(DATA_DIR, f'{ds_name}.csv'), ds_
```

```
application_train: shape is (307511, 122)
<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
None
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWI
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	

5 rows × 122 columns

```
application_test: shape is (48744, 121)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48744 entries, 0 to 48743
Columns: 121 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(40), object(16)
memory usage: 45.0+ MB
None
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
0	100001	Cash loans	F	N	Y
1	100005	Cash loans	M	N	Y
2	100013	Cash loans	M	Y	Y
3	100028	Cash loans	F	N	Y
4	100038	Cash loans	M	Y	N

5 rows × 121 columns

```
bureau: shape is (1716428, 17)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1716428 entries, 0 to 1716427
Data columns (total 17 columns):
 #   Column                  Dtype  
---  ------                  -----  
 0   SK_ID_CURR              int64  
 1   SK_ID_BUREAU            int64  
 2   CREDIT_ACTIVE           object 
 3   CREDIT_CURRENCY         object 
 4   DAYS_CREDIT             int64  
 5   CREDIT_DAY_OVERDUE      int64  
 6   DAYS_CREDIT_ENDDATE     float64
 7   DAYS_ENDDATE_FACT       float64
 8   AMT_CREDIT_MAX_OVERDUE  float64
 9   CNT_CREDIT_PROLONG      int64  
 10  AMT_CREDIT_SUM          float64
 11  AMT_CREDIT_SUM_DEBT     float64
 12  AMT_CREDIT_SUM_LIMIT    float64
 13  AMT_CREDIT_SUM_OVERDUE  float64
 14  CREDIT_TYPE             object 
 15  DAYS_CREDIT_UPDATE      int64  
 16  AMT_ANNUITY             float64
dtypes: float64(8), int64(6), object(3)
memory usage: 222.6+ MB
None
```

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_D
0	215354	5714462	Closed	currency 1	-497	
1	215354	5714463	Active	currency 1	-208	
2	215354	5714464	Active	currency 1	-203	
3	215354	5714465	Active	currency 1	-203	
4	215354	5714466	Active	currency 1	-629	

```
bureau_balance: shape is (27299925, 3)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27299925 entries, 0 to 27299924
Data columns (total 3 columns):
 #   Column          Dtype
---  ------          -----
 0   SK_ID_BUREAU    int64
 1   MONTHS_BALANCE  int64
 2   STATUS          object
dtypes: int64(2), object(1)
memory usage: 624.8+ MB
None
```

	SK_ID_BUREAU	MONTHS_BALANCE	STATUS
0	5715448	0	C
1	5715448	-1	C
2	5715448	-2	C
3	5715448	-3	C
4	5715448	-4	C

```
credit_card_balance: shape is (3840312, 23)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3840312 entries, 0 to 3840311
Data columns (total 23 columns):
 #   Column                       Dtype
---  ------                       -----
 0   SK_ID_PREV                   int64
 1   SK_ID_CURR                   int64
 2   MONTHS_BALANCE               int64
 3   AMT_BALANCE                  float64
 4   AMT_CREDIT_LIMIT_ACTUAL      int64
 5   AMT_DRAWINGS_ATM_CURRENT     float64
 6   AMT_DRAWINGS_CURRENT         float64
 7   AMT_DRAWINGS_OTHER_CURRENT   float64
 8   AMT_DRAWINGS_POS_CURRENT     float64
 9   AMT_INST_MIN_REGULARITY      float64
 10  AMT_PAYMENT_CURRENT          float64
 11  AMT_PAYMENT_TOTAL_CURRENT    float64
 12  AMT_RECEIVABLE_PRINCIPAL     float64
 13  AMT_RECIVABLE                float64
 14  AMT_TOTAL_RECEIVABLE         float64
 15  CNT_DRAWINGS_ATM_CURRENT     float64
 16  CNT_DRAWINGS_CURRENT         int64
 17  CNT_DRAWINGS_OTHER_CURRENT   float64
 18  CNT_DRAWINGS_POS_CURRENT     float64
 19  CNT_INSTALMENT_MATURE_CUM    float64
 20  NAME_CONTRACT_STATUS         object
 21  SK_DPD                       int64
 22  SK_DPD_DEF                   int64
dtypes: float64(15), int64(7), object(1)
memory usage: 673.9+ MB
None
```

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL
0	2562384	378907	-6	56.970	135000
1	2582071	363914	-1	63975.555	45000
2	1740877	371185	-7	31815.225	450000
3	1389973	337855	-4	236572.110	225000
4	1891521	126868	-1	453919.455	450000

5 rows × 23 columns

```
installments_payments: shape is (13605401, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13605401 entries, 0 to 13605400
Data columns (total 8 columns):
 #   Column                  Dtype  
---  ------                  -----  
 0   SK_ID_PREV              int64  
 1   SK_ID_CURR              int64  
 2   NUM_INSTALMENT_VERSION  float64
 3   NUM_INSTALMENT_NUMBER   int64  
 4   DAYS_INSTALMENT         float64
 5   DAYS_ENTRY_PAYMENT      float64
 6   AMT_INSTALMENT          float64
 7   AMT_PAYMENT             float64
dtypes: float64(5), int64(3)
memory usage: 830.4 MB
None
```

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYS_
0	1054186	161674	1.0	6	
1	1330831	151639	0.0	34	
2	2085231	193053	2.0	1	
3	2452527	199697	1.0	3	
4	2714724	167756	1.0	2	

```
previous_application: shape is (1670214, 37)
<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  
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 16  NAME_CONTRACT_STATUS         1670214 non-null  object 
 17  DAYS_DECISION                1670214 non-null  int64  
 18  NAME_PAYMENT_TYPE            1670214 non-null  object 
 19  CODE_REJECT_REASON           1670214 non-null  object 
 20  NAME_TYPE_SUITE              849809 non-null   object 
 21  NAME_CLIENT_TYPE             1670214 non-null  object 
 22  NAME_GOODS_CATEGORY          1670214 non-null  object 
 23  NAME_PORTFOLIO               1670214 non-null  object 
 24  NAME_PRODUCT_TYPE            1670214 non-null  object 
 25  CHANNEL_TYPE                 1670214 non-null  object 
 26  SELLERPLACE_AREA             1670214 non-null  int64  
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object 
 30  PRODUCT_COMBINATION          1669868 non-null  object 
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
None
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AM
0	2030495	271877	Consumer loans	1730.430	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	
2	2523466	122040	Cash loans	15060.735	112500.0	
3	2819243	176158	Cash loans	47041.335	450000.0	
4	1784265	202054	Cash loans	31924.395	337500.0	

5 rows × 37 columns

```
POS_CASH_balance: shape is (10001358, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10001358 entries, 0 to 10001357
Data columns (total 8 columns):
 #   Column                 Dtype
---  ------                 -----
 0   SK_ID_PREV             int64
 1   SK_ID_CURR             int64
 2   MONTHS_BALANCE         int64
 3   CNT_INSTALMENT         float64
 4   CNT_INSTALMENT_FUTURE  float64
 5   NAME_CONTRACT_STATUS   object
 6   SK_DPD                 int64
 7   SK_DPD_DEF             int64
dtypes: float64(2), int64(5), object(1)
memory usage: 610.4+ MB
None
```

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	CNT_INSTALMENT	CNT_INSTALMENT_FUTUR
0	1803195	182943	-31	48.0	45
1	1715348	367990	-33	36.0	35
2	1784872	397406	-32	12.0	9
3	1903291	269225	-35	48.0	42
4	2341044	334279	-35	36.0	35

```
CPU times: user 52.6 s, sys: 10.4 s, total: 1min 2s
Wall time: 1min 18s
```

In [14]:
```
for ds_name in datasets.keys():
    print(f'dataset {ds_name:24}: [ {datasets[ds_name].shape[0]:10,}, {datasets
```

```
dataset application_train       : [    307,511, 122]
dataset application_test        : [     48,744, 121]
dataset bureau                  : [  1,716,428, 17]
dataset bureau_balance          : [ 27,299,925, 3]
dataset credit_card_balance     : [  3,840,312, 23]
dataset installments_payments   : [ 13,605,401, 8]
dataset previous_application    : [  1,670,214, 37]
dataset POS_CASH_balance        : [ 10,001,358, 8]
```

In [ ]:
```

```

Exploratory Data Analysis

Summary of Application train

In [165…
```
datasets["application_train"].info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 122 columns):
 #    Column                        Non-Null Count   Dtype
---   ------                        --------------   -----
 0    SK_ID_CURR                    307511 non-null  int64
 1    TARGET                        307511 non-null  int64
 2    NAME_CONTRACT_TYPE            307511 non-null  object
 3    CODE_GENDER                   307511 non-null  object
 4    FLAG_OWN_CAR                  307511 non-null  object
 5    FLAG_OWN_REALTY               307511 non-null  object
 6    CNT_CHILDREN                  307511 non-null  int64
 7    AMT_INCOME_TOTAL              307511 non-null  float64
 8    AMT_CREDIT                    307511 non-null  float64
 9    AMT_ANNUITY                   307499 non-null  float64
 10   AMT_GOODS_PRICE               307233 non-null  float64
 11   NAME_TYPE_SUITE               306219 non-null  object
 12   NAME_INCOME_TYPE              307511 non-null  object
 13   NAME_EDUCATION_TYPE           307511 non-null  object
 14   NAME_FAMILY_STATUS            307511 non-null  object
 15   NAME_HOUSING_TYPE             307511 non-null  object
 16   REGION_POPULATION_RELATIVE    307511 non-null  float64
 17   DAYS_BIRTH                    307511 non-null  int64
 18   DAYS_EMPLOYED                 307511 non-null  int64
 19   DAYS_REGISTRATION             307511 non-null  float64
 20   DAYS_ID_PUBLISH               307511 non-null  int64
 21   OWN_CAR_AGE                   104582 non-null  float64
 22   FLAG_MOBIL                    307511 non-null  int64
 23   FLAG_EMP_PHONE                307511 non-null  int64
 24   FLAG_WORK_PHONE               307511 non-null  int64
 25   FLAG_CONT_MOBILE              307511 non-null  int64
 26   FLAG_PHONE                    307511 non-null  int64
 27   FLAG_EMAIL                    307511 non-null  int64
 28   OCCUPATION_TYPE               211120 non-null  object
 29   CNT_FAM_MEMBERS               307509 non-null  float64
 30   REGION_RATING_CLIENT          307511 non-null  int64
 31   REGION_RATING_CLIENT_W_CITY   307511 non-null  int64
 32   WEEKDAY_APPR_PROCESS_START    307511 non-null  object
 33   HOUR_APPR_PROCESS_START       307511 non-null  int64
 34   REG_REGION_NOT_LIVE_REGION    307511 non-null  int64
 35   REG_REGION_NOT_WORK_REGION    307511 non-null  int64
 36   LIVE_REGION_NOT_WORK_REGION   307511 non-null  int64
 37   REG_CITY_NOT_LIVE_CITY        307511 non-null  int64
 38   REG_CITY_NOT_WORK_CITY        307511 non-null  int64
 39   LIVE_CITY_NOT_WORK_CITY       307511 non-null  int64
 40   ORGANIZATION_TYPE             307511 non-null  object
 41   EXT_SOURCE_1                  134133 non-null  float64
 42   EXT_SOURCE_2                  306851 non-null  float64
 43   EXT_SOURCE_3                  246546 non-null  float64
 44   APARTMENTS_AVG                151450 non-null  float64
 45   BASEMENTAREA_AVG              127568 non-null  float64
 46   YEARS_BEGINEXPLUATATION_AVG   157504 non-null  float64
 47   YEARS_BUILD_AVG               103023 non-null  float64
 48   COMMONAREA_AVG                92646 non-null   float64
 49   ELEVATORS_AVG                 143620 non-null  float64
 50   ENTRANCES_AVG                 152683 non-null  float64
 51   FLOORSMAX_AVG                 154491 non-null  float64
 52   FLOORSMIN_AVG                 98869 non-null   float64
 53   LANDAREA_AVG                  124921 non-null  float64
 54   LIVINGAPARTMENTS_AVG          97312 non-null   float64
```

55	LIVINGAREA_AVG	153161 non-null	float64
56	NONLIVINGAPARTMENTS_AVG	93997 non-null	float64
57	NONLIVINGAREA_AVG	137829 non-null	float64
58	APARTMENTS_MODE	151450 non-null	float64
59	BASEMENTAREA_MODE	127568 non-null	float64
60	YEARS_BEGINEXPLUATATION_MODE	157504 non-null	float64
61	YEARS_BUILD_MODE	103023 non-null	float64
62	COMMONAREA_MODE	92646 non-null	float64
63	ELEVATORS_MODE	143620 non-null	float64
64	ENTRANCES_MODE	152683 non-null	float64
65	FLOORSMAX_MODE	154491 non-null	float64
66	FLOORSMIN_MODE	98869 non-null	float64
67	LANDAREA_MODE	124921 non-null	float64
68	LIVINGAPARTMENTS_MODE	97312 non-null	float64
69	LIVINGAREA_MODE	153161 non-null	float64
70	NONLIVINGAPARTMENTS_MODE	93997 non-null	float64
71	NONLIVINGAREA_MODE	137829 non-null	float64
72	APARTMENTS_MEDI	151450 non-null	float64
73	BASEMENTAREA_MEDI	127568 non-null	float64
74	YEARS_BEGINEXPLUATATION_MEDI	157504 non-null	float64
75	YEARS_BUILD_MEDI	103023 non-null	float64
76	COMMONAREA_MEDI	92646 non-null	float64
77	ELEVATORS_MEDI	143620 non-null	float64
78	ENTRANCES_MEDI	152683 non-null	float64
79	FLOORSMAX_MEDI	154491 non-null	float64
80	FLOORSMIN_MEDI	98869 non-null	float64
81	LANDAREA_MEDI	124921 non-null	float64
82	LIVINGAPARTMENTS_MEDI	97312 non-null	float64
83	LIVINGAREA_MEDI	153161 non-null	float64
84	NONLIVINGAPARTMENTS_MEDI	93997 non-null	float64
85	NONLIVINGAREA_MEDI	137829 non-null	float64
86	FONDKAPREMONT_MODE	97216 non-null	object
87	HOUSETYPE_MODE	153214 non-null	object
88	TOTALAREA_MODE	159080 non-null	float64
89	WALLSMATERIAL_MODE	151170 non-null	object
90	EMERGENCYSTATE_MODE	161756 non-null	object
91	OBS_30_CNT_SOCIAL_CIRCLE	306490 non-null	float64
92	DEF_30_CNT_SOCIAL_CIRCLE	306490 non-null	float64
93	OBS_60_CNT_SOCIAL_CIRCLE	306490 non-null	float64
94	DEF_60_CNT_SOCIAL_CIRCLE	306490 non-null	float64
95	DAYS_LAST_PHONE_CHANGE	307510 non-null	float64
96	FLAG_DOCUMENT_2	307511 non-null	int64
97	FLAG_DOCUMENT_3	307511 non-null	int64
98	FLAG_DOCUMENT_4	307511 non-null	int64
99	FLAG_DOCUMENT_5	307511 non-null	int64
100	FLAG_DOCUMENT_6	307511 non-null	int64
101	FLAG_DOCUMENT_7	307511 non-null	int64
102	FLAG_DOCUMENT_8	307511 non-null	int64
103	FLAG_DOCUMENT_9	307511 non-null	int64
104	FLAG_DOCUMENT_10	307511 non-null	int64
105	FLAG_DOCUMENT_11	307511 non-null	int64
106	FLAG_DOCUMENT_12	307511 non-null	int64
107	FLAG_DOCUMENT_13	307511 non-null	int64
108	FLAG_DOCUMENT_14	307511 non-null	int64
109	FLAG_DOCUMENT_15	307511 non-null	int64
110	FLAG_DOCUMENT_16	307511 non-null	int64
111	FLAG_DOCUMENT_17	307511 non-null	int64
112	FLAG_DOCUMENT_18	307511 non-null	int64
113	FLAG_DOCUMENT_19	307511 non-null	int64
114	FLAG_DOCUMENT_20	307511 non-null	int64

```
 115  FLAG_DOCUMENT_21              307511 non-null  int64
 116  AMT_REQ_CREDIT_BUREAU_HOUR    265992 non-null  float64
 117  AMT_REQ_CREDIT_BUREAU_DAY     265992 non-null  float64
 118  AMT_REQ_CREDIT_BUREAU_WEEK    265992 non-null  float64
 119  AMT_REQ_CREDIT_BUREAU_MON     265992 non-null  float64
 120  AMT_REQ_CREDIT_BUREAU_QRT     265992 non-null  float64
 121  AMT_REQ_CREDIT_BUREAU_YEAR    265992 non-null  float64
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

In [166…
```
datasets["application_train"].shape
```

Out[166]:
```
(307511, 122)
```

In [167…
```
datasets["application_train"].size
```

Out[167]:
```
37516342
```

In [168…
```
datasets["application_train"].describe() #numerical only features
```

Out[168]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	A
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	3
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	2

8 rows × 106 columns

In [169…
```
datasets["application_test"].describe() #numerical only features
```

Out[169]:

	SK_ID_CURR	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
count	48744.000000	48744.000000	4.874400e+04	4.874400e+04	48720.000000	
mean	277796.676350	0.397054	1.784318e+05	5.167404e+05	29426.240209	
std	103169.547296	0.709047	1.015226e+05	3.653970e+05	16016.368315	
min	100001.000000	0.000000	2.694150e+04	4.500000e+04	2295.000000	
25%	188557.750000	0.000000	1.125000e+05	2.606400e+05	17973.000000	
50%	277549.000000	0.000000	1.575000e+05	4.500000e+05	26199.000000	
75%	367555.500000	1.000000	2.250000e+05	6.750000e+05	37390.500000	
max	456250.000000	20.000000	4.410000e+06	2.245500e+06	180576.000000	

8 rows × 105 columns

In [170…
```
datasets["application_train"].head()
```

Out[170]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OW
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	

5 rows × 122 columns

In [171…
```
#datasets["application_test"].describe() #numerical only features
```

In [172…
```
datasets["application_train"].describe(include='all') #look at all categorical
```

Out[172]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_C
count	307511.000000	307511.000000	307511	307511	307
unique	NaN	NaN	2	3	
top	NaN	NaN	Cash loans	F	
freq	NaN	NaN	278232	202448	202
mean	278180.518577	0.080729	NaN	NaN	N
std	102790.175348	0.272419	NaN	NaN	N
min	100002.000000	0.000000	NaN	NaN	N
25%	189145.500000	0.000000	NaN	NaN	N
50%	278202.000000	0.000000	NaN	NaN	N
75%	367142.500000	0.000000	NaN	NaN	N
max	456255.000000	1.000000	NaN	NaN	N

11 rows × 122 columns

In [173…
```
#df_app_train= pd.read_csv (r'/Users/nanda/Downloads/home-credit-default-risk/a
```

In [174…
```
print("Different datatypes in Apllication_train dataset")
datasets["application_train"].dtypes.value_counts()
```

Out[174]:
```
Different datatypes in Apllication_train dataset
 float64    65
 int64      41
 object     16
 dtype: int64
```

In [175…
```
columns= datasets["application_train"].columns
print(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)
```

In [176…

```
# i=0
for key in datasets["application_train"].keys():
    n=len(pd.unique(datasets["application_train"][key]))
    print('Nunber of Unique values in ', format(key), 'is', format(n))
#     i=i+1
# print(i)
```

```
Nunber of Unique values in  SK_ID_CURR is 307511
Nunber of Unique values in  TARGET is 2
Nunber of Unique values in  NAME_CONTRACT_TYPE is 2
Nunber of Unique values in  CODE_GENDER is 3
Nunber of Unique values in  FLAG_OWN_CAR is 2
Nunber of Unique values in  FLAG_OWN_REALTY is 2
Nunber of Unique values in  CNT_CHILDREN is 15
Nunber of Unique values in  AMT_INCOME_TOTAL is 2548
Nunber of Unique values in  AMT_CREDIT is 5603
Nunber of Unique values in  AMT_ANNUITY is 13673
Nunber of Unique values in  AMT_GOODS_PRICE is 1003
Nunber of Unique values in  NAME_TYPE_SUITE is 8
Nunber of Unique values in  NAME_INCOME_TYPE is 8
Nunber of Unique values in  NAME_EDUCATION_TYPE is 5
Nunber of Unique values in  NAME_FAMILY_STATUS is 6
Nunber of Unique values in  NAME_HOUSING_TYPE is 6
Nunber of Unique values in  REGION_POPULATION_RELATIVE is 81
Nunber of Unique values in  DAYS_BIRTH is 17460
Nunber of Unique values in  DAYS_EMPLOYED is 12574
Nunber of Unique values in  DAYS_REGISTRATION is 15688
Nunber of Unique values in  DAYS_ID_PUBLISH is 6168
Nunber of Unique values in  OWN_CAR_AGE is 63
Nunber of Unique values in  FLAG_MOBIL is 2
Nunber of Unique values in  FLAG_EMP_PHONE is 2
Nunber of Unique values in  FLAG_WORK_PHONE is 2
Nunber of Unique values in  FLAG_CONT_MOBILE is 2
Nunber of Unique values in  FLAG_PHONE is 2
Nunber of Unique values in  FLAG_EMAIL is 2
Nunber of Unique values in  OCCUPATION_TYPE is 19
Nunber of Unique values in  CNT_FAM_MEMBERS is 18
Nunber of Unique values in  REGION_RATING_CLIENT is 3
Nunber of Unique values in  REGION_RATING_CLIENT_W_CITY is 3
Nunber of Unique values in  WEEKDAY_APPR_PROCESS_START is 7
Nunber of Unique values in  HOUR_APPR_PROCESS_START is 24
Nunber of Unique values in  REG_REGION_NOT_LIVE_REGION is 2
Nunber of Unique values in  REG_REGION_NOT_WORK_REGION is 2
Nunber of Unique values in  LIVE_REGION_NOT_WORK_REGION is 2
Nunber of Unique values in  REG_CITY_NOT_LIVE_CITY is 2
Nunber of Unique values in  REG_CITY_NOT_WORK_CITY is 2
Nunber of Unique values in  LIVE_CITY_NOT_WORK_CITY is 2
Nunber of Unique values in  ORGANIZATION_TYPE is 58
Nunber of Unique values in  EXT_SOURCE_1 is 114585
Nunber of Unique values in  EXT_SOURCE_2 is 119832
Nunber of Unique values in  EXT_SOURCE_3 is 815
Nunber of Unique values in  APARTMENTS_AVG is 2340
Nunber of Unique values in  BASEMENTAREA_AVG is 3781
Nunber of Unique values in  YEARS_BEGINEXPLUATATION_AVG is 286
Nunber of Unique values in  YEARS_BUILD_AVG is 150
Nunber of Unique values in  COMMONAREA_AVG is 3182
Nunber of Unique values in  ELEVATORS_AVG is 258
Nunber of Unique values in  ENTRANCES_AVG is 286
Nunber of Unique values in  FLOORSMAX_AVG is 404
Nunber of Unique values in  FLOORSMIN_AVG is 306
Nunber of Unique values in  LANDAREA_AVG is 3528
Nunber of Unique values in  LIVINGAPARTMENTS_AVG is 1869
Nunber of Unique values in  LIVINGAREA_AVG is 5200
Nunber of Unique values in  NONLIVINGAPARTMENTS_AVG is 387
Nunber of Unique values in  NONLIVINGAREA_AVG is 3291
Nunber of Unique values in  APARTMENTS_MODE is 761
Nunber of Unique values in  BASEMENTAREA_MODE is 3842
```

```
Nunber of Unique values in  YEARS_BEGINEXPLUATATION_MODE is 222
Nunber of Unique values in  YEARS_BUILD_MODE is 155
Nunber of Unique values in  COMMONAREA_MODE is 3129
Nunber of Unique values in  ELEVATORS_MODE is 27
Nunber of Unique values in  ENTRANCES_MODE is 31
Nunber of Unique values in  FLOORSMAX_MODE is 26
Nunber of Unique values in  FLOORSMIN_MODE is 26
Nunber of Unique values in  LANDAREA_MODE is 3564
Nunber of Unique values in  LIVINGAPARTMENTS_MODE is 737
Nunber of Unique values in  LIVINGAREA_MODE is 5302
Nunber of Unique values in  NONLIVINGAPARTMENTS_MODE is 168
Nunber of Unique values in  NONLIVINGAREA_MODE is 3328
Nunber of Unique values in  APARTMENTS_MEDI is 1149
Nunber of Unique values in  BASEMENTAREA_MEDI is 3773
Nunber of Unique values in  YEARS_BEGINEXPLUATATION_MEDI is 246
Nunber of Unique values in  YEARS_BUILD_MEDI is 152
Nunber of Unique values in  COMMONAREA_MEDI is 3203
Nunber of Unique values in  ELEVATORS_MEDI is 47
Nunber of Unique values in  ENTRANCES_MEDI is 47
Nunber of Unique values in  FLOORSMAX_MEDI is 50
Nunber of Unique values in  FLOORSMIN_MEDI is 48
Nunber of Unique values in  LANDAREA_MEDI is 3561
Nunber of Unique values in  LIVINGAPARTMENTS_MEDI is 1098
Nunber of Unique values in  LIVINGAREA_MEDI is 5282
Nunber of Unique values in  NONLIVINGAPARTMENTS_MEDI is 215
Nunber of Unique values in  NONLIVINGAREA_MEDI is 3324
Nunber of Unique values in  FONDKAPREMONT_MODE is 5
Nunber of Unique values in  HOUSETYPE_MODE is 4
Nunber of Unique values in  TOTALAREA_MODE is 5117
Nunber of Unique values in  WALLSMATERIAL_MODE is 8
Nunber of Unique values in  EMERGENCYSTATE_MODE is 3
Nunber of Unique values in  OBS_30_CNT_SOCIAL_CIRCLE is 34
Nunber of Unique values in  DEF_30_CNT_SOCIAL_CIRCLE is 11
Nunber of Unique values in  OBS_60_CNT_SOCIAL_CIRCLE is 34
Nunber of Unique values in  DEF_60_CNT_SOCIAL_CIRCLE is 10
Nunber of Unique values in  DAYS_LAST_PHONE_CHANGE is 3774
Nunber of Unique values in  FLAG_DOCUMENT_2 is 2
Nunber of Unique values in  FLAG_DOCUMENT_3 is 2
Nunber of Unique values in  FLAG_DOCUMENT_4 is 2
Nunber of Unique values in  FLAG_DOCUMENT_5 is 2
Nunber of Unique values in  FLAG_DOCUMENT_6 is 2
Nunber of Unique values in  FLAG_DOCUMENT_7 is 2
Nunber of Unique values in  FLAG_DOCUMENT_8 is 2
Nunber of Unique values in  FLAG_DOCUMENT_9 is 2
Nunber of Unique values in  FLAG_DOCUMENT_10 is 2
Nunber of Unique values in  FLAG_DOCUMENT_11 is 2
Nunber of Unique values in  FLAG_DOCUMENT_12 is 2
Nunber of Unique values in  FLAG_DOCUMENT_13 is 2
Nunber of Unique values in  FLAG_DOCUMENT_14 is 2
Nunber of Unique values in  FLAG_DOCUMENT_15 is 2
Nunber of Unique values in  FLAG_DOCUMENT_16 is 2
Nunber of Unique values in  FLAG_DOCUMENT_17 is 2
Nunber of Unique values in  FLAG_DOCUMENT_18 is 2
Nunber of Unique values in  FLAG_DOCUMENT_19 is 2
Nunber of Unique values in  FLAG_DOCUMENT_20 is 2
Nunber of Unique values in  FLAG_DOCUMENT_21 is 2
Nunber of Unique values in  AMT_REQ_CREDIT_BUREAU_HOUR is 6
Nunber of Unique values in  AMT_REQ_CREDIT_BUREAU_DAY is 10
Nunber of Unique values in  AMT_REQ_CREDIT_BUREAU_WEEK is 10
Nunber of Unique values in  AMT_REQ_CREDIT_BUREAU_MON is 25
```

```
Nunber of Unique values in  AMT_REQ_CREDIT_BUREAU_QRT is 12
Nunber of Unique values in  AMT_REQ_CREDIT_BUREAU_YEAR is 26
```

In [177…
```
datasets["application_train"].isnull().head()
```

Out[177]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OW
0	False	False	False	False	False	
1	False	False	False	False	False	
2	False	False	False	False	False	
3	False	False	False	False	False	
4	False	False	False	False	False	

5 rows × 122 columns

In [178…
```
s = datasets["application_train"].isnull().sum()
```

In [179…
```
print("Shows all the coulumns ")
s[s!=0]
```

Out[179]:
```
Shows all the coulumns
AMT_ANNUITY                      12
AMT_GOODS_PRICE                 278
NAME_TYPE_SUITE                1292
OWN_CAR_AGE                  202929
OCCUPATION_TYPE               96391
                              ...
AMT_REQ_CREDIT_BUREAU_DAY     41519
AMT_REQ_CREDIT_BUREAU_WEEK    41519
AMT_REQ_CREDIT_BUREAU_MON     41519
AMT_REQ_CREDIT_BUREAU_QRT     41519
AMT_REQ_CREDIT_BUREAU_YEAR    41519
Length: 67, dtype: int64
```

Splitting categorical and numerical features

In [180…
```
##num_columns = datasets["application_train"].select_dtypes(include = ['int64',
##print(f"\n Numerical columns in Application_train : {list(num_columns)}")
categorical_columns = []
for col in datasets["application_train"]:
        if datasets["application_train"][col].dtype == 'object' or datasets["ap
            categorical_columns.append(col)
print("Categorical columns in dataset: Application train", categorical_columns)
```

```
Categorical columns in dataset: Application train ['NAME_CONTRACT_TYPE', 'CODE
_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'NAME_TYPE_SUITE', 'NAME_INCOME_T
YPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUP
ATION_TYPE', 'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE', 'FONDKAPREMONT
_MODE', 'HOUSETYPE_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE']
```

In [181…
```
num_columns = []
for col in datasets["application_train"]:
        if datasets["application_train"][col].dtype == 'int64' or datasets["app
            num_columns.append(col)
print("Numerical columns in dataset: Application train", num_columns)
```

```
Numerical columns in dataset: Application train ['SK_ID_CURR', 'TARGET', 'CNT_
CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATIO
N', 'DAYS_ID_PUBLISH', 'OWN_CAR_AGE', 'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WO
RK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'CNT_FAM_MEMBERS',
'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY', 'HOUR_APPR_PROCESS_STAR
T', 'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_N
OT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CIT
Y_NOT_WORK_CITY', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'APARTMENTS_
AVG', 'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BUILD_AVG', 'C
OMMONAREA_AVG', 'ELEVATORS_AVG', 'ENTRANCES_AVG', 'FLOORSMAX_AVG', 'FLOORSMIN_
AVG', 'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG', 'NONLIVINGAPAR
TMENTS_AVG', 'NONLIVINGAREA_AVG', 'APARTMENTS_MODE', 'BASEMENTAREA_MODE', 'YEA
RS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE', 'COMMONAREA_MODE', 'ELEVATORS_
MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE', 'FLOORSMIN_MODE', 'LANDAREA_MODE',
'LIVINGAPARTMENTS_MODE', 'LIVINGAREA_MODE', 'NONLIVINGAPARTMENTS_MODE', 'NONLI
VINGAREA_MODE', 'APARTMENTS_MEDI', 'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATI
ON_MEDI', 'YEARS_BUILD_MEDI', 'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_
MEDI', 'FLOORSMAX_MEDI', 'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_
MEDI', 'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI', 'T
OTALAREA_MODE', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_6
0_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'F
LAG_DOCUMENT_2', 'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLA
G_DOCUMENT_6', 'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_
DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13', 'FLA
G_DOCUMENT_14', 'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'F
LAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21',
'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUR
EAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_
CREDIT_BUREAU_YEAR']
```

Splitting Application train data

In [182…

```
X = datasets["application_train"].drop(['TARGET'], axis = 1)
y = datasets["application_train"]["TARGET"]
X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X, y, test_size=0.3
X_train_1, X_valid_1, y_train_1, y_valid_1 = train_test_split(X_train_1, y_trai
print(f"Shape of X train: {X_train_1.shape}")
print(f"Shape of X validation: {X_valid_1.shape}")
print(f"Shape of X test: {X_test_1.shape}")
```

```
Shape of X train: (150679, 121)
Shape of X validation: (64578, 121)
Shape of X test: (92254, 121)
```

In [183…

```
X_train_1.info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150679 entries, 285945 to 43103
Data columns (total 121 columns):
 #    Column                          Non-Null Count   Dtype
---   ------                          --------------   -----
 0    SK_ID_CURR                      150679 non-null  int64
 1    NAME_CONTRACT_TYPE              150679 non-null  object
 2    CODE_GENDER                     150679 non-null  object
 3    FLAG_OWN_CAR                    150679 non-null  object
 4    FLAG_OWN_REALTY                 150679 non-null  object
 5    CNT_CHILDREN                    150679 non-null  int64
 6    AMT_INCOME_TOTAL                150679 non-null  float64
 7    AMT_CREDIT                      150679 non-null  float64
 8    AMT_ANNUITY                     150674 non-null  float64
 9    AMT_GOODS_PRICE                 150549 non-null  float64
 10   NAME_TYPE_SUITE                 150028 non-null  object
 11   NAME_INCOME_TYPE                150679 non-null  object
 12   NAME_EDUCATION_TYPE             150679 non-null  object
 13   NAME_FAMILY_STATUS              150679 non-null  object
 14   NAME_HOUSING_TYPE               150679 non-null  object
 15   REGION_POPULATION_RELATIVE      150679 non-null  float64
 16   DAYS_BIRTH                      150679 non-null  int64
 17   DAYS_EMPLOYED                   150679 non-null  int64
 18   DAYS_REGISTRATION               150679 non-null  float64
 19   DAYS_ID_PUBLISH                 150679 non-null  int64
 20   OWN_CAR_AGE                     51392 non-null   float64
 21   FLAG_MOBIL                      150679 non-null  int64
 22   FLAG_EMP_PHONE                  150679 non-null  int64
 23   FLAG_WORK_PHONE                 150679 non-null  int64
 24   FLAG_CONT_MOBILE                150679 non-null  int64
 25   FLAG_PHONE                      150679 non-null  int64
 26   FLAG_EMAIL                      150679 non-null  int64
 27   OCCUPATION_TYPE                 103516 non-null  object
 28   CNT_FAM_MEMBERS                 150678 non-null  float64
 29   REGION_RATING_CLIENT            150679 non-null  int64
 30   REGION_RATING_CLIENT_W_CITY     150679 non-null  int64
 31   WEEKDAY_APPR_PROCESS_START      150679 non-null  object
 32   HOUR_APPR_PROCESS_START         150679 non-null  int64
 33   REG_REGION_NOT_LIVE_REGION      150679 non-null  int64
 34   REG_REGION_NOT_WORK_REGION      150679 non-null  int64
 35   LIVE_REGION_NOT_WORK_REGION     150679 non-null  int64
 36   REG_CITY_NOT_LIVE_CITY          150679 non-null  int64
 37   REG_CITY_NOT_WORK_CITY          150679 non-null  int64
 38   LIVE_CITY_NOT_WORK_CITY         150679 non-null  int64
 39   ORGANIZATION_TYPE               150679 non-null  object
 40   EXT_SOURCE_1                    65865 non-null   float64
 41   EXT_SOURCE_2                    150354 non-null  float64
 42   EXT_SOURCE_3                    120847 non-null  float64
 43   APARTMENTS_AVG                  74071 non-null   float64
 44   BASEMENTAREA_AVG                62475 non-null   float64
 45   YEARS_BEGINEXPLUATATION_AVG     77128 non-null   float64
 46   YEARS_BUILD_AVG                 50387 non-null   float64
 47   COMMONAREA_AVG                  45378 non-null   float64
 48   ELEVATORS_AVG                   70299 non-null   float64
 49   ENTRANCES_AVG                   74743 non-null   float64
 50   FLOORSMAX_AVG                   75622 non-null   float64
 51   FLOORSMIN_AVG                   48433 non-null   float64
 52   LANDAREA_AVG                    61128 non-null   float64
 53   LIVINGAPARTMENTS_AVG            47641 non-null   float64
 54   LIVINGAREA_AVG                  74937 non-null   float64
```

55	NONLIVINGAPARTMENTS_AVG	46033 non-null	float64
56	NONLIVINGAREA_AVG	67458 non-null	float64
57	APARTMENTS_MODE	74071 non-null	float64
58	BASEMENTAREA_MODE	62475 non-null	float64
59	YEARS_BEGINEXPLUATATION_MODE	77128 non-null	float64
60	YEARS_BUILD_MODE	50387 non-null	float64
61	COMMONAREA_MODE	45378 non-null	float64
62	ELEVATORS_MODE	70299 non-null	float64
63	ENTRANCES_MODE	74743 non-null	float64
64	FLOORSMAX_MODE	75622 non-null	float64
65	FLOORSMIN_MODE	48433 non-null	float64
66	LANDAREA_MODE	61128 non-null	float64
67	LIVINGAPARTMENTS_MODE	47641 non-null	float64
68	LIVINGAREA_MODE	74937 non-null	float64
69	NONLIVINGAPARTMENTS_MODE	46033 non-null	float64
70	NONLIVINGAREA_MODE	67458 non-null	float64
71	APARTMENTS_MEDI	74071 non-null	float64
72	BASEMENTAREA_MEDI	62475 non-null	float64
73	YEARS_BEGINEXPLUATATION_MEDI	77128 non-null	float64
74	YEARS_BUILD_MEDI	50387 non-null	float64
75	COMMONAREA_MEDI	45378 non-null	float64
76	ELEVATORS_MEDI	70299 non-null	float64
77	ENTRANCES_MEDI	74743 non-null	float64
78	FLOORSMAX_MEDI	75622 non-null	float64
79	FLOORSMIN_MEDI	48433 non-null	float64
80	LANDAREA_MEDI	61128 non-null	float64
81	LIVINGAPARTMENTS_MEDI	47641 non-null	float64
82	LIVINGAREA_MEDI	74937 non-null	float64
83	NONLIVINGAPARTMENTS_MEDI	46033 non-null	float64
84	NONLIVINGAREA_MEDI	67458 non-null	float64
85	FONDKAPREMONT_MODE	47623 non-null	object
86	HOUSETYPE_MODE	75039 non-null	object
87	TOTALAREA_MODE	77874 non-null	float64
88	WALLSMATERIAL_MODE	74015 non-null	object
89	EMERGENCYSTATE_MODE	79204 non-null	object
90	OBS_30_CNT_SOCIAL_CIRCLE	150203 non-null	float64
91	DEF_30_CNT_SOCIAL_CIRCLE	150203 non-null	float64
92	OBS_60_CNT_SOCIAL_CIRCLE	150203 non-null	float64
93	DEF_60_CNT_SOCIAL_CIRCLE	150203 non-null	float64
94	DAYS_LAST_PHONE_CHANGE	150679 non-null	float64
95	FLAG_DOCUMENT_2	150679 non-null	int64
96	FLAG_DOCUMENT_3	150679 non-null	int64
97	FLAG_DOCUMENT_4	150679 non-null	int64
98	FLAG_DOCUMENT_5	150679 non-null	int64
99	FLAG_DOCUMENT_6	150679 non-null	int64
100	FLAG_DOCUMENT_7	150679 non-null	int64
101	FLAG_DOCUMENT_8	150679 non-null	int64
102	FLAG_DOCUMENT_9	150679 non-null	int64
103	FLAG_DOCUMENT_10	150679 non-null	int64
104	FLAG_DOCUMENT_11	150679 non-null	int64
105	FLAG_DOCUMENT_12	150679 non-null	int64
106	FLAG_DOCUMENT_13	150679 non-null	int64
107	FLAG_DOCUMENT_14	150679 non-null	int64
108	FLAG_DOCUMENT_15	150679 non-null	int64
109	FLAG_DOCUMENT_16	150679 non-null	int64
110	FLAG_DOCUMENT_17	150679 non-null	int64
111	FLAG_DOCUMENT_18	150679 non-null	int64
112	FLAG_DOCUMENT_19	150679 non-null	int64
113	FLAG_DOCUMENT_20	150679 non-null	int64
114	FLAG_DOCUMENT_21	150679 non-null	int64

```
 115  AMT_REQ_CREDIT_BUREAU_HOUR    130233 non-null  float64
 116  AMT_REQ_CREDIT_BUREAU_DAY     130233 non-null  float64
 117  AMT_REQ_CREDIT_BUREAU_WEEK    130233 non-null  float64
 118  AMT_REQ_CREDIT_BUREAU_MON     130233 non-null  float64
 119  AMT_REQ_CREDIT_BUREAU_QRT     130233 non-null  float64
 120  AMT_REQ_CREDIT_BUREAU_YEAR    130233 non-null  float64
dtypes: float64(65), int64(40), object(16)
memory usage: 140.2+ MB
```

Missing data for application train

In [184…

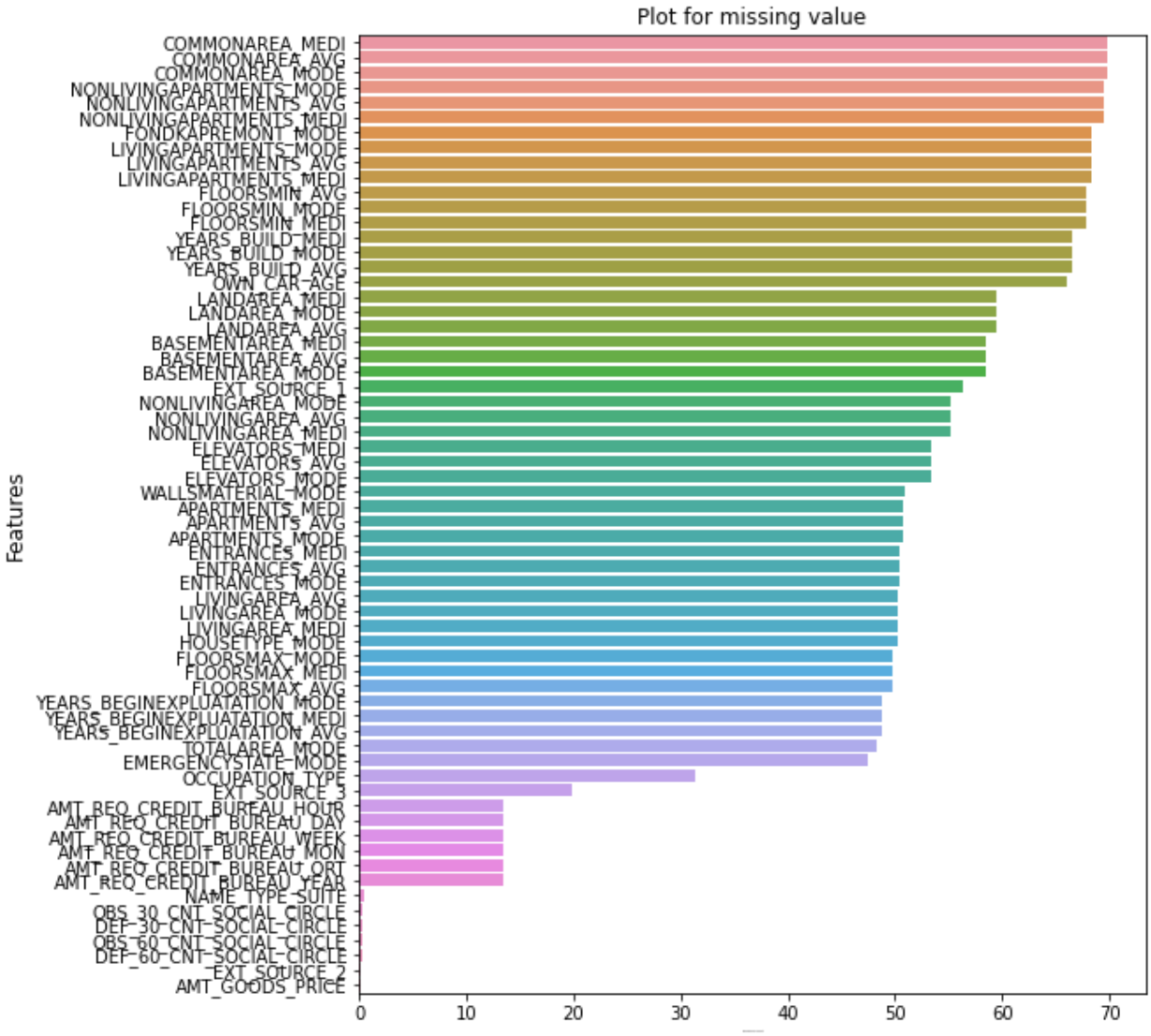
```
def missing_value_info_plot(df, df_name):
    percent = (df.isnull().sum()/df.isnull().count()*100).sort_values(ascending
    sum_missing = df.isna().sum().sort_values(ascending = False)
    missing_data  = pd.concat([percent, sum_missing], axis=1, keys=['Percent',
    missing_data = missing_data[missing_data['Percent'] > 0]
    if len(missing_data) > 0:
        a,x=plt.subplots(figsize=(8,10))
        f = sns.barplot(missing_data['Percent'],missing_data.index)
        plt.title('Plot for missing value')
        plt.xlabel('Missing values percent', fontsize = 1)
        plt.ylabel('Features',fontsize = 12)
    return missing_data.head(20)
```

In [185…

```
missing_value_info_plot(datasets['application_train'], 'application_train')
```

Out[185]:

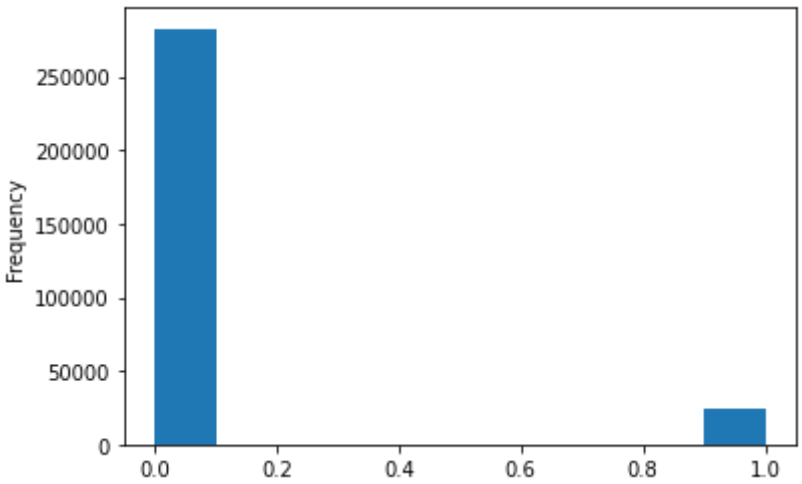
	Percent	Train Missing Count
COMMONAREA_MEDI	69.87	214865
COMMONAREA_AVG	69.87	214865
COMMONAREA_MODE	69.87	214865
NONLIVINGAPARTMENTS_MODE	69.43	213514
NONLIVINGAPARTMENTS_AVG	69.43	213514
NONLIVINGAPARTMENTS_MEDI	69.43	213514
FONDKAPREMONT_MODE	68.39	210295
LIVINGAPARTMENTS_MODE	68.35	210199
LIVINGAPARTMENTS_AVG	68.35	210199
LIVINGAPARTMENTS_MEDI	68.35	210199
FLOORSMIN_AVG	67.85	208642
FLOORSMIN_MODE	67.85	208642
FLOORSMIN_MEDI	67.85	208642
YEARS_BUILD_MEDI	66.50	204488
YEARS_BUILD_MODE	66.50	204488
YEARS_BUILD_AVG	66.50	204488
OWN_CAR_AGE	65.99	202929
LANDAREA_MEDI	59.38	182590
LANDAREA_MODE	59.38	182590
LANDAREA_AVG	59.38	182590



Distribution of the target column

In [186…

```
datasets["application_train"]['TARGET'].astype(int).plot.hist();
```



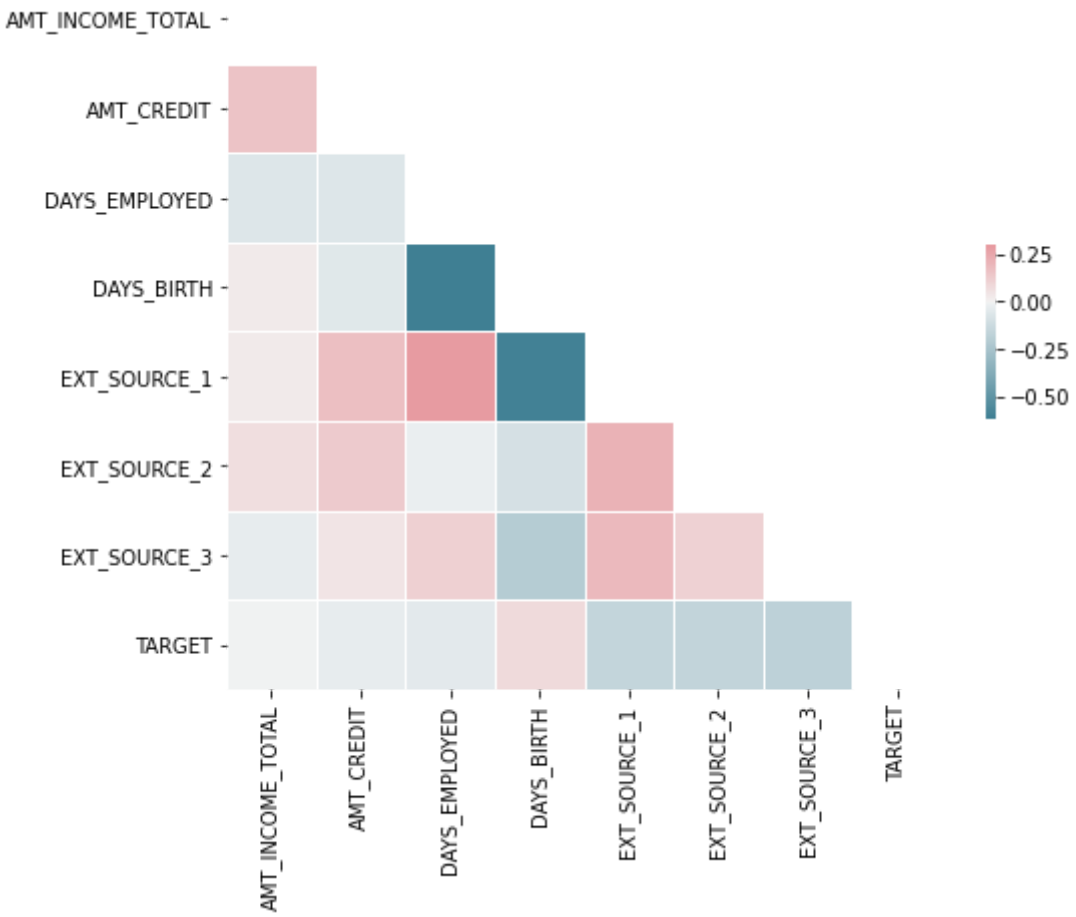
Correlation with the target column

In [187…
```
correlations = datasets["application_train"].corr()['TARGET'].sort_values()
print('Most Positive Correlations:\n', correlations.tail(10))
print('\nMost Negative Correlations:\n', correlations.head(10))
```

```
Most Positive Correlations:
 FLAG_DOCUMENT_3                0.044346
REG_CITY_NOT_LIVE_CITY          0.044395
FLAG_EMP_PHONE                  0.045982
REG_CITY_NOT_WORK_CITY          0.050994
DAYS_ID_PUBLISH                 0.051457
DAYS_LAST_PHONE_CHANGE          0.055218
REGION_RATING_CLIENT            0.058899
REGION_RATING_CLIENT_W_CITY     0.060893
DAYS_BIRTH                      0.078239
TARGET                          1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
 EXT_SOURCE_3                 -0.178919
EXT_SOURCE_2                 -0.160472
EXT_SOURCE_1                 -0.155317
DAYS_EMPLOYED                -0.044932
FLOORSMAX_AVG                -0.044003
FLOORSMAX_MEDI               -0.043768
FLOORSMAX_MODE               -0.043226
AMT_GOODS_PRICE              -0.039645
REGION_POPULATION_RELATIVE   -0.037227
ELEVATORS_AVG                -0.034199
Name: TARGET, dtype: float64
```

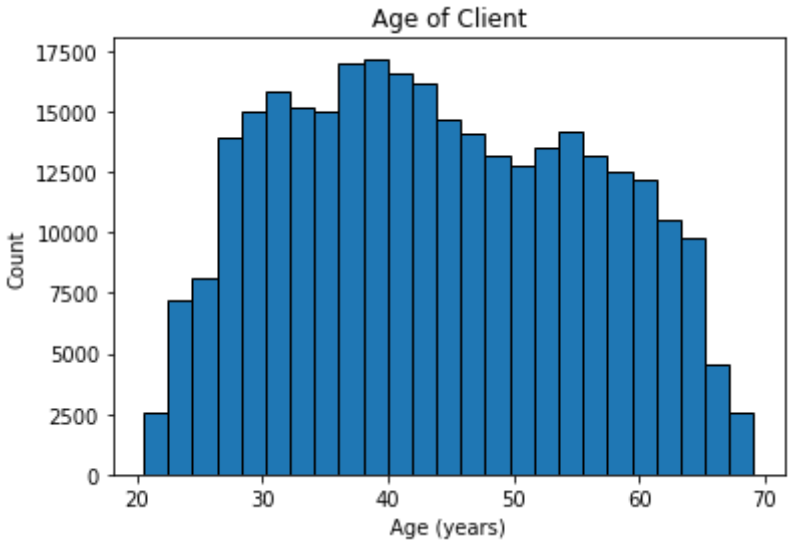
In [188…
```
numeric_features = datasets['application_train'][[
    'AMT_INCOME_TOTAL',  'AMT_CREDIT','DAYS_EMPLOYED','DAYS_BIRTH','EXT_SOURCE_
    'EXT_SOURCE_2','EXT_SOURCE_3', 'TARGET']]
```

In [189…
```
corr = numeric_features.corr()

mask = np.triu(np.ones_like(corr, dtype=np.bool))

f, ax = plt.subplots(figsize=(8, 8))

cmap = sns.diverging_palette(220, 10, as_cmap=True)

sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .2})
plt.show();
```



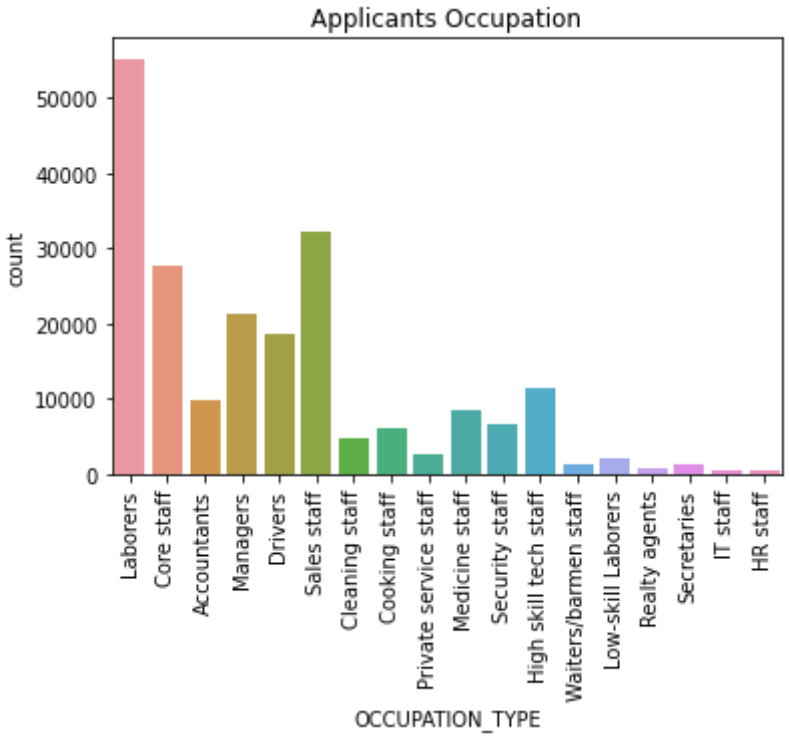
Applicants Age

In [190…
```
plt.hist(datasets["application_train"]['DAYS_BIRTH'] / -365, edgecolor = 'k', b
plt.title('Age of Client'); plt.xlabel('Age (years)'); plt.ylabel('Count');
```



Applicants occupations

In [191…
```
sns.countplot(x='OCCUPATION_TYPE', data=datasets["application_train"]);
plt.title('Applicants Occupation');
plt.xticks(rotation=90);
```

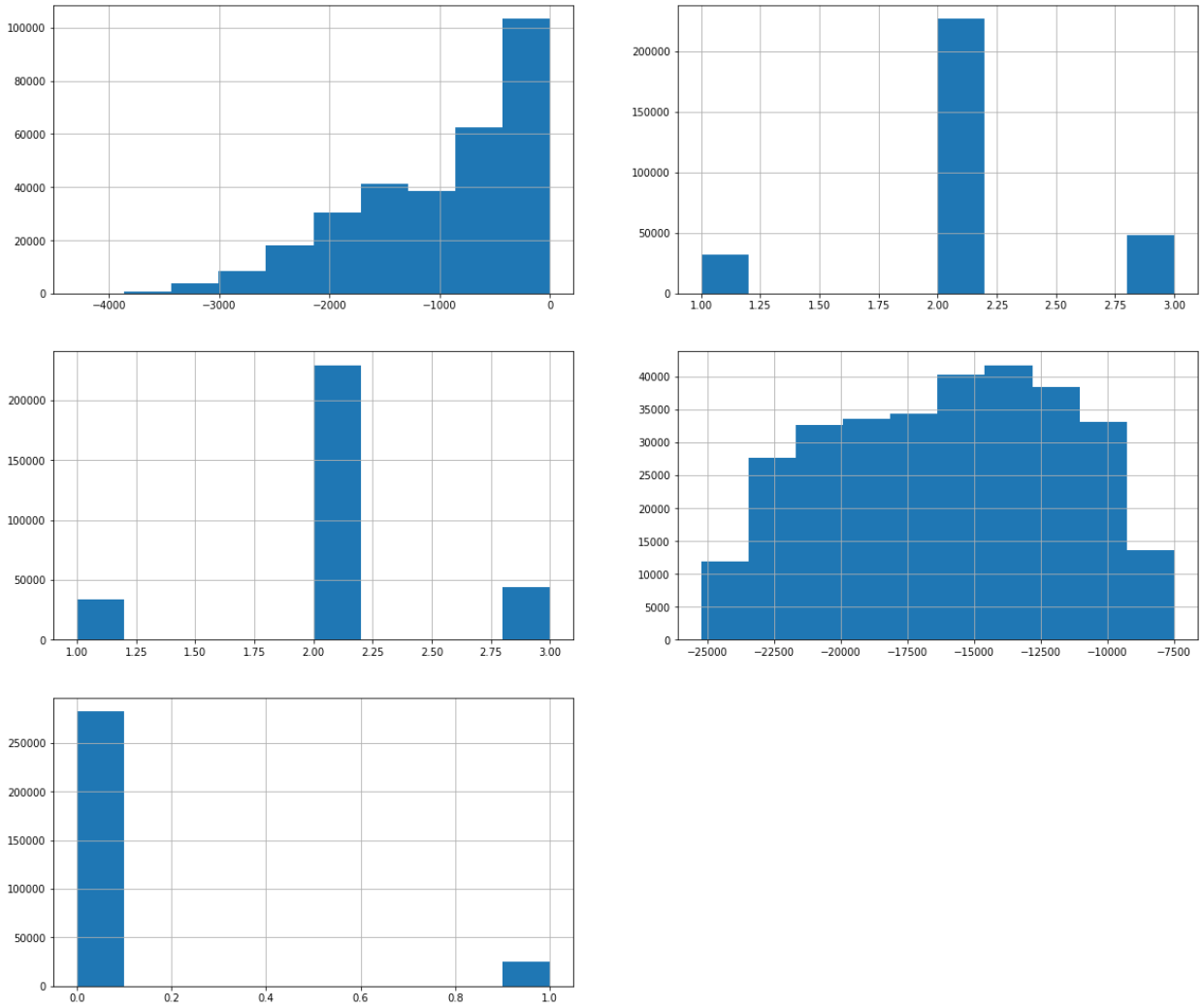


VISUAL Exploratory Data Analysis

Distribution of application train dataset

Distribution of positively correlated features

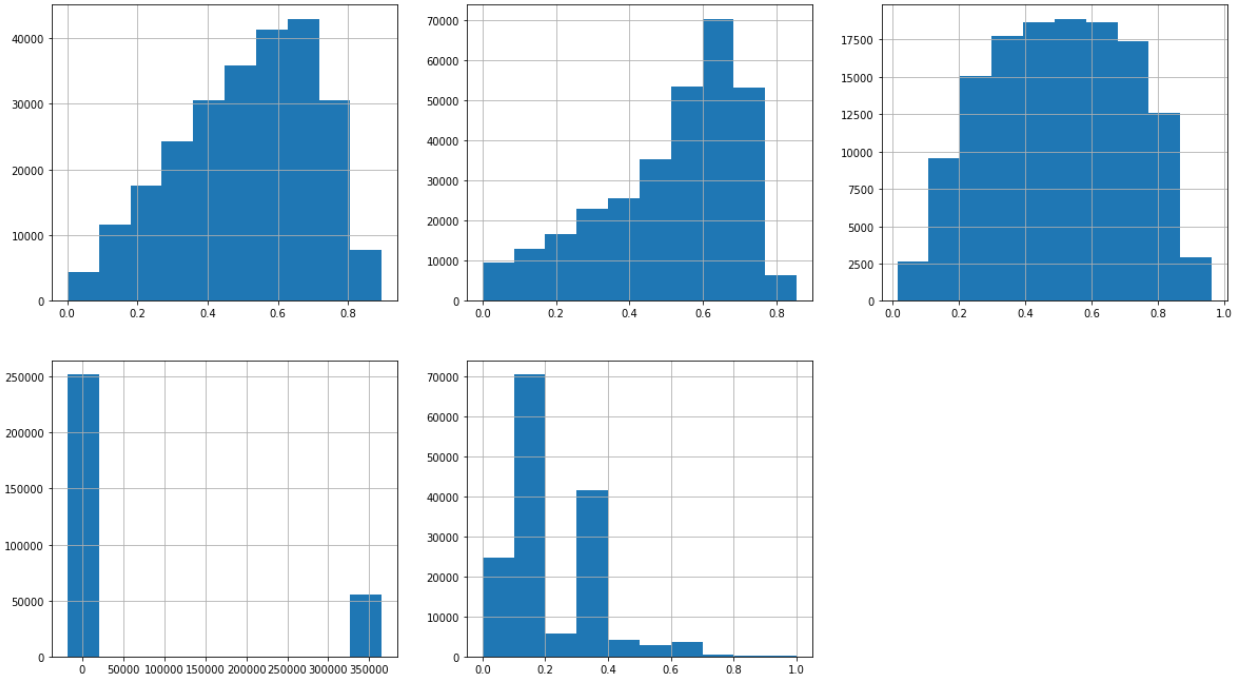
In [192…
```
corr_pos = correlations.tail(5).index.values
shape = corr_pos.shape[0]
plt.figure(figsize = (20,30))
for a,b in enumerate(corr_pos):
    plt.subplot(shape,2,a+1)
    datasets["application_train"][b].hist()
plt.show()
```



Distribution of negatively correlated features

In [193…

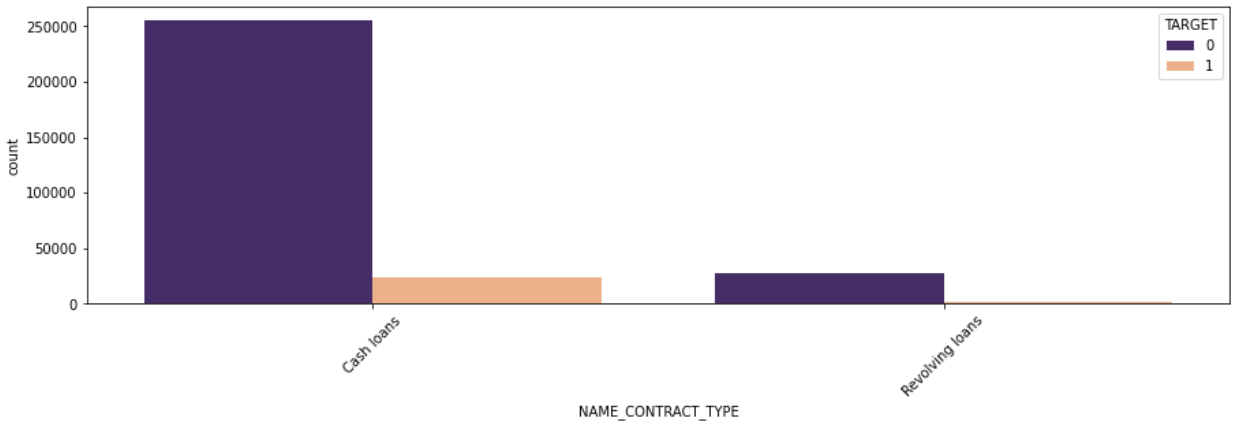
```
corr_pos = correlations.head(5).index.values
shape = corr_pos.shape[0]
plt.figure(figsize = (20,30))
for a,b in enumerate(corr_pos):
    plt.subplot(shape,3,a+1)
    datasets["application_train"][b].hist()
plt.show()
```

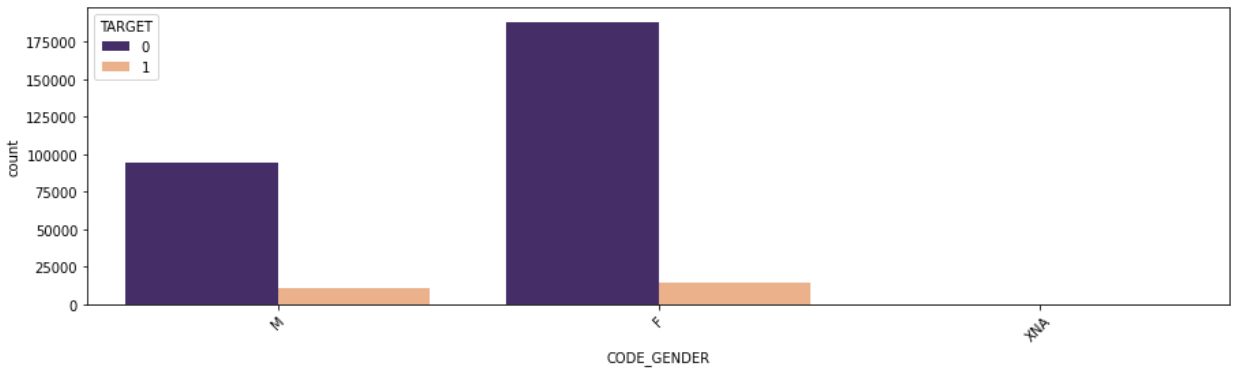


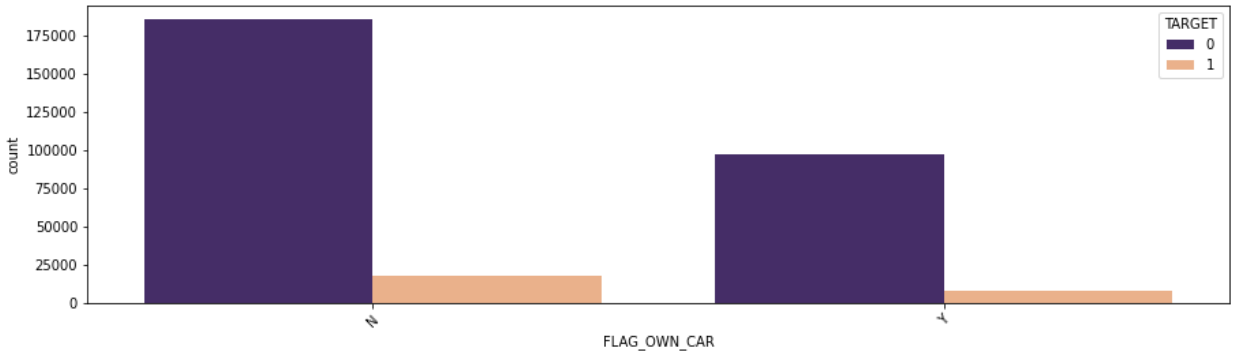
Plots comparing target variables with input features.

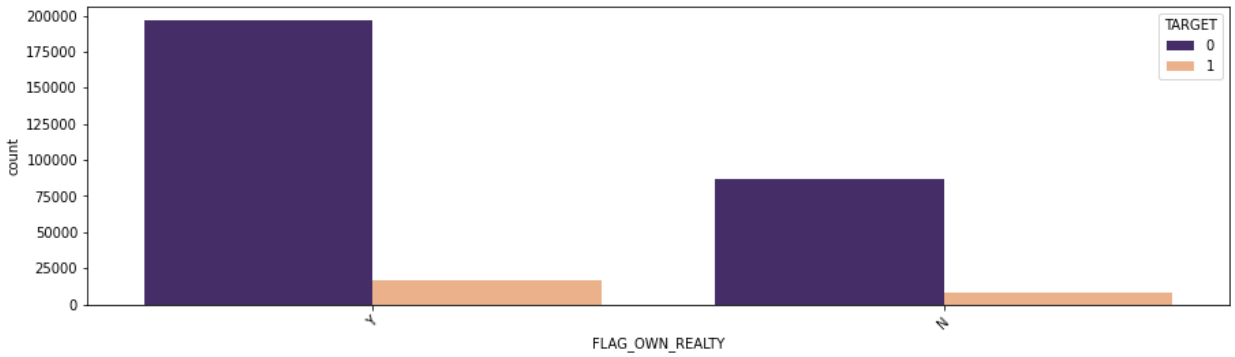
In [194…

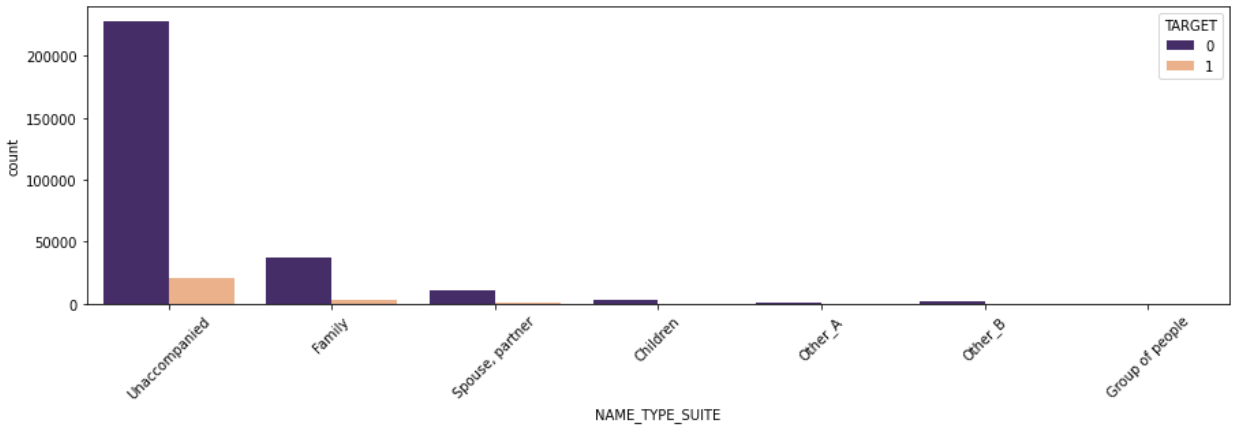
```
for categorical in categorical_columns:
    plt.figure(figsize=(15,4))
    g=plot = sns.countplot(x=datasets["application_train"][categorical],hue=dat
                           , palette= ['#432371',"#FAAE7B"])
    g.set_xticklabels(g.get_xticklabels(), rotation=45)
    plt.show()
```

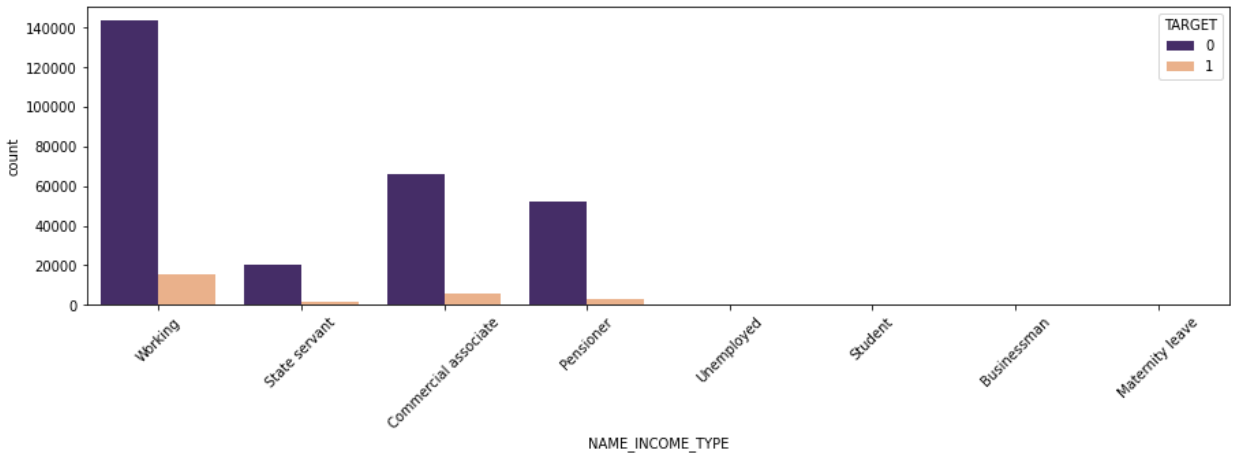


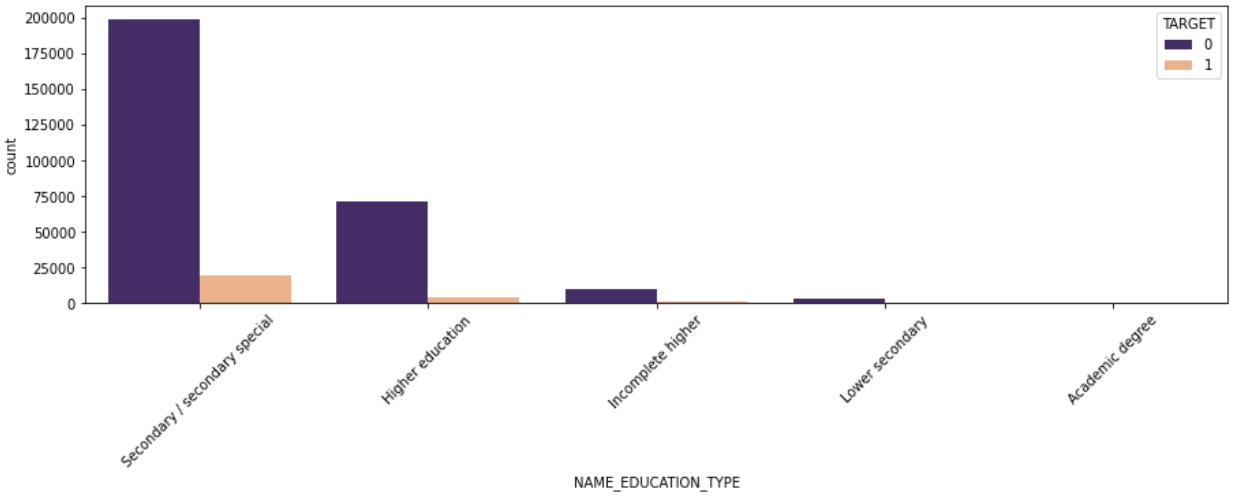


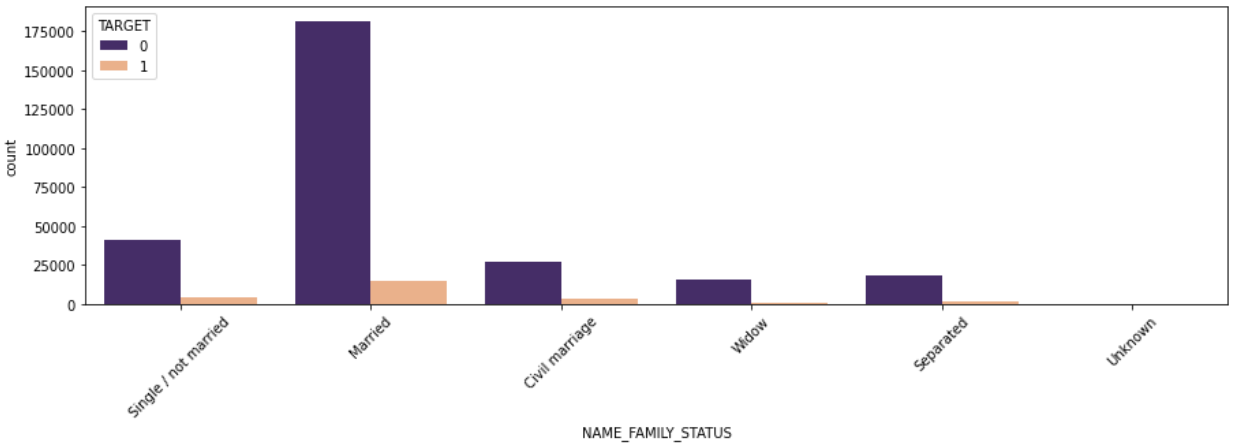

TARGET
0
1
count
175000
150000
125000
100000
75000
50000
25000
0
N
Y
FLAG_OWN_CAR

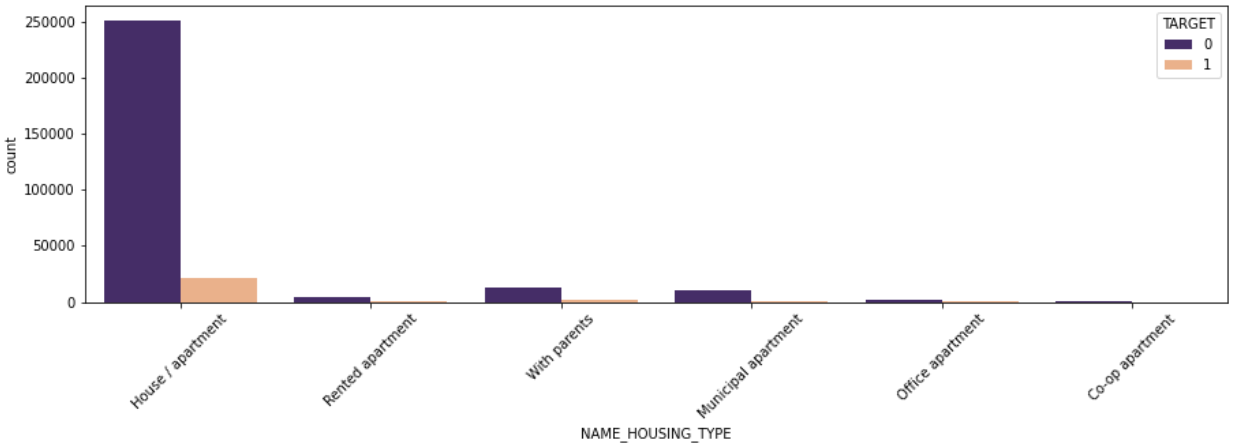

TARGET
0
1
count
200000
175000
150000
125000
100000
75000
50000
25000
0
Y
N
FLAG_OWN_REALTY

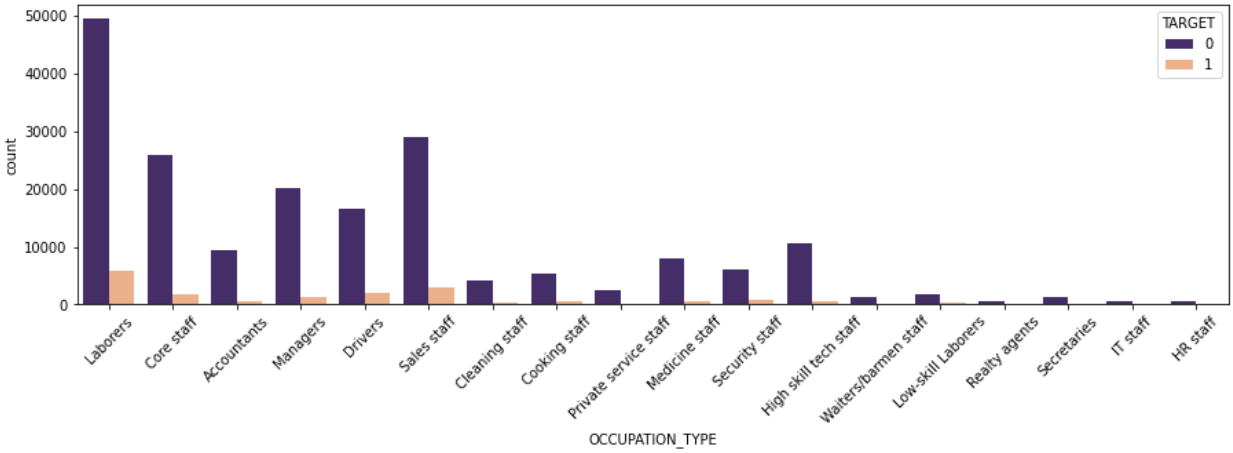

TARGET
0
1
count
200000
150000
100000
50000
0
Unaccompanied
Family
Spouse, partner
Children
Other_A
Other_B
Group of people
NAME_TYPE_SUITE

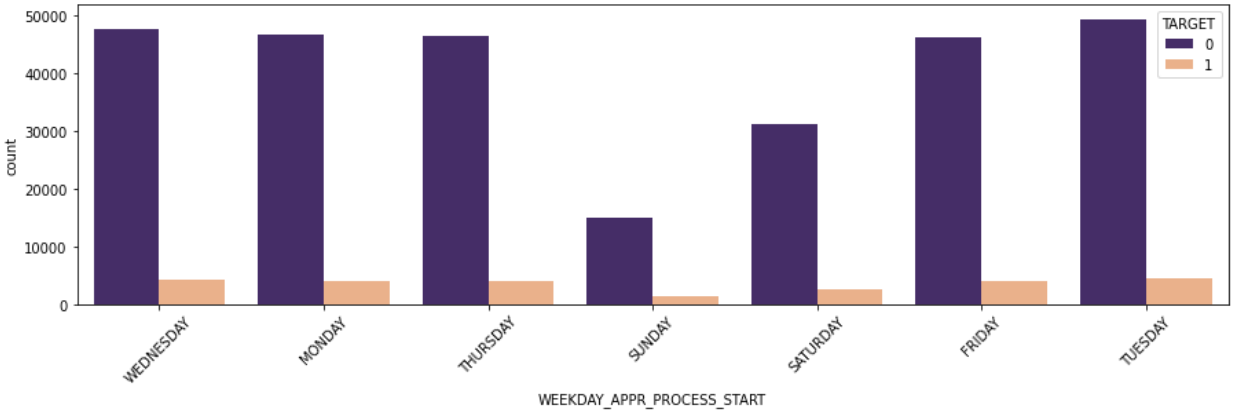

TARGET
0
1
count
140000
120000
100000
80000
60000
40000
20000
0
Working
State servant
Commercial associate
Pensioner
Unemployed
Student
Businessman
Maternity leave
NAME_INCOME_TYPE

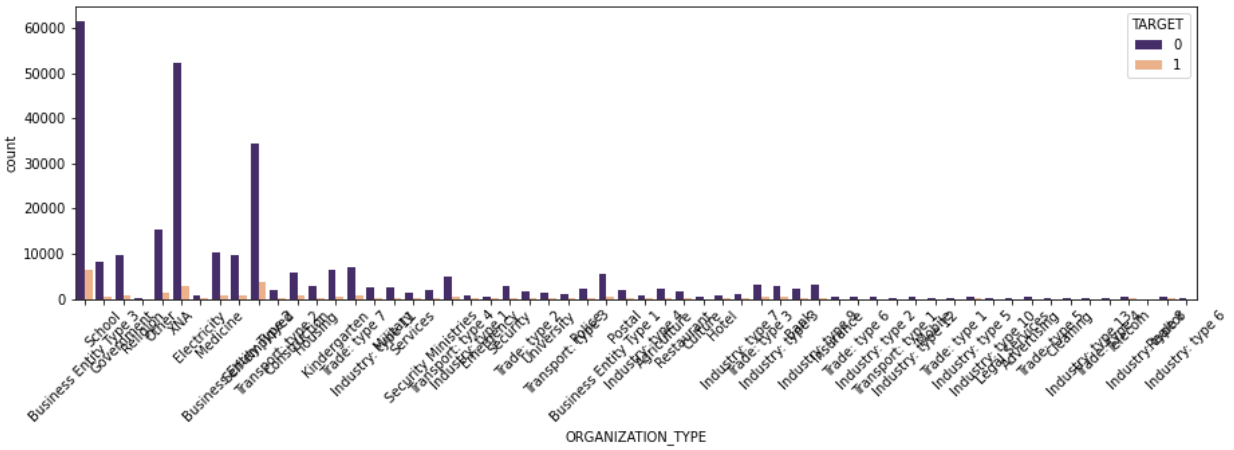

TARGET
0
1
count
200000
175000
150000
125000
100000
75000
50000
25000
0
Secondary / secondary special
Higher education
Incomplete higher
Lower secondary
Academic degree
NAME_EDUCATION_TYPE

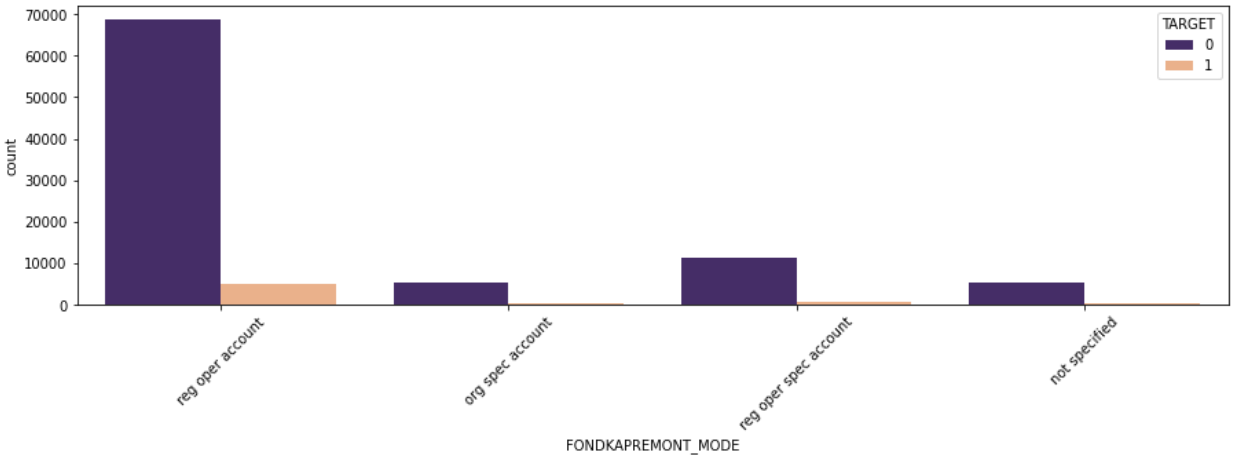

TARGET
0
1
count
175000
150000
125000
100000
75000
50000
25000
0
Single / not married
Married
Civil marriage
Widow
Separated
Unknown
NAME_FAMILY_STATUS

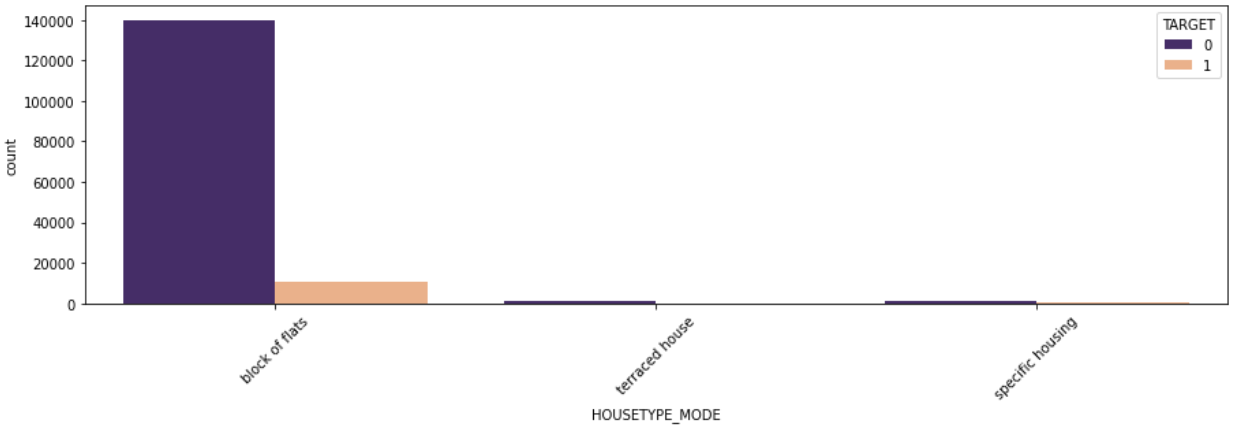

TARGET
0
1
count
250000
200000
150000
100000
50000
0
House / apartment
Rented apartment
With parents
Municipal apartment
Office apartment
Co-op apartment
NAME_HOUSING_TYPE

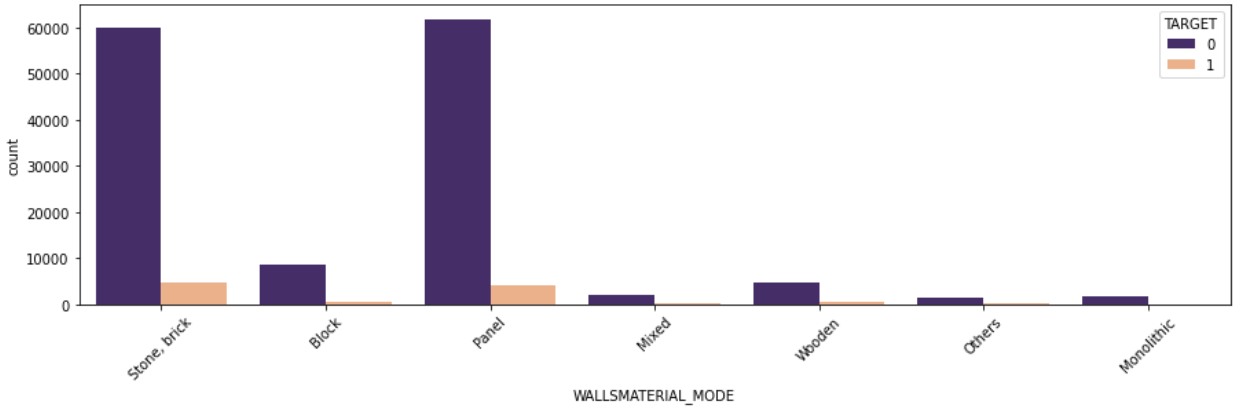

TARGET
0
1
count
50000
40000
30000
20000
10000
0
Laborers
Core staff
Accountants
Managers
Drivers
Sales staff
Cleaning staff
Cooking staff
Private service staff
Medicine staff
Security staff
High skill tech staff
Waiters/barmen staff
Low-skill Laborers
Realty agents
Secretaries
IT staff
HR staff
OCCUPATION_TYPE

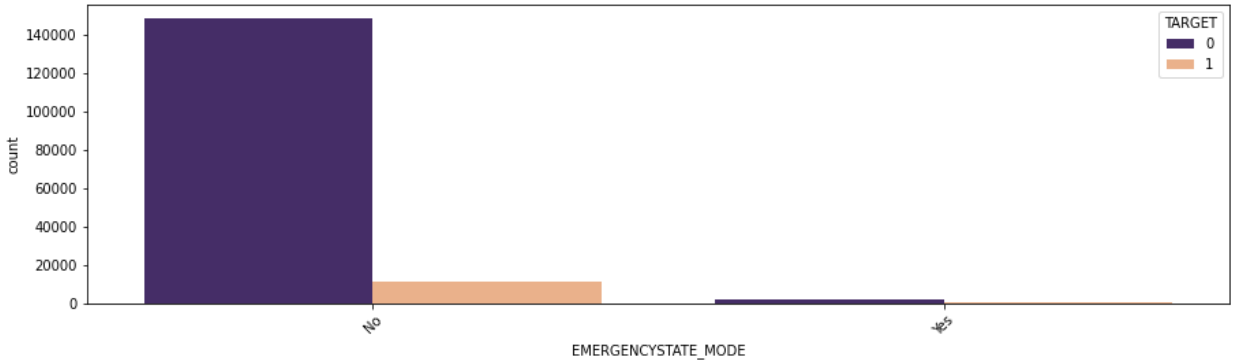

TARGET
0
1
count
WEDNESDAY
MONDAY
THURSDAY
SUNDAY
SATURDAY
FRIDAY
TUESDAY
WEEKDAY_APPR_PROCESS_START


TARGET
0
1
count
ORGANIZATION_TYPE


TARGET
0
1
count
reg oper account
org spec account
reg oper spec account
not specified
FONDKAPREMONT_MODE

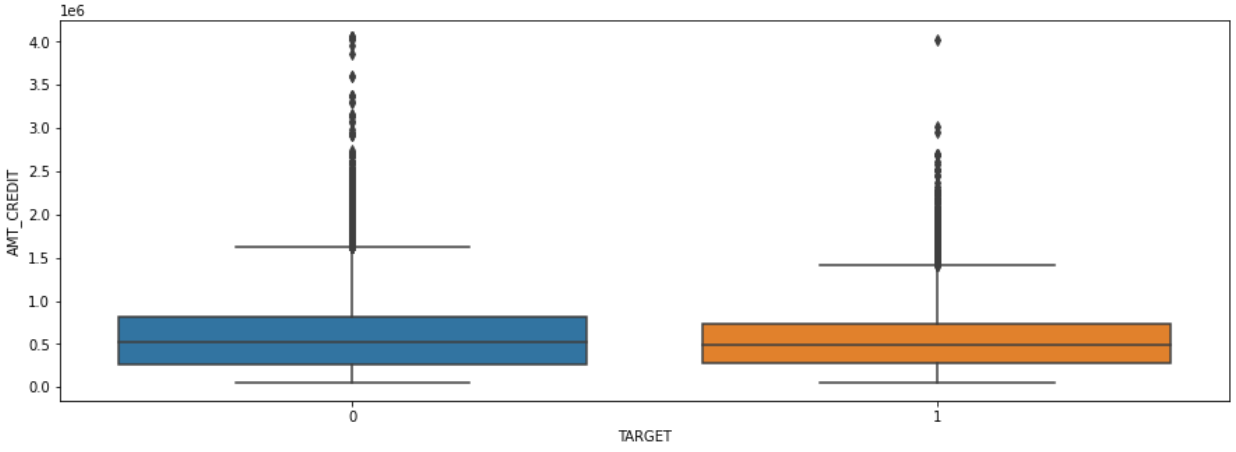

TARGET
0
1
count
block of flats
terraced house
specific housing
HOUSETYPE_MODE

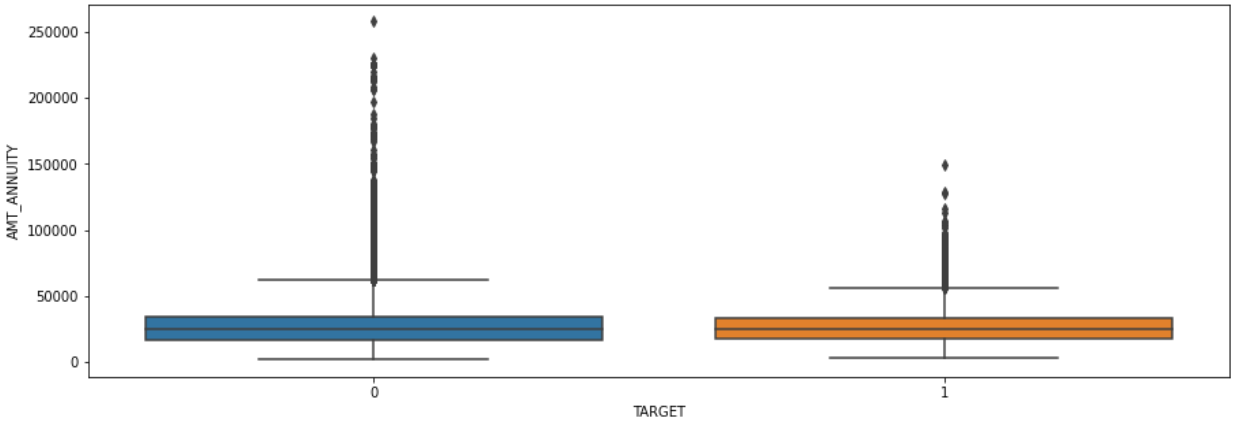


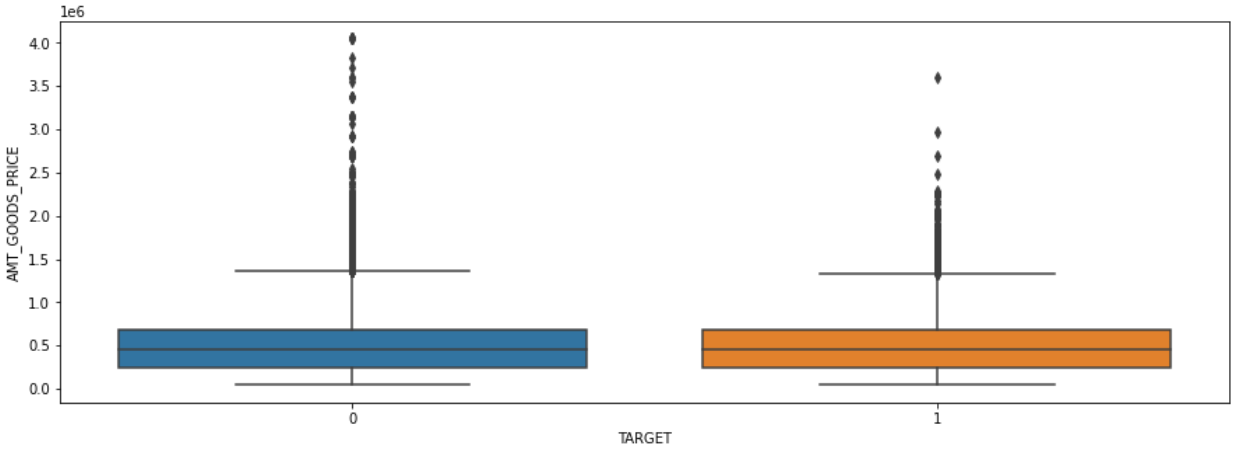


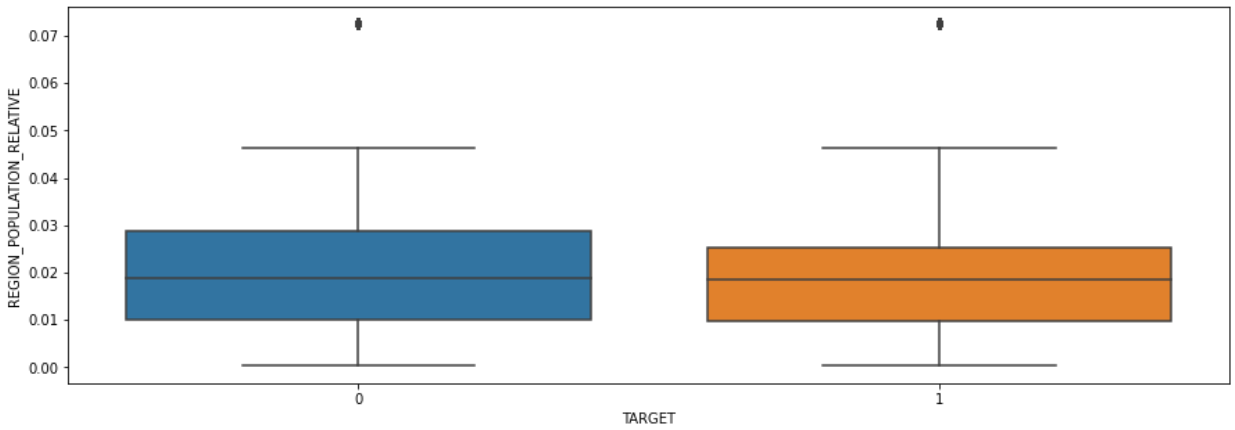
In [195…

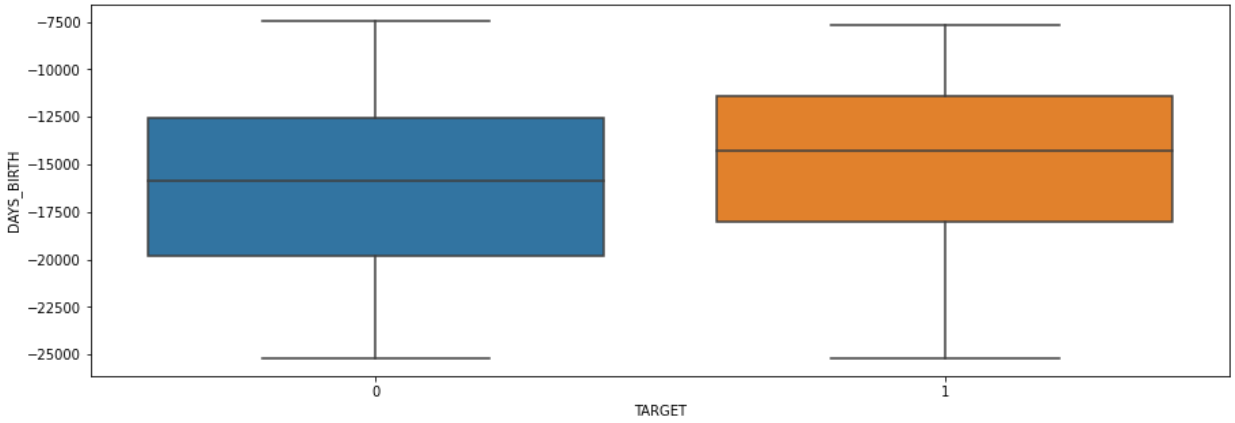
```
for numerical in num_columns[4:10]:
    plt.figure(figsize=(15,5))
    g=plot = sns.boxplot(x=datasets["application_train"]['TARGET'],  y=datasets
                         )
    plt.show()
```

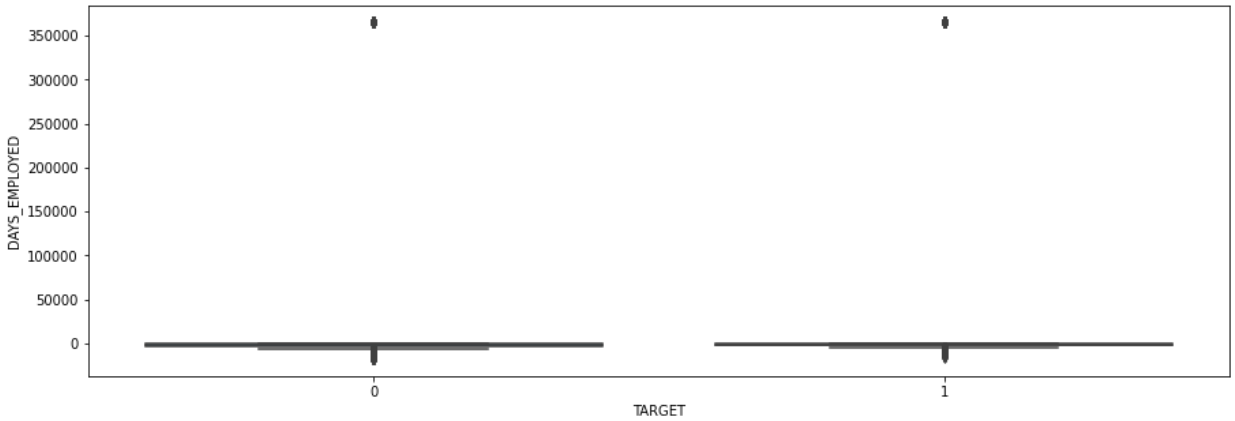








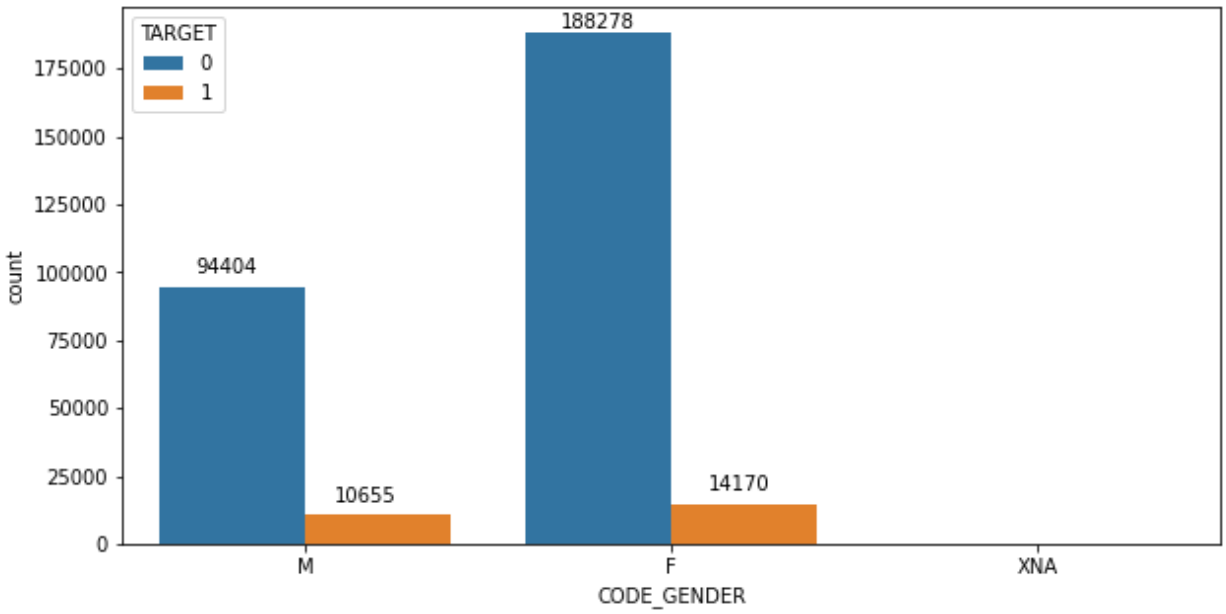




In [196…

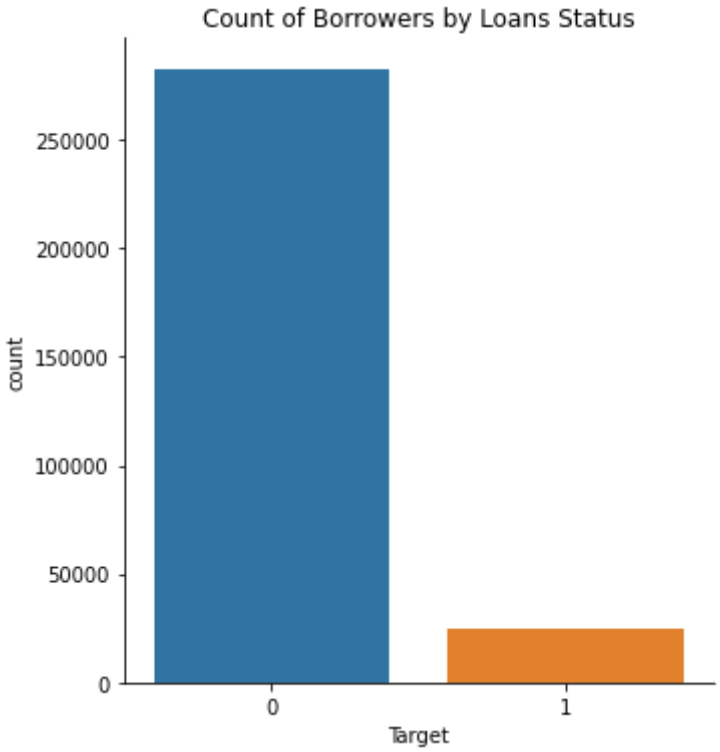
```
male_zero = datasets["application_train"].loc[(datasets["application_train"]["C
male_one =  datasets["application_train"].loc[(datasets["application_train"]["C
female_zero =  datasets["application_train"].loc[(datasets["application_train"]
female_one =  datasets["application_train"].loc[(datasets["application_train"][
```

```
plt.figure(figsize=(10,5))
sns.countplot(data=datasets["application_train"],x="CODE_GENDER", hue="TARGET",
plt.text(-0.3,100000,male_zero)
plt.text(0.08,15000,male_one)
plt.text(0.7,190000,female_zero)
plt.text(1.1,20000,female_one)
plt.show()
```

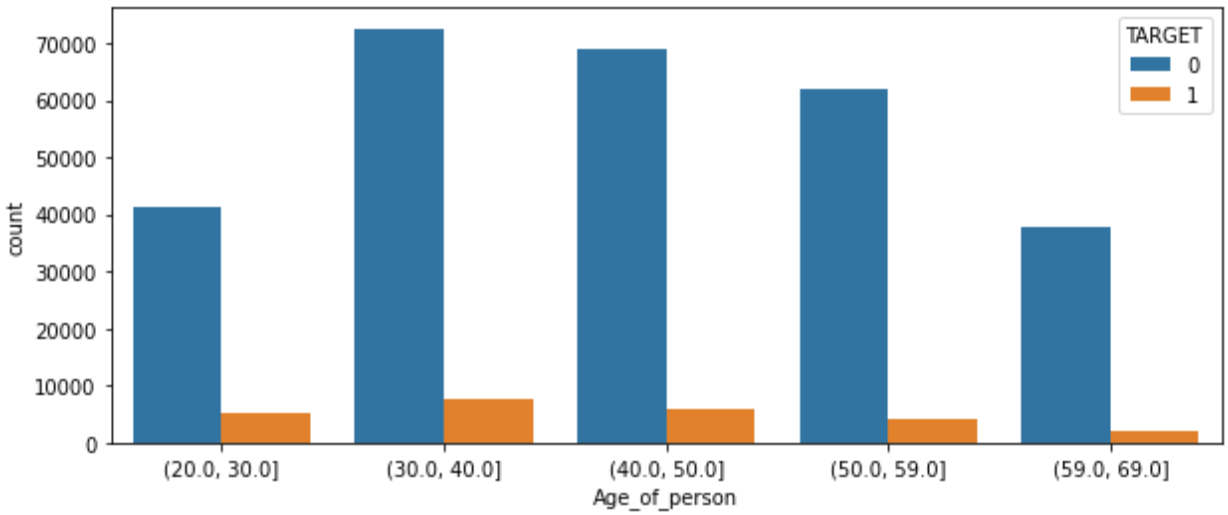


In [197…

```
import seaborn as sns
sns.catplot(data= datasets["application_train"], x = 'TARGET', kind='count' )
plt.xlabel('Target')
plt.title('Count of Borrowers by Loans Status')
plt.show()
```

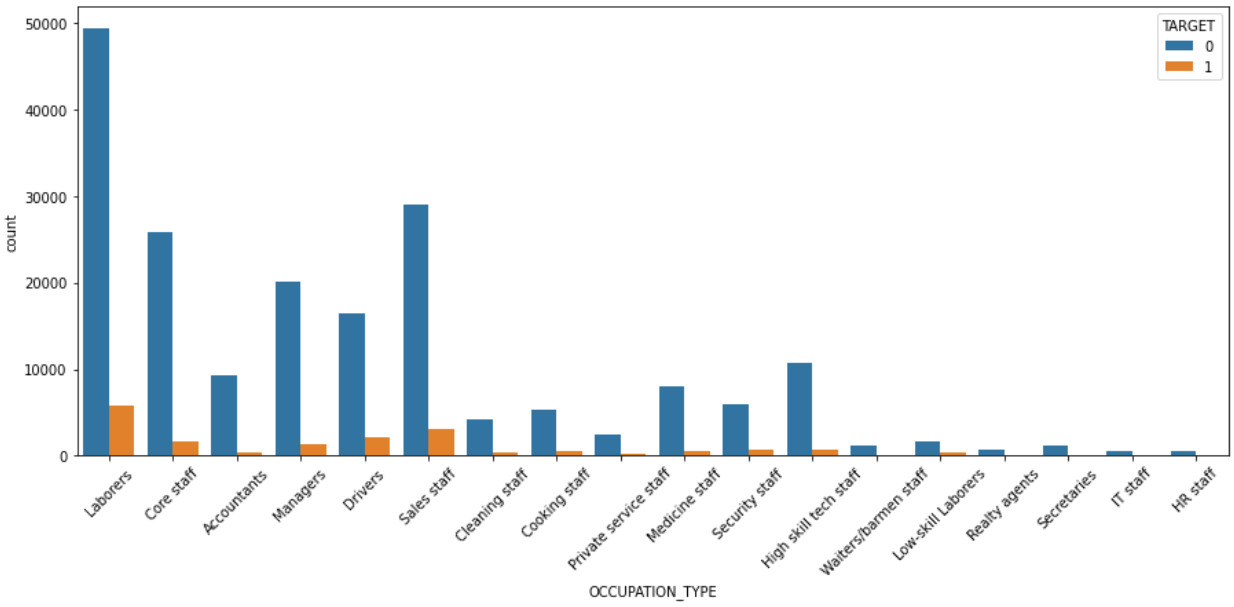


In [198…
```
plt.figure(figsize=(10,4))
sns.countplot(data=datasets["application_train"],x=pd.cut(datasets["applicatior
plt.xlabel('Age_of_person')
plt.show()
```

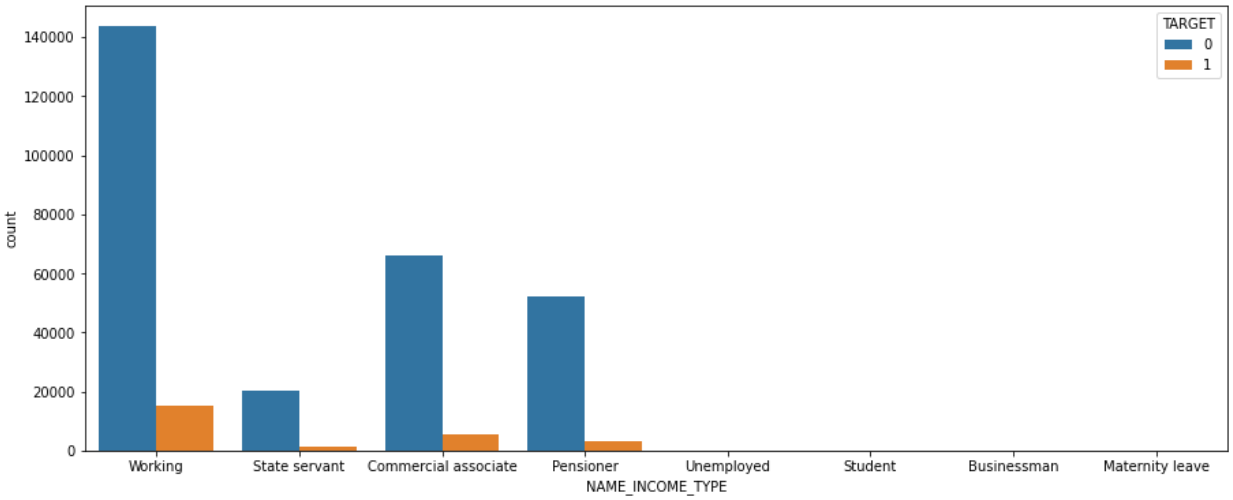


Observation: From the above graph, we can see that the people between age of 30 and 40 have maximum number who are at risk.

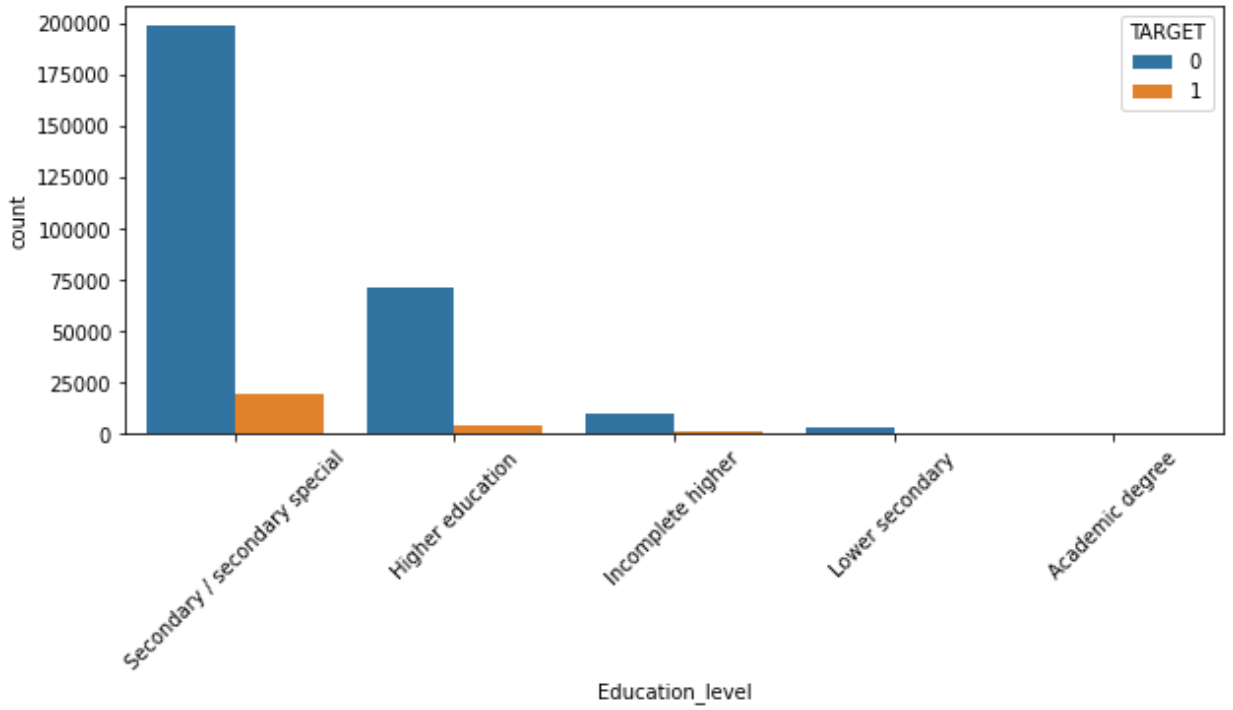
In [199…
```
plt.figure(figsize=(15,6))
g=sns.countplot(data=datasets["application_train"], x="OCCUPATION_TYPE", hue="T
plt.xlabel('OCCUPATION_TYPE')
g.set_xticklabels(g.get_xticklabels(), rotation=45)
plt.show()
```



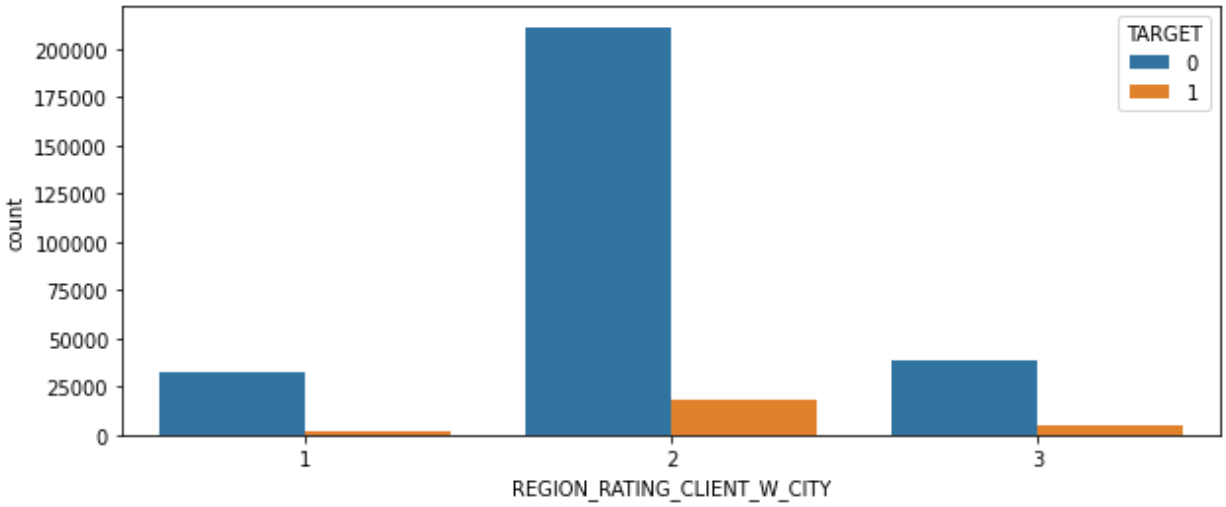
In [200…
```
plt.figure(figsize=(15,6))
sns.countplot(data=datasets["application_train"], x="NAME_INCOME_TYPE", hue="TA
plt.xlabel('NAME_INCOME_TYPE')
plt.show()
```



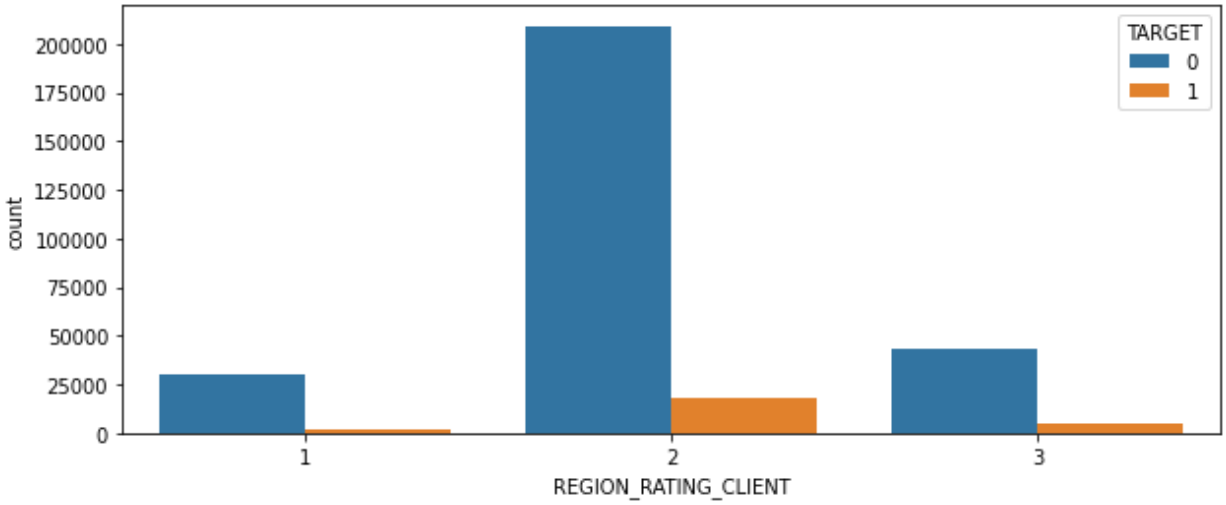
In [201…
```
plt.figure(figsize=(10,4))
i=sns.countplot(data=datasets["application_train"],x="NAME_EDUCATION_TYPE", hue
plt.xlabel('Education_level')
i.set_xticklabels(i.get_xticklabels(), rotation=45)
plt.show()
```



In [202…
```
plt.figure(figsize=(10,4))
i=sns.countplot(data=datasets["application_train"],x="REGION_RATING_CLIENT_W_CI
plt.xlabel('REGION_RATING_CLIENT_W_CITY')
plt.show()
```



In [203…
```
plt.figure(figsize=(10,4))
i=sns.countplot(data=datasets["application_train"],x="REGION_RATING_CLIENT", hu
plt.xlabel('REGION_RATING_CLIENT')
plt.show()
```



Summary of bureau

In [ ]:
```
```

In [204…
```
datasets["bureau"].info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1716428 entries, 0 to 1716427
Data columns (total 17 columns):
 #   Column                  Non-Null Count    Dtype  
---  ------                  --------------    -----  
 0   SK_ID_CURR              1716428 non-null  int64  
 1   SK_ID_BUREAU            1716428 non-null  int64  
 2   CREDIT_ACTIVE           1716428 non-null  object 
 3   CREDIT_CURRENCY         1716428 non-null  object 
 4   DAYS_CREDIT             1716428 non-null  int64  
 5   CREDIT_DAY_OVERDUE      1716428 non-null  int64  
 6   DAYS_CREDIT_ENDDATE     1610875 non-null  float64
 7   DAYS_ENDDATE_FACT       1082775 non-null  float64
 8   AMT_CREDIT_MAX_OVERDUE  591940 non-null   float64
 9   CNT_CREDIT_PROLONG      1716428 non-null  int64  
 10  AMT_CREDIT_SUM          1716415 non-null  float64
 11  AMT_CREDIT_SUM_DEBT     1458759 non-null  float64
 12  AMT_CREDIT_SUM_LIMIT    1124648 non-null  float64
 13  AMT_CREDIT_SUM_OVERDUE  1716428 non-null  float64
 14  CREDIT_TYPE             1716428 non-null  object 
 15  DAYS_CREDIT_UPDATE      1716428 non-null  int64  
 16  AMT_ANNUITY             489637 non-null   float64
dtypes: float64(8), int64(6), object(3)
memory usage: 222.6+ MB
```

In [205…
```
datasets["bureau"].shape
```

Out[205]:
```
(1716428, 17)
```

In [206…
```
datasets["bureau"].size
```

Out[206]:
```
29179276
```

In [207…
```
datasets["bureau"].head()
```

Out[207]:

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_
0	215354	5714462	Closed	currency 1	-497	
1	215354	5714463	Active	currency 1	-208	
2	215354	5714464	Active	currency 1	-203	
3	215354	5714465	Active	currency 1	-203	
4	215354	5714466	Active	currency 1	-629	

In [208…
```
datasets["bureau"].describe()
```

Out[208]:

	SK_ID_CURR	SK_ID_BUREAU	DAYS_CREDIT	CREDIT_DAY_OVERDUE	DAYS_CREDIT_EI
count	1.716428e+06	1.716428e+06	1.716428e+06	1.716428e+06	1.6108
mean	2.782149e+05	5.924434e+06	-1.142108e+03	8.181666e-01	5.105
std	1.029386e+05	5.322657e+05	7.951649e+02	3.654443e+01	4.9942
min	1.000010e+05	5.000000e+06	-2.922000e+03	0.000000e+00	-4.2060
25%	1.888668e+05	5.463954e+06	-1.666000e+03	0.000000e+00	-1.1380
50%	2.780550e+05	5.926304e+06	-9.870000e+02	0.000000e+00	-3.3000
75%	3.674260e+05	6.385681e+06	-4.740000e+02	0.000000e+00	4.7400
max	4.562550e+05	6.843457e+06	0.000000e+00	2.792000e+03	3.1199

In [209…
```
datasets["bureau"].describe(include='all')
```

Out[209]:

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	C
count	1.716428e+06	1.716428e+06	1716428	1716428	1.716428e+06	
unique	NaN	NaN	4	4	NaN	
top	NaN	NaN	Closed	currency 1	NaN	
freq	NaN	NaN	1079273	1715020	NaN	
mean	2.782149e+05	5.924434e+06	NaN	NaN	-1.142108e+03	
std	1.029386e+05	5.322657e+05	NaN	NaN	7.951649e+02	
min	1.000010e+05	5.000000e+06	NaN	NaN	-2.922000e+03	
25%	1.888668e+05	5.463954e+06	NaN	NaN	-1.666000e+03	
50%	2.780550e+05	5.926304e+06	NaN	NaN	-9.870000e+02	
75%	3.674260e+05	6.385681e+06	NaN	NaN	-4.740000e+02	
max	4.562550e+05	6.843457e+06	NaN	NaN	0.000000e+00	

In [210…
```
print("Different datatypes in Apllication_train dataset")
datasets["bureau"].dtypes.value_counts()
```

Out[210]:
```
Different datatypes in Apllication_train dataset
float64    8
int64      6
object     3
dtype: int64
```

In [211…
```
bureau_columns= datasets["bureau"].columns
print(bureau_columns)
```

```
Index(['SK_ID_CURR', 'SK_ID_BUREAU', 'CREDIT_ACTIVE', 'CREDIT_CURRENCY',
       'DAYS_CREDIT', 'CREDIT_DAY_OVERDUE', 'DAYS_CREDIT_ENDDATE',
       'DAYS_ENDDATE_FACT', 'AMT_CREDIT_MAX_OVERDUE', 'CNT_CREDIT_PROLONG',
       'AMT_CREDIT_SUM', 'AMT_CREDIT_SUM_DEBT', 'AMT_CREDIT_SUM_LIMIT',
       'AMT_CREDIT_SUM_OVERDUE', 'CREDIT_TYPE', 'DAYS_CREDIT_UPDATE',
       'AMT_ANNUITY'],
      dtype='object')
```

In [212…
```
for key in datasets["bureau"].keys():
    n=len(pd.unique(datasets["bureau"][key]))
    print('Nunber of Unique values in ', format(key), 'is', format(n))
```

```
Nunber of Unique values in  SK_ID_CURR is 305811
Nunber of Unique values in  SK_ID_BUREAU is 1716428
Nunber of Unique values in  CREDIT_ACTIVE is 4
Nunber of Unique values in  CREDIT_CURRENCY is 4
Nunber of Unique values in  DAYS_CREDIT is 2923
Nunber of Unique values in  CREDIT_DAY_OVERDUE is 942
Nunber of Unique values in  DAYS_CREDIT_ENDDATE is 14097
Nunber of Unique values in  DAYS_ENDDATE_FACT is 2918
Nunber of Unique values in  AMT_CREDIT_MAX_OVERDUE is 68252
Nunber of Unique values in  CNT_CREDIT_PROLONG is 10
Nunber of Unique values in  AMT_CREDIT_SUM is 236709
Nunber of Unique values in  AMT_CREDIT_SUM_DEBT is 226538
Nunber of Unique values in  AMT_CREDIT_SUM_LIMIT is 51727
Nunber of Unique values in  AMT_CREDIT_SUM_OVERDUE is 1616
Nunber of Unique values in  CREDIT_TYPE is 15
Nunber of Unique values in  DAYS_CREDIT_UPDATE is 2982
Nunber of Unique values in  AMT_ANNUITY is 40322
```

In [213…
```
datasets["bureau"].isnull().head()
```

Out[213]:

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_
0	False	False	False	False	False	
1	False	False	False	False	False	
2	False	False	False	False	False	
3	False	False	False	False	False	
4	False	False	False	False	False	

In [214…
```
s = datasets["bureau"].isnull().sum()
```

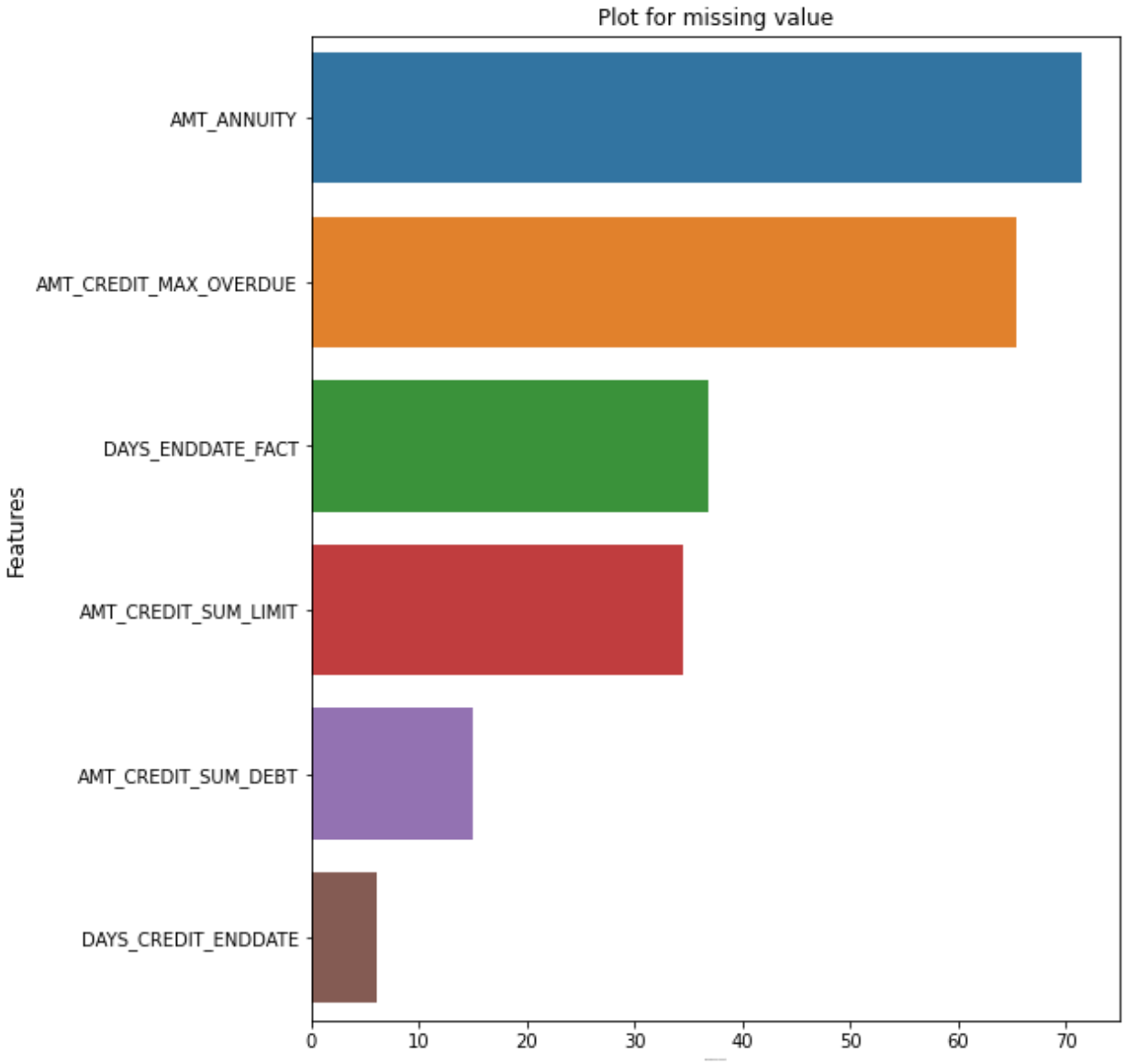
In [215…
```
s[s!=0]
```

Out[215]:
```
DAYS_CREDIT_ENDDATE        105553
DAYS_ENDDATE_FACT          633653
AMT_CREDIT_MAX_OVERDUE    1124488
AMT_CREDIT_SUM                 13
AMT_CREDIT_SUM_DEBT        257669
AMT_CREDIT_SUM_LIMIT       591780
AMT_ANNUITY               1226791
dtype: int64
```

In [216…
```
missing_value_info_plot(datasets['bureau'], 'bureau')
```

Out[216]:

	Percent	Train Missing Count
AMT_ANNUITY	71.47	1226791
AMT_CREDIT_MAX_OVERDUE	65.51	1124488
DAYS_ENDDATE_FACT	36.92	633653
AMT_CREDIT_SUM_LIMIT	34.48	591780
AMT_CREDIT_SUM_DEBT	15.01	257669
DAYS_CREDIT_ENDDATE	6.15	105553



Summary of bureau_balance

In [217…

```
datasets["bureau_balance"].info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27299925 entries, 0 to 27299924
Data columns (total 3 columns):
 #   Column          Non-Null Count     Dtype
---  ------          --------------     -----
 0   SK_ID_BUREAU    27299925 non-null  int64
 1   MONTHS_BALANCE  27299925 non-null  int64
 2   STATUS          27299925 non-null  object
dtypes: int64(2), object(1)
memory usage: 624.8+ MB
```

In [218…
```
datasets["bureau_balance"].shape
```

Out[218]:
```
(27299925, 3)
```

In [219…
```
datasets["bureau_balance"].head()
```

Out[219]:

	SK_ID_BUREAU	MONTHS_BALANCE	STATUS
0	5715448	0	C
1	5715448	-1	C
2	5715448	-2	C
3	5715448	-3	C
4	5715448	-4	C

In [220…
```
datasets["bureau_balance"].describe()
```

Out[220]:

	SK_ID_BUREAU	MONTHS_BALANCE
count	2.729992e+07	2.729992e+07
mean	6.036297e+06	-3.074169e+01
std	4.923489e+05	2.386451e+01
min	5.001709e+06	-9.600000e+01
25%	5.730933e+06	-4.600000e+01
50%	6.070821e+06	-2.500000e+01
75%	6.431951e+06	-1.100000e+01
max	6.842888e+06	0.000000e+00

In [221…
```
print("Different datatypes in bureau_balance dataset")
datasets["bureau_balance"].dtypes.value_counts()
```

Out[221]:
```
Different datatypes in bureau_balance dataset
int64     2
object    1
dtype: int64
```

In [222…
```
bureau_balance_columns= datasets["bureau_balance"].columns
print(bureau_balance_columns)
```

```
Index(['SK_ID_BUREAU', 'MONTHS_BALANCE', 'STATUS'], dtype='object')
```

In [223…
```
for key in datasets["bureau_balance"].keys():
    n=len(pd.unique(datasets["bureau_balance"][key]))
    print('Nunber of Unique values in ', format(key), 'is', format(n))
```

```
Nunber of Unique values in  SK_ID_BUREAU is 817395
Nunber of Unique values in  MONTHS_BALANCE is 97
Nunber of Unique values in  STATUS is 8
```

In [224…
```
s = datasets["bureau_balance"].isnull().sum()
```

In [225…
```
s[s!=0]
```

Out[225]:
```
Series([], dtype: int64)
```

Missing data for bureau_balance

In [226…
```
missing_value_info_plot(datasets['bureau_balance'], 'bureau_balance')
```

Out[226]:

	Percent	Train Missing Count

Observation: As we can observe, bureaua_balance has no missing data.

Summary of Previous_application

In [227…
```
datasets["previous_application"].info(verbose=True, null_counts=True)
```

```
<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  
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 16  NAME_CONTRACT_STATUS         1670214 non-null  object 
 17  DAYS_DECISION                1670214 non-null  int64  
 18  NAME_PAYMENT_TYPE            1670214 non-null  object 
 19  CODE_REJECT_REASON           1670214 non-null  object 
 20  NAME_TYPE_SUITE              849809 non-null   object 
 21  NAME_CLIENT_TYPE             1670214 non-null  object 
 22  NAME_GOODS_CATEGORY          1670214 non-null  object 
 23  NAME_PORTFOLIO               1670214 non-null  object 
 24  NAME_PRODUCT_TYPE            1670214 non-null  object 
 25  CHANNEL_TYPE                 1670214 non-null  object 
 26  SELLERPLACE_AREA             1670214 non-null  int64  
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object 
 30  PRODUCT_COMBINATION          1669868 non-null  object 
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

In [228…
```
datasets["previous_application"].shape
```

Out[228]:
```
(1670214, 37)
```

In [229…
```
datasets["previous_application"].head()
```

Out[229]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AM
0	2030495	271877	Consumer loans	1730.430	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	
2	2523466	122040	Cash loans	15060.735	112500.0	
3	2819243	176158	Cash loans	47041.335	450000.0	
4	1784265	202054	Cash loans	31924.395	337500.0	

5 rows × 37 columns

In [230…
```
datasets["previous_application"].describe()
```

Out[230]:

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_I
count	1.670214e+06	1.670214e+06	1.297979e+06	1.670214e+06	1.670213e+06	
mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05	1.961140e+05	
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05	3.185746e+05	
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	0.000000e+00	
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04	2.416050e+04	
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	8.054100e+04	
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05	2.164185e+05	
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	6.905160e+06	

8 rows × 21 columns

In [231…
```
print("Different datatypes in previous_application dataset")
datasets["previous_application"].dtypes.value_counts()
```

Out[231]:
```
Different datatypes in previous_application dataset
object     16
float64    15
int64       6
dtype: int64
```

In [232…
```
prev_columns= datasets["previous_application"].columns
print(prev_columns)
```

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY',
       'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_DOWN_PAYMENT', 'AMT_GOODS_PRICE',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY',
       'RATE_DOWN_PAYMENT', 'RATE_INTEREST_PRIMARY',
       'RATE_INTEREST_PRIVILEGED', 'NAME_CASH_LOAN_PURPOSE',
       'NAME_CONTRACT_STATUS', 'DAYS_DECISION', 'NAME_PAYMENT_TYPE',
       'CODE_REJECT_REASON', 'NAME_TYPE_SUITE', 'NAME_CLIENT_TYPE',
       'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE',
       'CHANNEL_TYPE', 'SELLERPLACE_AREA', 'NAME_SELLER_INDUSTRY',
       'CNT_PAYMENT', 'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION',
       'DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION',
       'DAYS_LAST_DUE', 'DAYS_TERMINATION', 'NFLAG_INSURED_ON_APPROVAL'],
      dtype='object')
```

In [233…

```
for key in datasets["previous_application"].keys():
    n=len(pd.unique(datasets["previous_application"][key]))
    print('Nunber of Unique values in ', format(key), 'is', format(n))
```

```
Nunber of Unique values in  SK_ID_PREV is 1670214
Nunber of Unique values in  SK_ID_CURR is 338857
Nunber of Unique values in  NAME_CONTRACT_TYPE is 4
Nunber of Unique values in  AMT_ANNUITY is 357960
Nunber of Unique values in  AMT_APPLICATION is 93885
Nunber of Unique values in  AMT_CREDIT is 86804
Nunber of Unique values in  AMT_DOWN_PAYMENT is 29279
Nunber of Unique values in  AMT_GOODS_PRICE is 93886
Nunber of Unique values in  WEEKDAY_APPR_PROCESS_START is 7
Nunber of Unique values in  HOUR_APPR_PROCESS_START is 24
Nunber of Unique values in  FLAG_LAST_APPL_PER_CONTRACT is 2
Nunber of Unique values in  NFLAG_LAST_APPL_IN_DAY is 2
Nunber of Unique values in  RATE_DOWN_PAYMENT is 207034
Nunber of Unique values in  RATE_INTEREST_PRIMARY is 149
Nunber of Unique values in  RATE_INTEREST_PRIVILEGED is 26
Nunber of Unique values in  NAME_CASH_LOAN_PURPOSE is 25
Nunber of Unique values in  NAME_CONTRACT_STATUS is 4
Nunber of Unique values in  DAYS_DECISION is 2922
Nunber of Unique values in  NAME_PAYMENT_TYPE is 4
Nunber of Unique values in  CODE_REJECT_REASON is 9
Nunber of Unique values in  NAME_TYPE_SUITE is 8
Nunber of Unique values in  NAME_CLIENT_TYPE is 4
Nunber of Unique values in  NAME_GOODS_CATEGORY is 28
Nunber of Unique values in  NAME_PORTFOLIO is 5
Nunber of Unique values in  NAME_PRODUCT_TYPE is 3
Nunber of Unique values in  CHANNEL_TYPE is 8
Nunber of Unique values in  SELLERPLACE_AREA is 2097
Nunber of Unique values in  NAME_SELLER_INDUSTRY is 11
Nunber of Unique values in  CNT_PAYMENT is 50
Nunber of Unique values in  NAME_YIELD_GROUP is 5
Nunber of Unique values in  PRODUCT_COMBINATION is 18
Nunber of Unique values in  DAYS_FIRST_DRAWING is 2839
Nunber of Unique values in  DAYS_FIRST_DUE is 2893
Nunber of Unique values in  DAYS_LAST_DUE_1ST_VERSION is 4606
Nunber of Unique values in  DAYS_LAST_DUE is 2874
Nunber of Unique values in  DAYS_TERMINATION is 2831
Nunber of Unique values in  NFLAG_INSURED_ON_APPROVAL is 3
```

In [234…

```
s = datasets["previous_application"].isnull().sum()
```

In [235…
```
s[s!=0]
```

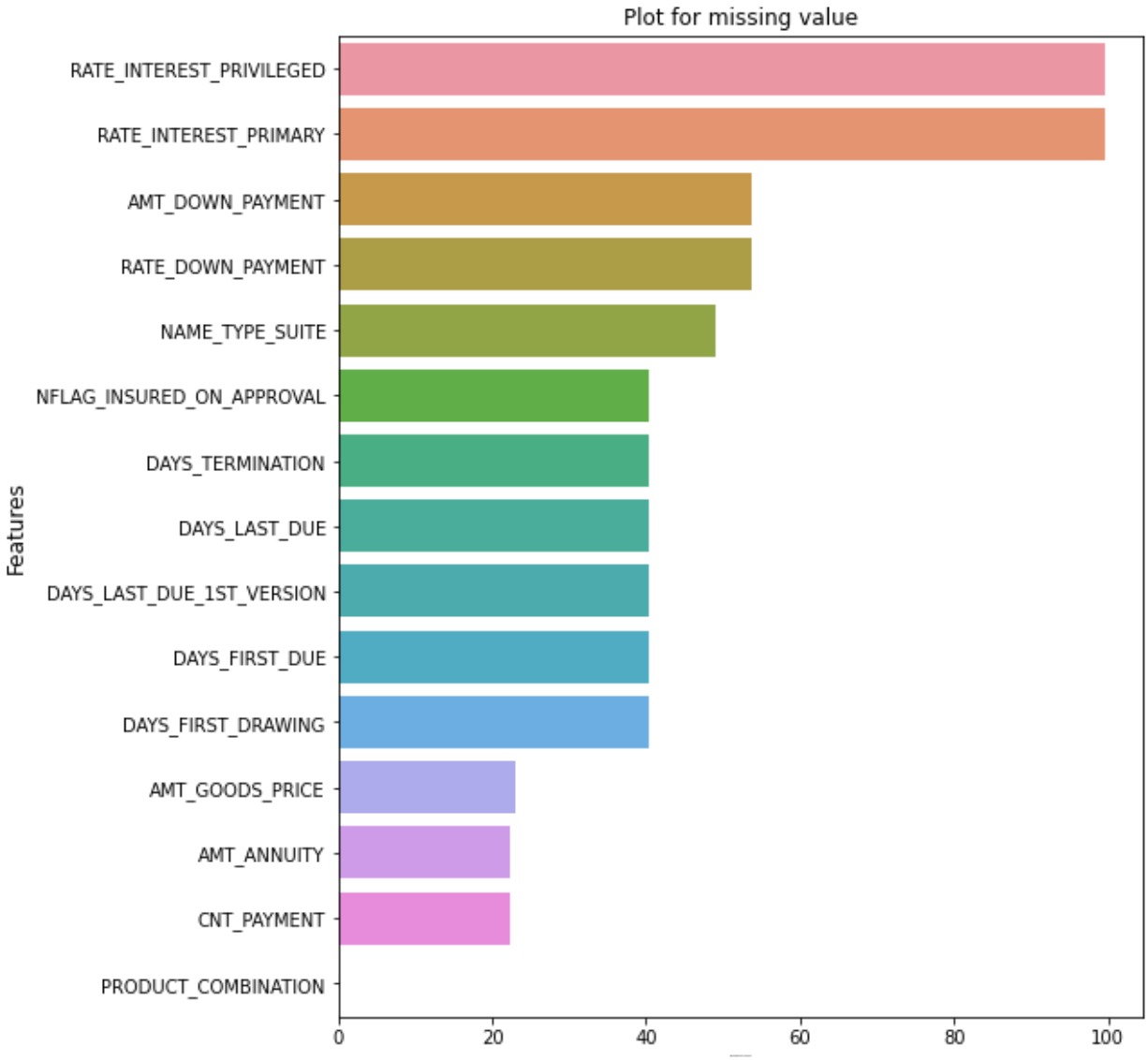
Out[235]:
```
AMT_ANNUITY                 372235
AMT_CREDIT                       1
AMT_DOWN_PAYMENT            895844
AMT_GOODS_PRICE             385515
RATE_DOWN_PAYMENT           895844
RATE_INTEREST_PRIMARY      1664263
RATE_INTEREST_PRIVILEGED   1664263
NAME_TYPE_SUITE             820405
CNT_PAYMENT                 372230
PRODUCT_COMBINATION            346
DAYS_FIRST_DRAWING          673065
DAYS_FIRST_DUE              673065
DAYS_LAST_DUE_1ST_VERSION   673065
DAYS_LAST_DUE               673065
DAYS_TERMINATION            673065
NFLAG_INSURED_ON_APPROVAL   673065
dtype: int64
```

Missing data for application train

In [236…
```
missing_value_info_plot(datasets['previous_application'], 'previous_applicatio
```

Out[236]:

	Percent	Train Missing Count
RATE_INTEREST_PRIVILEGED	99.64	1664263
RATE_INTEREST_PRIMARY	99.64	1664263
AMT_DOWN_PAYMENT	53.64	895844
RATE_DOWN_PAYMENT	53.64	895844
NAME_TYPE_SUITE	49.12	820405
NFLAG_INSURED_ON_APPROVAL	40.30	673065
DAYS_TERMINATION	40.30	673065
DAYS_LAST_DUE	40.30	673065
DAYS_LAST_DUE_1ST_VERSION	40.30	673065
DAYS_FIRST_DUE	40.30	673065
DAYS_FIRST_DRAWING	40.30	673065
AMT_GOODS_PRICE	23.08	385515
AMT_ANNUITY	22.29	372235
CNT_PAYMENT	22.29	372230
PRODUCT_COMBINATION	0.02	346



Summary of POS_CASH_BALANCE

In [237…
```
datasets["POS_CASH_balance"].info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10001358 entries, 0 to 10001357
Data columns (total 8 columns):
 #   Column                 Non-Null Count     Dtype  
---  ------                 --------------     -----  
 0   SK_ID_PREV             10001358 non-null  int64  
 1   SK_ID_CURR             10001358 non-null  int64  
 2   MONTHS_BALANCE         10001358 non-null  int64  
 3   CNT_INSTALMENT         9975287 non-null   float64
 4   CNT_INSTALMENT_FUTURE  9975271 non-null   float64
 5   NAME_CONTRACT_STATUS   10001358 non-null  object 
 6   SK_DPD                 10001358 non-null  int64  
 7   SK_DPD_DEF             10001358 non-null  int64  
dtypes: float64(2), int64(5), object(1)
memory usage: 610.4+ MB
```

In [238…
```
datasets["POS_CASH_balance"].shape
```

Out[238]: (10001358, 8)

In [239…
```
datasets["POS_CASH_balance"].head()
```

Out[239]:

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	CNT_INSTALMENT	CNT_INSTALMENT_FUTU
0	1803195	182943	-31	48.0	4
1	1715348	367990	-33	36.0	3
2	1784872	397406	-32	12.0	
3	1903291	269225	-35	48.0	4
4	2341044	334279	-35	36.0	3

In [240…
```
datasets["POS_CASH_balance"].describe()
```

Out[240]:

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	CNT_INSTALMENT	CNT_INSTALMENT
count	1.000136e+07	1.000136e+07	1.000136e+07	9.975287e+06	9.97
mean	1.903217e+06	2.784039e+05	-3.501259e+01	1.708965e+01	1.04
std	5.358465e+05	1.027637e+05	2.606657e+01	1.199506e+01	1.11
min	1.000001e+06	1.000010e+05	-9.600000e+01	1.000000e+00	0.00
25%	1.434405e+06	1.895500e+05	-5.400000e+01	1.000000e+01	3.00
50%	1.896565e+06	2.786540e+05	-2.800000e+01	1.200000e+01	7.00
75%	2.368963e+06	3.674290e+05	-1.300000e+01	2.400000e+01	1.40
max	2.843499e+06	4.562550e+05	-1.000000e+00	9.200000e+01	8.50

In [241…
```
print("Different datatypes in POS_CASH_balance dataset")
datasets["POS_CASH_balance"].dtypes.value_counts()
```

Out[241]:
```
Different datatypes in POS_CASH_balance dataset
int64      5
float64    2
object     1
dtype: int64
```

In [242…
```
pos_columns= datasets["POS_CASH_balance"].columns
print(pos_columns)
```

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'MONTHS_BALANCE', 'CNT_INSTALMENT',
       'CNT_INSTALMENT_FUTURE', 'NAME_CONTRACT_STATUS', 'SK_DPD',
       'SK_DPD_DEF'],
      dtype='object')
```

In [243…
```
for key in datasets["POS_CASH_balance"].keys():
    n=len(pd.unique(datasets["POS_CASH_balance"][key]))
    print('Nunber of Unique values in ', format(key), 'is', format(n))
```

```
Nunber of Unique values in  SK_ID_PREV is 936325
Nunber of Unique values in  SK_ID_CURR is 337252
Nunber of Unique values in  MONTHS_BALANCE is 96
Nunber of Unique values in  CNT_INSTALMENT is 74
Nunber of Unique values in  CNT_INSTALMENT_FUTURE is 80
Nunber of Unique values in  NAME_CONTRACT_STATUS is 9
Nunber of Unique values in  SK_DPD is 3400
Nunber of Unique values in  SK_DPD_DEF is 2307
```

In [244…
```
s = datasets["POS_CASH_balance"].isnull().sum()
```

In [245…
```
s[s!=0]
```

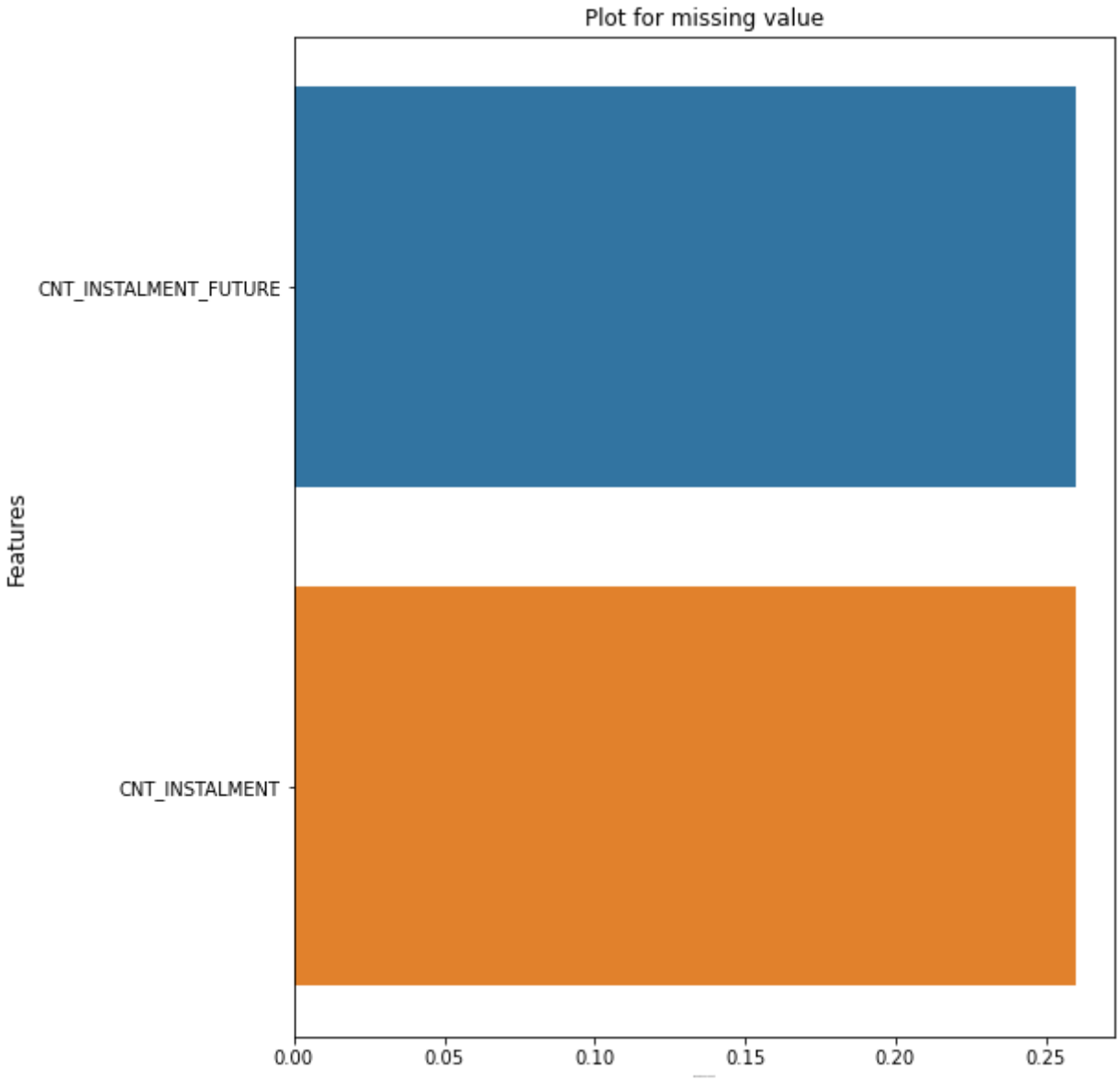
Out[245]:
```
CNT_INSTALMENT           26071
CNT_INSTALMENT_FUTURE    26087
dtype: int64
```

Missing data for POS_CASH_balance

In [246…
```
missing_value_info_plot(datasets['POS_CASH_balance'], 'POS_CASH_balance')
```

Out[246]:

	Percent	Train Missing Count
CNT_INSTALMENT_FUTURE	0.26	26087
CNT_INSTALMENT	0.26	26071



Summary of credit_card_balance

In [247…

```
datasets["credit_card_balance"].info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3840312 entries, 0 to 3840311
Data columns (total 23 columns):
 #   Column                      Non-Null Count    Dtype  
---  ------                      --------------    -----  
 0   SK_ID_PREV                  3840312 non-null  int64  
 1   SK_ID_CURR                  3840312 non-null  int64  
 2   MONTHS_BALANCE              3840312 non-null  int64  
 3   AMT_BALANCE                 3840312 non-null  float64
 4   AMT_CREDIT_LIMIT_ACTUAL     3840312 non-null  int64  
 5   AMT_DRAWINGS_ATM_CURRENT    3090496 non-null  float64
 6   AMT_DRAWINGS_CURRENT        3840312 non-null  float64
 7   AMT_DRAWINGS_OTHER_CURRENT  3090496 non-null  float64
 8   AMT_DRAWINGS_POS_CURRENT    3090496 non-null  float64
 9   AMT_INST_MIN_REGULARITY     3535076 non-null  float64
 10  AMT_PAYMENT_CURRENT         3072324 non-null  float64
 11  AMT_PAYMENT_TOTAL_CURRENT   3840312 non-null  float64
 12  AMT_RECEIVABLE_PRINCIPAL    3840312 non-null  float64
 13  AMT_RECIVABLE               3840312 non-null  float64
 14  AMT_TOTAL_RECEIVABLE        3840312 non-null  float64
 15  CNT_DRAWINGS_ATM_CURRENT    3090496 non-null  float64
 16  CNT_DRAWINGS_CURRENT        3840312 non-null  int64  
 17  CNT_DRAWINGS_OTHER_CURRENT  3090496 non-null  float64
 18  CNT_DRAWINGS_POS_CURRENT    3090496 non-null  float64
 19  CNT_INSTALMENT_MATURE_CUM   3535076 non-null  float64
 20  NAME_CONTRACT_STATUS        3840312 non-null  object 
 21  SK_DPD                      3840312 non-null  int64  
 22  SK_DPD_DEF                  3840312 non-null  int64  
dtypes: float64(15), int64(7), object(1)
memory usage: 673.9+ MB
```

In [248…
```
datasets["credit_card_balance"].shape
```

Out[248]:
```
(3840312, 23)
```

In [249…
```
datasets["credit_card_balance"].head()
```

Out[249]:

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL
0	2562384	378907	-6	56.970	135000
1	2582071	363914	-1	63975.555	45000
2	1740877	371185	-7	31815.225	450000
3	1389973	337855	-4	236572.110	225000
4	1891521	126868	-1	453919.455	450000

5 rows × 23 columns

In [250…
```
datasets["credit_card_balance"].describe()
```

Out[250]:

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_
count	3.840312e+06	3.840312e+06	3.840312e+06	3.840312e+06	3.840
mean	1.904504e+06	2.783242e+05	-3.452192e+01	5.830016e+04	1.538
std	5.364695e+05	1.027045e+05	2.666775e+01	1.063070e+05	1.651
min	1.000018e+06	1.000060e+05	-9.600000e+01	-4.202502e+05	0.000
25%	1.434385e+06	1.895170e+05	-5.500000e+01	0.000000e+00	4.500
50%	1.897122e+06	2.783960e+05	-2.800000e+01	0.000000e+00	1.125
75%	2.369328e+06	3.675800e+05	-1.100000e+01	8.904669e+04	1.800
max	2.843496e+06	4.562500e+05	-1.000000e+00	1.505902e+06	1.350

8 rows × 22 columns

In [251…
```
print("Different datatypes in credit_card_balance dataset")
datasets["credit_card_balance"].dtypes.value_counts()
```

Out[251]:
```
Different datatypes in credit_card_balance dataset
float64    15
int64       7
object      1
dtype: int64
```

In [252…
```
credit_columns= datasets["credit_card_balance"].columns
print(credit_columns)
```

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'MONTHS_BALANCE', 'AMT_BALANCE',
       'AMT_CREDIT_LIMIT_ACTUAL', 'AMT_DRAWINGS_ATM_CURRENT',
       'AMT_DRAWINGS_CURRENT', 'AMT_DRAWINGS_OTHER_CURRENT',
       'AMT_DRAWINGS_POS_CURRENT', 'AMT_INST_MIN_REGULARITY',
       'AMT_PAYMENT_CURRENT', 'AMT_PAYMENT_TOTAL_CURRENT',
       'AMT_RECEIVABLE_PRINCIPAL', 'AMT_RECIVABLE', 'AMT_TOTAL_RECEIVABLE',
       'CNT_DRAWINGS_ATM_CURRENT', 'CNT_DRAWINGS_CURRENT',
       'CNT_DRAWINGS_OTHER_CURRENT', 'CNT_DRAWINGS_POS_CURRENT',
       'CNT_INSTALMENT_MATURE_CUM', 'NAME_CONTRACT_STATUS', 'SK_DPD',
       'SK_DPD_DEF'],
      dtype='object')
```

In [253…
```
for key in datasets["credit_card_balance"].keys():
    n=len(pd.unique(datasets["credit_card_balance"][key]))
    print('Nunber of Unique values in ', format(key), 'is', format(n))
```

```
Nunber of Unique values in  SK_ID_PREV is 104307
Nunber of Unique values in  SK_ID_CURR is 103558
Nunber of Unique values in  MONTHS_BALANCE is 96
Nunber of Unique values in  AMT_BALANCE is 1347904
Nunber of Unique values in  AMT_CREDIT_LIMIT_ACTUAL is 181
Nunber of Unique values in  AMT_DRAWINGS_ATM_CURRENT is 2268
Nunber of Unique values in  AMT_DRAWINGS_CURRENT is 187005
Nunber of Unique values in  AMT_DRAWINGS_OTHER_CURRENT is 1833
Nunber of Unique values in  AMT_DRAWINGS_POS_CURRENT is 168749
Nunber of Unique values in  AMT_INST_MIN_REGULARITY is 312267
Nunber of Unique values in  AMT_PAYMENT_CURRENT is 163210
Nunber of Unique values in  AMT_PAYMENT_TOTAL_CURRENT is 182957
Nunber of Unique values in  AMT_RECEIVABLE_PRINCIPAL is 1195839
Nunber of Unique values in  AMT_RECIVABLE is 1338878
Nunber of Unique values in  AMT_TOTAL_RECEIVABLE is 1339008
Nunber of Unique values in  CNT_DRAWINGS_ATM_CURRENT is 45
Nunber of Unique values in  CNT_DRAWINGS_CURRENT is 129
Nunber of Unique values in  CNT_DRAWINGS_OTHER_CURRENT is 12
Nunber of Unique values in  CNT_DRAWINGS_POS_CURRENT is 134
Nunber of Unique values in  CNT_INSTALMENT_MATURE_CUM is 122
Nunber of Unique values in  NAME_CONTRACT_STATUS is 7
Nunber of Unique values in  SK_DPD is 917
Nunber of Unique values in  SK_DPD_DEF is 378
```

In [254…
```
s = datasets["credit_card_balance"].isnull().sum()
```

In [255…
```
s[s!=0]
```

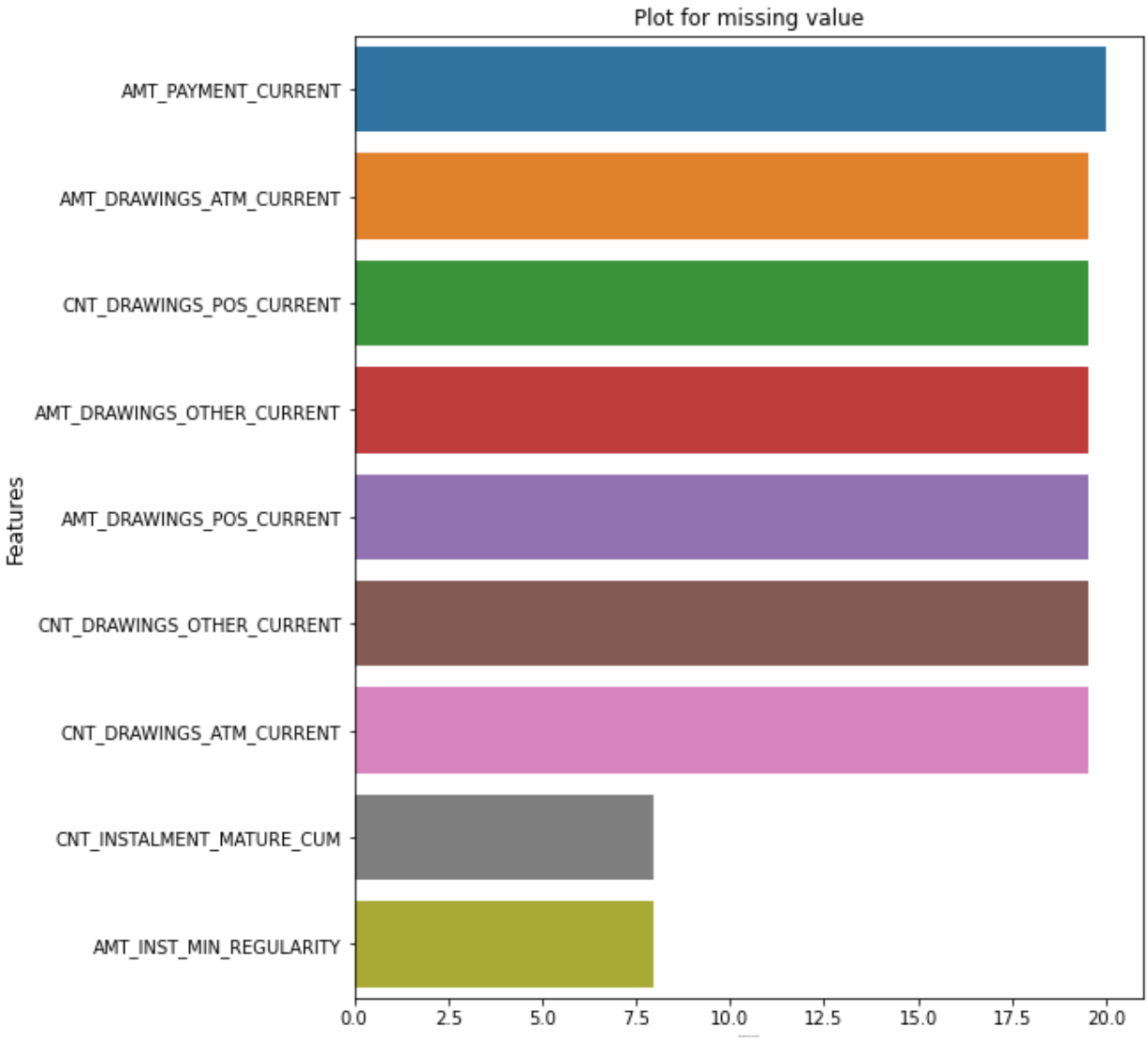
Out[255]:
```
AMT_DRAWINGS_ATM_CURRENT      749816
AMT_DRAWINGS_OTHER_CURRENT    749816
AMT_DRAWINGS_POS_CURRENT      749816
AMT_INST_MIN_REGULARITY       305236
AMT_PAYMENT_CURRENT           767988
CNT_DRAWINGS_ATM_CURRENT      749816
CNT_DRAWINGS_OTHER_CURRENT    749816
CNT_DRAWINGS_POS_CURRENT      749816
CNT_INSTALMENT_MATURE_CUM     305236
dtype: int64
```

Missing data for credit_card_balance

In [256…
```
missing_value_info_plot(datasets['credit_card_balance'], 'credit_card_balance')
```

Out[256]:

	Percent	Train Missing Count
AMT_PAYMENT_CURRENT	20.00	767988
AMT_DRAWINGS_ATM_CURRENT	19.52	749816
CNT_DRAWINGS_POS_CURRENT	19.52	749816
AMT_DRAWINGS_OTHER_CURRENT	19.52	749816
AMT_DRAWINGS_POS_CURRENT	19.52	749816
CNT_DRAWINGS_OTHER_CURRENT	19.52	749816
CNT_DRAWINGS_ATM_CURRENT	19.52	749816
CNT_INSTALMENT_MATURE_CUM	7.95	305236
AMT_INST_MIN_REGULARITY	7.95	305236



Summary of installments_payments

In [257…

```
datasets["installments_payments"].info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13605401 entries, 0 to 13605400
Data columns (total 8 columns):
 #   Column                  Non-Null Count     Dtype  
---  ------                  --------------     -----  
 0   SK_ID_PREV              13605401 non-null  int64  
 1   SK_ID_CURR              13605401 non-null  int64  
 2   NUM_INSTALMENT_VERSION  13605401 non-null  float64
 3   NUM_INSTALMENT_NUMBER   13605401 non-null  int64  
 4   DAYS_INSTALMENT         13605401 non-null  float64
 5   DAYS_ENTRY_PAYMENT      13602496 non-null  float64
 6   AMT_INSTALMENT          13605401 non-null  float64
 7   AMT_PAYMENT             13602496 non-null  float64
dtypes: float64(5), int64(3)
memory usage: 830.4 MB
```

In [258…
```
datasets["installments_payments"].shape
```

Out[258]:
```
(13605401, 8)
```

In [259…
```
datasets["installments_payments"].head()
```

Out[259]:

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYS
0	1054186	161674	1.0	6	
1	1330831	151639	0.0	34	
2	2085231	193053	2.0	1	
3	2452527	199697	1.0	3	
4	2714724	167756	1.0	2	

In [260…
```
datasets["installments_payments"].describe()
```

Out[260]:

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER
count	1.360540e+07	1.360540e+07	1.360540e+07	1.360540e+07
mean	1.903365e+06	2.784449e+05	8.566373e-01	1.887090e+01
std	5.362029e+05	1.027183e+05	1.035216e+00	2.666407e+01
min	1.000001e+06	1.000010e+05	0.000000e+00	1.000000e+00
25%	1.434191e+06	1.896390e+05	0.000000e+00	4.000000e+00
50%	1.896520e+06	2.786850e+05	1.000000e+00	8.000000e+00
75%	2.369094e+06	3.675300e+05	1.000000e+00	1.900000e+01
max	2.843499e+06	4.562550e+05	1.780000e+02	2.770000e+02

In [261…
```
print("Different datatypes in installments_payments dataset")
datasets["installments_payments"].dtypes.value_counts()
```

Out[261]:
```
Different datatypes in installments_payments dataset
float64    5
int64      3
dtype: int64
```

In [262…
```
inst_columns= datasets["installments_payments"].columns
print(inst_columns)
```

```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NUM_INSTALMENT_VERSION',
       'NUM_INSTALMENT_NUMBER', 'DAYS_INSTALMENT', 'DAYS_ENTRY_PAYMENT',
       'AMT_INSTALMENT', 'AMT_PAYMENT'],
      dtype='object')
```

In [263…
```
for key in datasets["installments_payments"].keys():
    n=len(pd.unique(datasets["installments_payments"][key]))
    print('Nunber of Unique values in ', format(key), 'is', format(n))
```

```
Nunber of Unique values in  SK_ID_PREV is 997752
Nunber of Unique values in  SK_ID_CURR is 339587
Nunber of Unique values in  NUM_INSTALMENT_VERSION is 65
Nunber of Unique values in  NUM_INSTALMENT_NUMBER is 277
Nunber of Unique values in  DAYS_INSTALMENT is 2922
Nunber of Unique values in  DAYS_ENTRY_PAYMENT is 3040
Nunber of Unique values in  AMT_INSTALMENT is 902539
Nunber of Unique values in  AMT_PAYMENT is 944236
```

In [264…
```
s = datasets["installments_payments"].isnull().sum()
```

In [265…
```
s[s!=0]
```

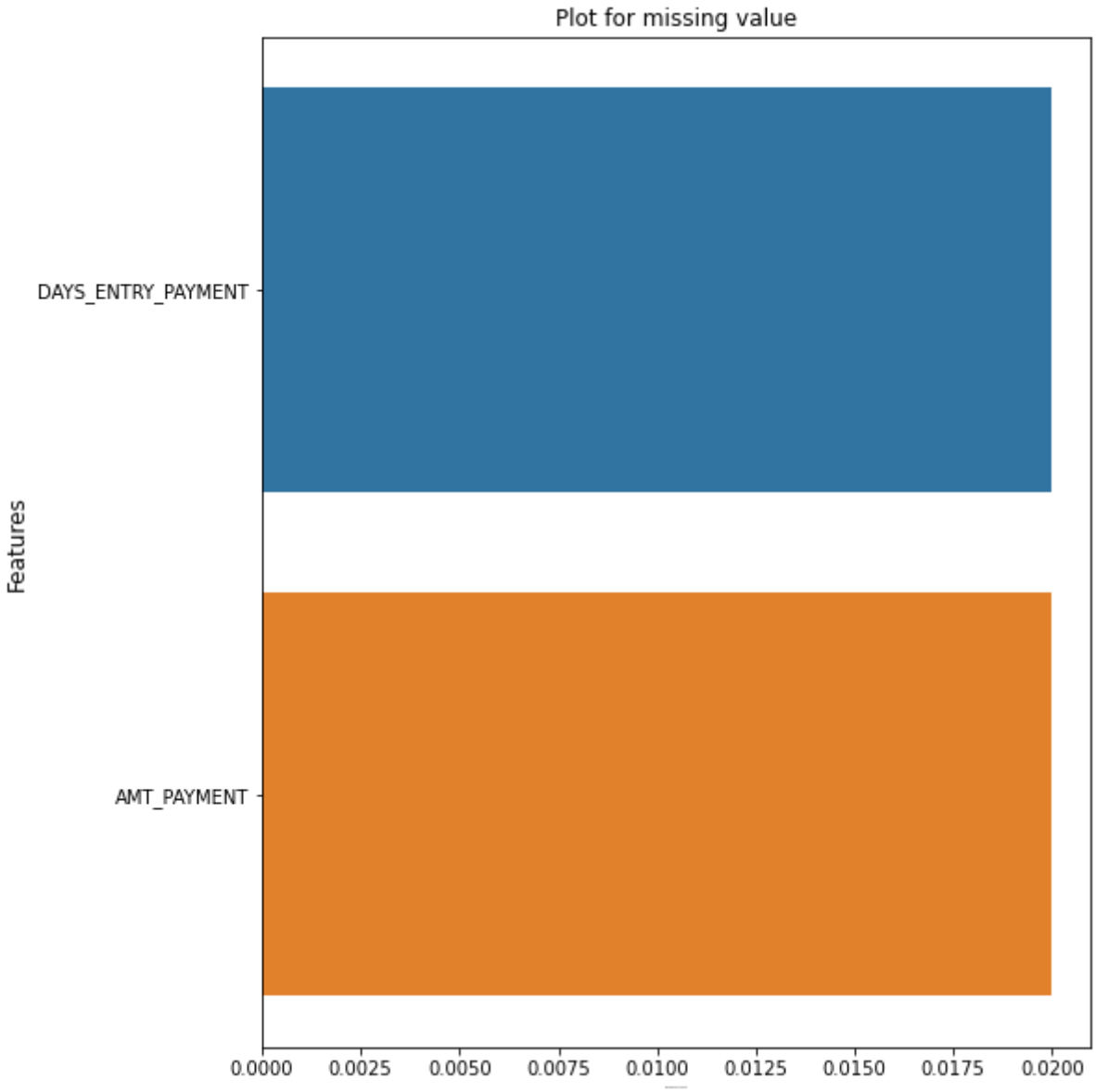
Out[265]:
```
DAYS_ENTRY_PAYMENT    2905
AMT_PAYMENT           2905
dtype: int64
```

Missing data for installments_payments

In [266…
```
missing_value_info_plot(datasets['installments_payments'], 'installments_paymer
```

Out[266]:

	Percent	Train Missing Count
DAYS_ENTRY_PAYMENT	0.02	2905
AMT_PAYMENT	0.02	2905



In [ ]:

Dataset questions

Unique record for each SK_ID_CURR

In [ ]:

In [267…
```
datasets.keys()
```

Out[267]:
```
dict_keys(['application_train', 'application_test', 'bureau', 'bureau_balanc
e', 'credit_card_balance', 'installments_payments', 'previous_application', 'P
OS_CASH_balance'])
```

In [268…
```
len(datasets["application_train"]["SK_ID_CURR"].unique()) == datasets["applicat
```

Out[268]:
```
True
```

In [269…
```
np.intersect1d(datasets["application_train"]["SK_ID_CURR"], datasets["applicati
```
Out[269]:
```
array([], dtype=int64)
```

In [270…
```
datasets["application_test"].shape
```
Out[270]:
```
(48744, 121)
```

In [271…
```
datasets["application_train"].shape
```
Out[271]:
```
(307511, 122)
```

previous applications for the submission file

The persons in the kaggle submission file have had previous applications in the `previous_application.csv`. 47,800 out 48,744 people have had previous appications.

In [272…
```
#appsDF.shape()
```

In [273…
```
appsDF = datasets["previous_application"]
```

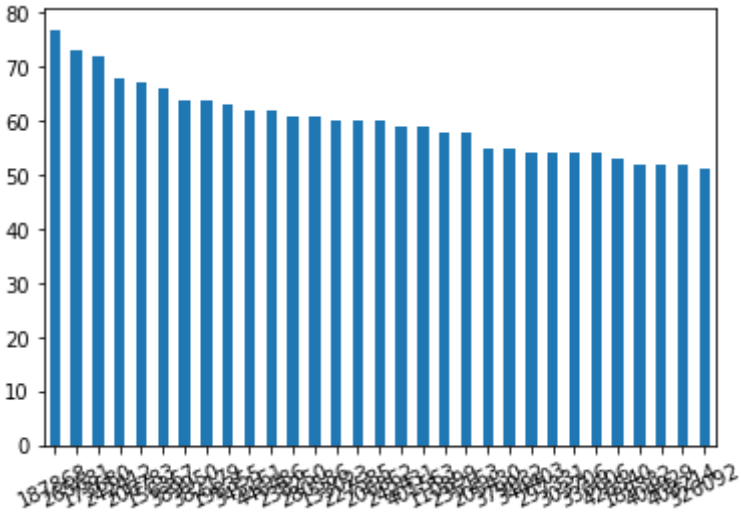
In [274…
```
len(np.intersect1d(datasets["previous_application"]["SK_ID_CURR"], datasets["ap
```
Out[274]:
```
47800
```

In [275…
```
print(f"There are  {appsDF.shape[0]:,} previous applications")
```
```
There are  1,670,214 previous applications
```

In [276…
```
# How many entries are there for each month?
prevAppCounts = appsDF['SK_ID_CURR'].value_counts(dropna=False)
```

In [277…
```
len(prevAppCounts[prevAppCounts >40])  #more that 40 previous applications
```
Out[277]:
```
101
```

In [278…
```
prevAppCounts[prevAppCounts >50].plot(kind='bar')
plt.xticks(rotation=25)
plt.show()
```



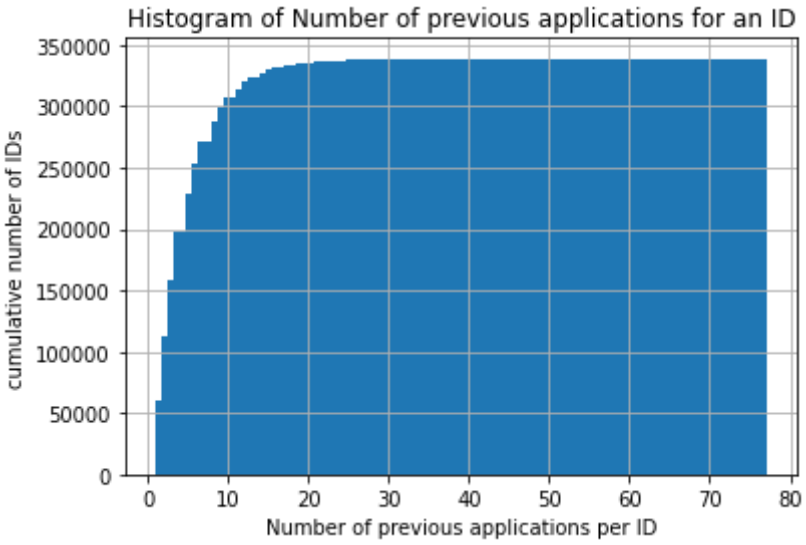
Histogram of Number of previous applications for an ID

In [279…
```
sum(appsDF['SK_ID_CURR'].value_counts()==1)
```

Out[279]:
```
60458
```

In [280…
```
plt.hist(appsDF['SK_ID_CURR'].value_counts(), cumulative =True, bins = 100);
plt.grid()
plt.ylabel('cumulative number of IDs')
plt.xlabel('Number of previous applications per ID')
plt.title('Histogram of Number of previous applications for an ID')
```

Out[280]:
```
Text(0.5, 1.0, 'Histogram of Number of previous applications for an ID')
```



Can we differentiate applications by low, medium and high previous apps?

* Low = <5 claims (22%)
* Medium = 10 to 39 claims (58%)
* High = 40 or more claims (20%)

In [281…
```
apps_all = appsDF['SK_ID_CURR'].nunique()
apps_5plus = appsDF['SK_ID_CURR'].value_counts()>=5
```

```
apps_40plus = appsDF['SK_ID_CURR'].value_counts()>=40
print('Percentage with 10 or more previous apps:', np.round(100.*(sum(apps_5plu
print('Percentage with 40 or more previous apps:', np.round(100.*(sum(apps_40pl
```

```
Percentage with 10 or more previous apps: 41.76895
Percentage with 40 or more previous apps: 0.03453
```

Feature Engineering and Selection

In [15]:
```
appsDF = datasets["previous_application"]
```

In [17]:
```
class FeaturesAggregator(BaseEstimator, TransformerMixin):
    def __init__(self, file_name=None, features=None, funcs=None, primary_id =
        self.file_name = file_name
        self.features = features
        self.funcs = funcs
        self.primary_id = primary_id
        self.agg_op_features = {}
        for f in self.features:
            temp = {f"{file_name}_{f}_{func}":func for func in self.funcs}
            self.agg_op_features[f]=[(k, v) for k, v in temp.items()]
        print(self.agg_op_features)

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        #from IPython.core.debugger import Pdb as pdb;    pdb().set_trace() #br
        result = X.groupby([self.primary_id]).agg(self.agg_op_features)
        result.columns = result.columns.droplevel()
        result = result.reset_index(level=[self.primary_id])
        return result # return dataframe with the join key "SK_ID_CURR"
```

In [18]:
```
agg_funcs = ['min','max']
```

Feature Engineering on bureau balance

In [19]:
```
datasets['bureau_balance'].shape
```

Out[19]:
```
(27299925, 3)
```

In [20]:
```
bl_bureau = datasets['bureau_balance']
```

In [21]:
```
bl_bureau.shape
```

Out[21]:
```
(27299925, 3)
```

In [22]:
```
bl_bureau.columns
```

Out[22]:
```
Index(['SK_ID_BUREAU', 'MONTHS_BALANCE', 'STATUS'], dtype='object')
```

In [23]:
```
bl_bureau.isnull().sum()
```

Out[23]:
```
SK_ID_BUREAU       0
MONTHS_BALANCE     0
STATUS             0
dtype: int64
```

In [24]:
```
bl_features = ['MONTHS_BALANCE']
bl_pipeline = Pipeline([
    ('bureau_balance', FeaturesAggregator('bureau_balance' ,bl_features , agg_f
    ])
```

```
{'MONTHS_BALANCE': [('bureau_balance_MONTHS_BALANCE_min', 'min'), ('bureau_bal
ance_MONTHS_BALANCE_max', 'max')]}
```

In [25]:
```
bl_agg = bl_pipeline.fit_transform(bl_bureau)
```

In [26]:
```
bl_agg
```

Out[26]:

	SK_ID_BUREAU	bureau_balance_MONTHS_BALANCE_min	bureau_balance_MONTHS_BA
0	5001709	-96	
1	5001710	-82	
2	5001711	-3	
3	5001712	-18	
4	5001713	-21	
...	...	...	
817390	6842884	-47	
817391	6842885	-23	
817392	6842886	-32	
817393	6842887	-36	
817394	6842888	-61	

817395 rows × 3 columns

feature engineering on pos_cash_balance

In [27]:
```
datasets['POS_CASH_balance'].shape
```

Out[27]:
```
(10001358, 8)
```

In [28]:
```
pc_balance = datasets['POS_CASH_balance']
```

In [29]:
```
pc_balance.shape
```

Out[29]:
```
(10001358, 8)
```

In [30]:
```
pc_balance.columns
```

Out[30]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'MONTHS_BALANCE', 'CNT_INSTALMENT',
       'CNT_INSTALMENT_FUTURE', 'NAME_CONTRACT_STATUS', 'SK_DPD',
       'SK_DPD_DEF'],
      dtype='object')
```

In [31]:
```
pc_balance.isnull().sum()
```

Out[31]:
```
SK_ID_PREV                 0
SK_ID_CURR                 0
MONTHS_BALANCE             0
CNT_INSTALMENT         26071
CNT_INSTALMENT_FUTURE  26087
NAME_CONTRACT_STATUS       0
SK_DPD                     0
SK_DPD_DEF                 0
dtype: int64
```

In [32]:
```
pcb_features = ['MONTHS_BALANCE','CNT_INSTALMENT','CNT_INSTALMENT_FUTURE']
```

In [33]:
```
pcb_pipeline = Pipeline([
    ('POS_CASH_balance', FeaturesAggregator('POS_CASH_balance' ,pcb_features ,
    ])
```

```
{'MONTHS_BALANCE': [('POS_CASH_balance_MONTHS_BALANCE_min', 'min'), ('POS_CASH
_balance_MONTHS_BALANCE_max', 'max')], 'CNT_INSTALMENT': [('POS_CASH_balance_C
NT_INSTALMENT_min', 'min'), ('POS_CASH_balance_CNT_INSTALMENT_max', 'max')],
'CNT_INSTALMENT_FUTURE': [('POS_CASH_balance_CNT_INSTALMENT_FUTURE_min', 'mi
n'), ('POS_CASH_balance_CNT_INSTALMENT_FUTURE_max', 'max')]}
```

In [34]:
```
pcg_agg = pcb_pipeline.fit_transform(pc_balance)
```

In [35]:
```
pcg_agg
```

Out[35]:

	SK_ID_PREV	POS_CASH_balance_MONTHS_BALANCE_min	POS_CASH_balance_MONTH
0	1000001	-10	
1	1000002	-54	
2	1000003	-4	
3	1000004	-29	
4	1000005	-56	
...	...	...	
936320	2843494	-26	
936321	2843495	-16	
936322	2843497	-21	
936323	2843498	-48	
936324	2843499	-40	

936325 rows × 7 columns

feature engineering on credit card balance

In [36]:
```
datasets['credit_card_balance'].shape
```

Out[36]:
```
(3840312, 23)
```

In [37]:
```
cc_balance = datasets['credit_card_balance']
```

In [38]:
```
cc_balance.shape
```

Out[38]:
```
(3840312, 23)
```

In [39]:
```
cc_balance.columns
```

Out[39]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'MONTHS_BALANCE', 'AMT_BALANCE',
       'AMT_CREDIT_LIMIT_ACTUAL', 'AMT_DRAWINGS_ATM_CURRENT',
       'AMT_DRAWINGS_CURRENT', 'AMT_DRAWINGS_OTHER_CURRENT',
       'AMT_DRAWINGS_POS_CURRENT', 'AMT_INST_MIN_REGULARITY',
       'AMT_PAYMENT_CURRENT', 'AMT_PAYMENT_TOTAL_CURRENT',
       'AMT_RECEIVABLE_PRINCIPAL', 'AMT_RECIVABLE', 'AMT_TOTAL_RECEIVABLE',
       'CNT_DRAWINGS_ATM_CURRENT', 'CNT_DRAWINGS_CURRENT',
       'CNT_DRAWINGS_OTHER_CURRENT', 'CNT_DRAWINGS_POS_CURRENT',
       'CNT_INSTALMENT_MATURE_CUM', 'NAME_CONTRACT_STATUS', 'SK_DPD',
       'SK_DPD_DEF'],
      dtype='object')
```

In [40]:
```
cc_balance.isnull().sum()
```

Out[40]:
```
SK_ID_PREV                      0
SK_ID_CURR                      0
MONTHS_BALANCE                  0
AMT_BALANCE                     0
AMT_CREDIT_LIMIT_ACTUAL         0
AMT_DRAWINGS_ATM_CURRENT      749816
AMT_DRAWINGS_CURRENT            0
AMT_DRAWINGS_OTHER_CURRENT    749816
AMT_DRAWINGS_POS_CURRENT      749816
AMT_INST_MIN_REGULARITY       305236
AMT_PAYMENT_CURRENT           767988
AMT_PAYMENT_TOTAL_CURRENT       0
AMT_RECEIVABLE_PRINCIPAL        0
AMT_RECIVABLE                   0
AMT_TOTAL_RECEIVABLE            0
CNT_DRAWINGS_ATM_CURRENT      749816
CNT_DRAWINGS_CURRENT            0
CNT_DRAWINGS_OTHER_CURRENT    749816
CNT_DRAWINGS_POS_CURRENT      749816
CNT_INSTALMENT_MATURE_CUM     305236
NAME_CONTRACT_STATUS            0
SK_DPD                          0
SK_DPD_DEF                      0
dtype: int64
```

In [41]:
```
cc_balance['difference_payment'] = (cc_balance['AMT_PAYMENT_TOTAL_CURRENT'] - c
```

In [42]:
```
cc_balance['AMT_DRAWINGS_ratio'] = (cc_balance['AMT_DRAWINGS_CURRENT']/cc_balar
```

In [43]:
```
ccb_features = ['AMT_BALANCE','AMT_DRAWINGS_CURRENT','difference_payment','AMT_
```

In [44]:
```
ccb_pipeline = Pipeline([
    ('credit_card_balance', FeaturesAggregator('credit_card_balance' ,ccb_featu
    ])
```

```
{'AMT_BALANCE': [('credit_card_balance_AMT_BALANCE_min', 'min'), ('credit_card
_balance_AMT_BALANCE_max', 'max')], 'AMT_DRAWINGS_CURRENT': [('credit_card_bal
ance_AMT_DRAWINGS_CURRENT_min', 'min'), ('credit_card_balance_AMT_DRAWINGS_CUR
RENT_max', 'max')], 'difference_payment': [('credit_card_balance_difference_pa
yment_min', 'min'), ('credit_card_balance_difference_payment_max', 'max')], 'A
MT_DRAWINGS_ratio': [('credit_card_balance_AMT_DRAWINGS_ratio_min', 'min'),
('credit_card_balance_AMT_DRAWINGS_ratio_max', 'max')]}
```

In [45]:
```
ccb_agg = ccb_pipeline.fit_transform(cc_balance)
```

In [46]:
```
ccb_agg
```

Out[46]:

	SK_ID_PREV	credit_card_balance_AMT_BALANCE_min	credit_card_balance_AMT_BALA
0	1000018	38879.145	13
1	1000030	0.000	1(
2	1000031	0.000	15
3	1000035	0.000	
4	1000077	0.000	
...	...	...	
104302	2843476	0.000	17
104303	2843477	0.000	6
104304	2843478	0.000	7
104305	2843493	0.000	7
104306	2843496	0.000	18

104307 rows × 9 columns

feature engineering on installment payments

In [47]:
```
datasets['installments_payments'].shape
```

Out[47]:
```
(13605401, 8)
```

In [48]:
```
i_payments = datasets['installments_payments']
```

In [49]:
```
i_payments.shape
```

Out[49]:
```
(13605401, 8)
```

In [50]:
```
i_payments.isnull().sum()
```

Out[50]:
```
SK_ID_PREV                 0
SK_ID_CURR                 0
NUM_INSTALMENT_VERSION     0
NUM_INSTALMENT_NUMBER      0
DAYS_INSTALMENT            0
DAYS_ENTRY_PAYMENT      2905
AMT_INSTALMENT             0
AMT_PAYMENT             2905
dtype: int64
```

In [51]:
```
i_payments.columns
```

Out[51]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NUM_INSTALMENT_VERSION',
       'NUM_INSTALMENT_NUMBER', 'DAYS_INSTALMENT', 'DAYS_ENTRY_PAYMENT',
       'AMT_INSTALMENT', 'AMT_PAYMENT'],
      dtype='object')
```

In [52]:
```
i_payments['DAYS_INSTALMENT_diff'] = (i_payments['DAYS_INSTALMENT'] - i_payment
```

In [53]:
```
ip_features = ['DAYS_INSTALMENT','AMT_INSTALMENT','DAYS_INSTALMENT_diff']
```

In [54]:
```
ip_pipeline = Pipeline([
    ('installments_payments', FeaturesAggregator('installments_payments' ,ip_fe
    ])
```

```
{'DAYS_INSTALMENT': [('installments_payments_DAYS_INSTALMENT_min', 'min'), ('i
nstallments_payments_DAYS_INSTALMENT_max', 'max')], 'AMT_INSTALMENT': [('insta
llments_payments_AMT_INSTALMENT_min', 'min'), ('installments_payments_AMT_INST
ALMENT_max', 'max')], 'DAYS_INSTALMENT_diff': [('installments_payments_DAYS_IN
STALMENT_diff_min', 'min'), ('installments_payments_DAYS_INSTALMENT_diff_max',
'max')]}
```

In [55]:
```
ip_agg = ip_pipeline.fit_transform(i_payments)
```

In [56]:
```
ip_agg
```

Out[56]:

	SK_ID_PREV	installments_payments_DAYS_INSTALMENT_min	installments_payments_D
0	1000001	-268.0	
1	1000002	-1600.0	
2	1000003	-94.0	
3	1000004	-862.0	
4	1000005	-1688.0	
...	...	...	
997747	2843495	-439.0	
997748	2843496	-438.0	
997749	2843497	-588.0	
997750	2843498	-1442.0	
997751	2843499	-1203.0	

997752 rows × 7 columns

merging tertiary datasets with secondary datasets

In [57]:
```
prev_app = datasets['previous_application']
```

In [58]:
```
prev_app.shape
```

Out[58]:
```
(1670214, 37)
```

In [59]:
```
prev_app.isnull().sum()
```

Out[59]:
```
SK_ID_PREV                        0
SK_ID_CURR                        0
NAME_CONTRACT_TYPE                0
AMT_ANNUITY                  372235
AMT_APPLICATION                   0
AMT_CREDIT                        1
AMT_DOWN_PAYMENT             895844
AMT_GOODS_PRICE              385515
WEEKDAY_APPR_PROCESS_START        0
HOUR_APPR_PROCESS_START           0
FLAG_LAST_APPL_PER_CONTRACT       0
NFLAG_LAST_APPL_IN_DAY            0
RATE_DOWN_PAYMENT            895844
RATE_INTEREST_PRIMARY       1664263
RATE_INTEREST_PRIVILEGED    1664263
NAME_CASH_LOAN_PURPOSE            0
NAME_CONTRACT_STATUS              0
DAYS_DECISION                     0
NAME_PAYMENT_TYPE                 0
CODE_REJECT_REASON                0
NAME_TYPE_SUITE              820405
NAME_CLIENT_TYPE                  0
NAME_GOODS_CATEGORY               0
NAME_PORTFOLIO                    0
NAME_PRODUCT_TYPE                 0
CHANNEL_TYPE                      0
SELLERPLACE_AREA                  0
NAME_SELLER_INDUSTRY              0
CNT_PAYMENT                  372230
NAME_YIELD_GROUP                  0
PRODUCT_COMBINATION             346
DAYS_FIRST_DRAWING           673065
DAYS_FIRST_DUE               673065
DAYS_LAST_DUE_1ST_VERSION    673065
DAYS_LAST_DUE                673065
DAYS_TERMINATION             673065
NFLAG_INSURED_ON_APPROVAL    673065
dtype: int64
```

In [61]:
```
prev_app.shape
```

Out[61]:
```
(1670214, 37)
```

In [62]:
```
prev_app['AMT_ANNUITY_ratio'] = (prev_app['AMT_ANNUITY']/prev_app['AMT_CREDIT']
```

In [69]:
```
prev_app = prev_app.merge(pcg_agg, how='left', on="SK_ID_PREV")
```

In [70]:
```
prev_app = prev_app.merge(ip_agg, how='left', on="SK_ID_PREV")
```

In [71]:
```
prev_app = prev_app.merge(ccb_agg, how='left', on="SK_ID_PREV")
```

In [72]:
```
prev_app.shape
```

Out[72]:
```
(1670214, 58)
```

In [73]:
```
pa_features = ['AMT_APPLICATION','AMT_CREDIT','AMT_ANNUITY','AMT_ANNUITY_ratio'
```

In [74]:
```
pa_features = pa_features + pcg_agg.columns[1:].tolist() + ccb_agg.columns[1:].
```

In [75]:
```
print(len(pa_features))
```

```
24
```

In [76]:
```
agg_funcs1 = ['count','max','min','sum']
```

In [77]:
```
pa_pipeline = Pipeline([
    ('previous_application', FeaturesAggregator('previous_application' ,pa_feat
    ])
```

{'AMT_APPLICATION': [('previous_application_AMT_APPLICATION_count', 'count'), ('previous_application_AMT_APPLICATION_max', 'max'), ('previous_application_AMT_APPLICATION_min', 'min'), ('previous_application_AMT_APPLICATION_sum', 'sum')], 'AMT_CREDIT': [('previous_application_AMT_CREDIT_count', 'count'), ('previous_application_AMT_CREDIT_max', 'max'), ('previous_application_AMT_CREDIT_min', 'min'), ('previous_application_AMT_CREDIT_sum', 'sum')], 'AMT_ANNUITY': [('previous_application_AMT_ANNUITY_count', 'count'), ('previous_application_AMT_ANNUITY_max', 'max'), ('previous_application_AMT_ANNUITY_min', 'min'), ('previous_application_AMT_ANNUITY_sum', 'sum')], 'AMT_ANNUITY_ratio': [('previous_application_AMT_ANNUITY_ratio_count', 'count'), ('previous_application_AMT_ANNUITY_ratio_max', 'max'), ('previous_application_AMT_ANNUITY_ratio_min', 'min'), ('previous_application_AMT_ANNUITY_ratio_sum', 'sum')], 'POS_CASH_balance_MONTHS_BALANCE_min': [('previous_application_POS_CASH_balance_MONTHS_BALANCE_min_count', 'count'), ('previous_application_POS_CASH_balance_MONTHS_BALANCE_min_max', 'max'), ('previous_application_POS_CASH_balance_MONTHS_BALANCE_min_min', 'min'), ('previous_application_POS_CASH_balance_MONTHS_BALANCE_min_sum', 'sum')], 'POS_CASH_balance_MONTHS_BALANCE_max': [('previous_application_POS_CASH_balance_MONTHS_BALANCE_max_count', 'count'), ('previous_application_POS_CASH_balance_MONTHS_BALANCE_max_max', 'max'), ('previous_application_POS_CASH_balance_MONTHS_BALANCE_max_min', 'min'), ('previous_application_POS_CASH_balance_MONTHS_BALANCE_max_sum', 'sum')], 'POS_CASH_balance_CNT_INSTALMENT_min': [('previous_application_POS_CASH_balance_CNT_INSTALMENT_min_count', 'count'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_min_max', 'max'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_min_min', 'min'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_min_sum', 'sum')], 'POS_CASH_balance_CNT_INSTALMENT_max': [('previous_application_POS_CASH_balance_CNT_INSTALMENT_max_count', 'count'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_max_max', 'max'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_max_min', 'min'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_max_sum', 'sum')], 'POS_CASH_balance_CNT_INSTALMENT_FUTURE_min': [('previous_application_POS_CASH_balance_CNT_INSTALMENT_FUTURE_min_count', 'count'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_FUTURE_min_max', 'max'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_FUTURE_min_min', 'min'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_FUTURE_min_sum', 'sum')], 'POS_CASH_balance_CNT_INSTALMENT_FUTURE_max': [('previous_application_POS_CASH_balance_CNT_INSTALMENT_FUTURE_max_count', 'count'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_FUTURE_max_max', 'max'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_FUTURE_max_min', 'min'), ('previous_application_POS_CASH_balance_CNT_INSTALMENT_FUTURE_max_sum', 'sum')], 'credit_card_balance_AMT_BALANCE_min': [('previous_application_credit_card_balance_AMT_BALANCE_min_count', 'count'), ('previous_application_credit_card_balance_AMT_BALANCE_min_max', 'max'), ('previous_application_credit_card_balance_AMT_BALANCE_min_min', 'min'), ('previous_application_credit_card_balance_AMT_BALANCE_min_sum', 'sum')], 'credit_card_balance_AMT_BALANCE_max': [('previous_application_credit_card_balance_AMT_BALANCE_max_count', 'count'), ('previous_application_credit_card_balance_AMT_BALANCE_max_max', 'max'), ('previous_application_credit_card_balance_AMT_BALANCE_max_min', 'min'), ('previous_application_credit_card_balance_AMT_BALANCE_max_sum', 'sum')], 'credit_card_balance_AMT_DRAWINGS_CURRENT_min': [('previous_application_credit_card_balance_AMT_DRAWINGS_CURRENT_min_count', 'count'), ('previous_application_credit_card_balance_AMT_DRAWINGS_CURRENT_min_max', 'max'), ('previous_application_credit_card_balance_AMT_DRAWINGS_CURRENT_min_min', 'min'), ('previous_application_credit_card_balance_AMT_DRAWINGS_CURRENT_min_sum', 'sum')], 'credit_card_balance_AMT_DRAWINGS_CURRENT_max': [('previous_application_credit_card_balance_AMT_DRAWINGS_CURRENT_max_count', 'count'), ('previous_application_credit_card_balance_AMT_DRAWINGS_CURRENT_max_max', 'max'), ('previous_application_credit_card_balance_AMT_DRAWINGS_CURRENT_max_min', 'min'), ('previous_application_credit_card_balance_AMT_DRAWINGS_CURRENT_max_sum', 'sum')], 'credit_card_balance_difference_payment_min': [('previous_application_credit_card_balance_difference_payment_min_count', 'count'), ('previous_application_credit_card_balance_difference_payment_min_max', 'max'), ('previous_app

```
lication_credit_card_balance_difference_payment_min_min', 'min'), ('previous_a
pplication_credit_card_balance_difference_payment_min_sum', 'sum')], 'credit_c
ard_balance_difference_payment_max': [('previous_application_credit_card_balan
ce_difference_payment_max_count', 'count'), ('previous_application_credit_card
_balance_difference_payment_max_max', 'max'), ('previous_application_credit_ca
rd_balance_difference_payment_max_min', 'min'), ('previous_application_credit_
card_balance_difference_payment_max_sum', 'sum')], 'credit_card_balance_AMT_DR
AWINGS_ratio_min': [('previous_application_credit_card_balance_AMT_DRAWINGS_ra
tio_min_count', 'count'), ('previous_application_credit_card_balance_AMT_DRAWI
NGS_ratio_min_max', 'max'), ('previous_application_credit_card_balance_AMT_DRA
WINGS_ratio_min_min', 'min'), ('previous_application_credit_card_balance_AMT_D
RAWINGS_ratio_min_sum', 'sum')], 'credit_card_balance_AMT_DRAWINGS_ratio_max':
[('previous_application_credit_card_balance_AMT_DRAWINGS_ratio_max_count', 'co
unt'), ('previous_application_credit_card_balance_AMT_DRAWINGS_ratio_max_max',
'max'), ('previous_application_credit_card_balance_AMT_DRAWINGS_ratio_max_mi
n', 'min'), ('previous_application_credit_card_balance_AMT_DRAWINGS_ratio_max_
sum', 'sum')], 'installments_payments_DAYS_INSTALMENT_min': [('previous_applic
ation_installments_payments_DAYS_INSTALMENT_min_count', 'count'), ('previous_a
pplication_installments_payments_DAYS_INSTALMENT_min_max', 'max'), ('previous_
application_installments_payments_DAYS_INSTALMENT_min_min', 'min'), ('previous
_application_installments_payments_DAYS_INSTALMENT_min_sum', 'sum')], 'install
ments_payments_DAYS_INSTALMENT_max': [('previous_application_installments_paym
ents_DAYS_INSTALMENT_max_count', 'count'), ('previous_application_installments
_payments_DAYS_INSTALMENT_max_max', 'max'), ('previous_application_installment
s_payments_DAYS_INSTALMENT_max_min', 'min'), ('previous_application_installmen
ts_payments_DAYS_INSTALMENT_max_sum', 'sum')], 'installments_payments_AMT_INST
ALMENT_min': [('previous_application_installments_payments_AMT_INSTALMENT_min_
count', 'count'), ('previous_application_installments_payments_AMT_INSTALMENT_
min_max', 'max'), ('previous_application_installments_payments_AMT_INSTALMENT_
min_min', 'min'), ('previous_application_installments_payments_AMT_INSTALMENT_
min_sum', 'sum')], 'installments_payments_AMT_INSTALMENT_max': [('previous_app
lication_installments_payments_AMT_INSTALMENT_max_count', 'count'), ('previous
_application_installments_payments_AMT_INSTALMENT_max_max', 'max'), ('previous
_application_installments_payments_AMT_INSTALMENT_max_min', 'min'), ('previous
_application_installments_payments_AMT_INSTALMENT_max_sum', 'sum')], 'installm
ents_payments_DAYS_INSTALMENT_diff_min': [('previous_application_installments_
payments_DAYS_INSTALMENT_diff_min_count', 'count'), ('previous_application_ins
tallments_payments_DAYS_INSTALMENT_diff_min_max', 'max'), ('previous_applicati
on_installments_payments_DAYS_INSTALMENT_diff_min_min', 'min'), ('previous_app
lication_installments_payments_DAYS_INSTALMENT_diff_min_sum', 'sum')], 'instal
lments_payments_DAYS_INSTALMENT_diff_max': [('previous_application_installment
s_payments_DAYS_INSTALMENT_diff_max_count', 'count'), ('previous_application_i
nstallments_payments_DAYS_INSTALMENT_diff_max_max', 'max'), ('previous_applica
tion_installments_payments_DAYS_INSTALMENT_diff_max_min', 'min'), ('previous_a
pplication_installments_payments_DAYS_INSTALMENT_diff_max_sum', 'sum')]}
```

In [78]:
```
pa_agg = pa_pipeline.fit_transform(prev_app)
```

In [79]:
```
pa_agg
```

Out[79]:

	SK_ID_CURR	previous_application_AMT_APPLICATION_count	previous_application_AMT
0	100001	1	
1	100002	1	
2	100003	3	
3	100004	1	
4	100005	2	
...	...	...	
338852	456251	1	
338853	456252	1	
338854	456253	2	
338855	456254	2	
338856	456255	8	

338857 rows × 97 columns

Domain based features

In [80]:
```
pa_agg['previous_application_AMT_APPLICATION_avg'] = (
   pa_agg['previous_application_AMT_APPLICATION_sum'] / pa_agg['previous_applic

pa_agg['previous_application_AMT_APPLICATION_range'] = (
   pa_agg['previous_application_AMT_APPLICATION_max'] - pa_agg['previous_applic
```

In [81]:
```
pa_agg.shape
```

Out[81]:
```
(338857, 99)
```

merging bureau balance with bureau

In [82]:
```
datasets['bureau'].shape
```

Out[82]:
```
(1716428, 17)
```

In [83]:
```
bureau = datasets['bureau']
```

In [84]:
```
bureau.isnull().sum()
```

Out[84]:
```
SK_ID_CURR                    0
SK_ID_BUREAU                  0
CREDIT_ACTIVE                 0
CREDIT_CURRENCY               0
DAYS_CREDIT                   0
CREDIT_DAY_OVERDUE            0
DAYS_CREDIT_ENDDATE      105553
DAYS_ENDDATE_FACT        633653
AMT_CREDIT_MAX_OVERDUE  1124488
CNT_CREDIT_PROLONG            0
AMT_CREDIT_SUM               13
AMT_CREDIT_SUM_DEBT      257669
AMT_CREDIT_SUM_LIMIT     591780
AMT_CREDIT_SUM_OVERDUE        0
CREDIT_TYPE                   0
DAYS_CREDIT_UPDATE            0
AMT_ANNUITY             1226791
dtype: int64
```

In [85]:
```
# df_missing = pd.DataFrame(np.round((bureau.isna().sum()/ bureau.shape[0]) * 1
# df_missing_50_cols = df_missing[df_missing.Percent >= 50].index

# # Drop
# bureau.drop(columns=df_missing_50_cols, inplace=True)
```

In [86]:
```
bureau.shape
```

Out[86]:
```
(1716428, 17)
```

In [87]:
```
bureau = bureau.merge(bl_agg, on = "SK_ID_BUREAU", how = 'left')
```

In [88]:
```
bureau.shape
```

Out[88]:
```
(1716428, 19)
```

In [89]:
```
bureau_features = ['AMT_CREDIT_SUM']
```

In [90]:
```
bureau_features = bureau_features + bl_agg.columns[1:].tolist()
```

In [91]:
```
len(bureau_features)
```

Out[91]:
```
3
```

In [92]:
```
bureau_pipeline = Pipeline([
    ('bureau', FeaturesAggregator('bureau' ,bureau_features , agg_funcs1, "SK_I
    ])
```

```
{'AMT_CREDIT_SUM': [('bureau_AMT_CREDIT_SUM_count', 'count'), ('bureau_AMT_CRE
DIT_SUM_max', 'max'), ('bureau_AMT_CREDIT_SUM_min', 'min'), ('bureau_AMT_CREDI
T_SUM_sum', 'sum')], 'bureau_balance_MONTHS_BALANCE_min': [('bureau_bureau_bal
ance_MONTHS_BALANCE_min_count', 'count'), ('bureau_bureau_balance_MONTHS_BALAN
CE_min_max', 'max'), ('bureau_bureau_balance_MONTHS_BALANCE_min_min', 'min'),
('bureau_bureau_balance_MONTHS_BALANCE_min_sum', 'sum')], 'bureau_balance_MONT
HS_BALANCE_max': [('bureau_bureau_balance_MONTHS_BALANCE_max_count', 'count'),
('bureau_bureau_balance_MONTHS_BALANCE_max_max', 'max'), ('bureau_bureau_balan
ce_MONTHS_BALANCE_max_min', 'min'), ('bureau_bureau_balance_MONTHS_BALANCE_max
_sum', 'sum')]}
```

In [93]:
```
bureau_agg = bureau_pipeline.fit_transform(bureau)
```

In [94]:
```
bureau_agg
```

Out[94]:

	SK_ID_CURR	bureau_AMT_CREDIT_SUM_count	bureau_AMT_CREDIT_SUM_max	bureau
0	100001	7	378000.00	
1	100002	8	450000.00	
2	100003	4	810000.00	
3	100004	2	94537.80	
4	100005	3	568800.00	
...	...	...	...	
305806	456249	13	765000.00	
305807	456250	3	2153110.05	
305808	456253	4	2250000.00	
305809	456254	1	45000.00	
305810	456255	11	900000.00	

305811 rows × 13 columns

Domain knowledge based features

In [95]:
```
bureau_agg['bureau_AMT_CREDIT_SUM_avg'] = (
   bureau_agg['bureau_AMT_CREDIT_SUM_sum'] / bureau_agg['bureau_AMT_CREDIT_SUM_

bureau_agg['bureau_AMT_APPLICATION_range'] = (
   bureau_agg['bureau_AMT_CREDIT_SUM_max'] - bureau_agg['bureau_AMT_CREDIT_SUM_
```

In [96]:
```
bureau_agg.shape
```

Out[96]:
```
(305811, 15)
```

Merging secondary with primary datasets

In [97]:
```
datasets['application_train'].shape
```

Out[97]:
```
(307511, 122)
```

In [98]:
```
app_train = datasets['application_train']
```

In [99]:
```
df_missing = pd.DataFrame(np.round((app_train.isna().sum()/ app_train.shape[0])
df_missing_50_cols = df_missing[df_missing.Percent >= 50].index

# Drop
app_train.drop(columns=df_missing_50_cols, inplace=True)
```

In [100…
```
app_train.shape
```
Out[100]: (307511, 81)

In [101…
```
pa_agg.shape
```
Out[101]: (338857, 99)

In [103…
```
pa_agg.shape
```
Out[103]: (338857, 99)

In [104…
```
bureau_agg.shape
```
Out[104]: (305811, 15)

In [106…
```
bureau_agg.shape
```
Out[106]: (305811, 15)

In [107…
```
app_train = app_train.merge(pa_agg, on = "SK_ID_CURR", how = 'left')
```

In [108…
```
app_train = app_train.merge(bureau_agg, on = "SK_ID_CURR", how = 'left')
```

In [109…
```
app_train.shape
```
Out[109]: (307511, 193)

In [111…
```
app_train.shape
```
Out[111]: (307511, 193)

In [112…
```
app_train['CREDIT_INCOME_PCT'] = app_train['AMT_CREDIT'] / app_train['AMT_INCOM
app_train['CREDIT_TERM'] = app_train['AMT_ANNUITY'] / app_train['AMT_CREDIT']
```

filling missing values with 0

In [113…
```
app_train[pa_agg.columns] = app_train[pa_agg.columns].fillna(0)
```

In [114…
```
app_train[bureau_agg.columns] = app_train[bureau_agg.columns].fillna(0)
```

In [115…
```
app_train.shape
```
Out[115]: (307511, 195)

In [116…
```
num_features = []

for col in app_train.columns:
    if app_train[col].dtype == 'int64' or app_train[col].dtype == 'float64':
```

```
        num_features.append(col)
len(num_features)
```

Out[116]:
```
182
```

In [117…
```
cat_cols2 = []

for col in app_train.columns:
    if app_train[col].dtype == 'object':
        cat_cols2.append(col)
len(cat_cols2)
```

Out[117]:
```
13
```

In [118…
```
total_features = num_features+cat_cols2
```

In [119…
```
app_train.isnull().sum()
```

Out[119]:
```
SK_ID_CURR                                    0
TARGET                                        0
NAME_CONTRACT_TYPE                            0
CODE_GENDER                                   0
FLAG_OWN_CAR                                  0
                                             ..
bureau_bureau_balance_MONTHS_BALANCE_max_sum  0
bureau_AMT_CREDIT_SUM_avg                     0
bureau_AMT_APPLICATION_range                  0
CREDIT_INCOME_PCT                             0
CREDIT_TERM                                  12
Length: 195, dtype: int64
```

Finding correlations for feature selection

In [120…
```
corr_with_target = app_train.corr()['TARGET'].abs().sort_values()
```

In [121…
```
corr_with_target
```

Out[121]:
```
FLAG_DOCUMENT_20                                                    0.0
00215
FLAG_DOCUMENT_5                                                     0.0
00316
FLAG_CONT_MOBILE                                                    0.0
00370
FLAG_MOBIL                                                          0.0
00534
previous_application_installments_payments_DAYS_INSTALMENT_diff_max_max    0.0
00604

...
REGION_RATING_CLIENT_W_CITY                                         0.0
60893
DAYS_BIRTH                                                          0.0
78239
EXT_SOURCE_2                                                        0.1
60472
EXT_SOURCE_3                                                        0.1
78919
TARGET                                                              1.0
00000
Name: TARGET, Length: 182, dtype: float64
```

In [122…
```
corr_with_target.tail(6)
```

Out[122]:
```
REGION_RATING_CLIENT           0.058899
REGION_RATING_CLIENT_W_CITY    0.060893
DAYS_BIRTH                     0.078239
EXT_SOURCE_2                   0.160472
EXT_SOURCE_3                   0.178919
TARGET                         1.000000
Name: TARGET, dtype: float64
```

In [123…
```
print("Top 40 correlated features are:")
high_corr = corr_with_target.tail(31)
```

```
Top 40 correlated features are:
```

In [124…
```
high_corr_list = high_corr.index.tolist()
```

In [125…
```
high_corr_list
```

Out[125]:
```
['previous_application_POS_CASH_balance_MONTHS_BALANCE_max_sum',
 'AMT_GOODS_PRICE',
 'previous_application_POS_CASH_balance_MONTHS_BALANCE_min_min',
 'previous_application_installments_payments_DAYS_INSTALMENT_min_min',
 'previous_application_credit_card_balance_AMT_BALANCE_max_min',
 'previous_application_credit_card_balance_AMT_BALANCE_max_sum',
 'previous_application_credit_card_balance_AMT_BALANCE_max_max',
 'DAYS_REGISTRATION',
 'previous_application_installments_payments_DAYS_INSTALMENT_max_min',
 'previous_application_POS_CASH_balance_MONTHS_BALANCE_max_min',
 'FLOORSMAX_MODE',
 'FLOORSMAX_MEDI',
 'previous_application_credit_card_balance_difference_payment_min_count',
 'previous_application_credit_card_balance_difference_payment_max_count',
 'FLOORSMAX_AVG',
 'FLAG_DOCUMENT_3',
 'REG_CITY_NOT_LIVE_CITY',
 'previous_application_credit_card_balance_AMT_DRAWINGS_ratio_max_min',
 'previous_application_credit_card_balance_AMT_DRAWINGS_ratio_max_max',
 'previous_application_credit_card_balance_AMT_DRAWINGS_ratio_max_sum',
 'DAYS_EMPLOYED',
 'FLAG_EMP_PHONE',
 'REG_CITY_NOT_WORK_CITY',
 'DAYS_ID_PUBLISH',
 'DAYS_LAST_PHONE_CHANGE',
 'REGION_RATING_CLIENT',
 'REGION_RATING_CLIENT_W_CITY',
 'DAYS_BIRTH',
 'EXT_SOURCE_2',
 'EXT_SOURCE_3',
 'TARGET']
```

Modeling pipelines

In [126…
```
app_train_df = app_train
```

In [127…
```
class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].values
```

In [128…
```
num_attribs=app_train_df.select_dtypes(include=['int64', 'float64']).columns.to
```

In [129…
```
num_attribs.remove('TARGET')
```

In [130…
```
cat_attribs = app_train_df.select_dtypes(exclude=['float64','int64']).columns.t
```

In [131…
```
from sklearn.preprocessing import OrdinalEncoder

# Identify the numeric features we wish to consider.
num_attribs = num_attribs

num_pipeline = Pipeline([
```

```
        ('selector', DataFrameSelector(num_attribs)),
        ('imputer', SimpleImputer(strategy='mean')),
        ('std_scaler', StandardScaler()),
    ])
# Identify the categorical features we wish to consider.
cat_attribs = cat_attribs

cat_pipeline = Pipeline([
        ('selector', DataFrameSelector(cat_attribs)),
        ('imputer', SimpleImputer(strategy='most_frequent')),
        #('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
        ('ohe', OneHotEncoder(sparse=False, handle_unknown="ignore"))
    ])

data_prep_pipeline = FeatureUnion(transformer_list=[
        ("num_pipeline", num_pipeline),
        ("cat_pipeline", cat_pipeline),
    ])
```

In [132…
```
selected_features = num_attribs+cat_attribs
```

In [133…
```
#from sklearn.ensemble import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

In [134…
```
def pct(x):
    return round(100*x,3)
```

In [174…
```
try:
    expLog
except NameError:
    expLog = pd.DataFrame(columns=["exp_name",
                                   "Train Acc",

                                   "Test  Acc",
                                   "Train AUC",

                                   "Test  AUC",
                                   "Train F1",
                                   "Test F1"
                                  ])
```

In [175…
```
#del expLog# total_features.remove('TARGET')
```

In [176…
```
expLog
```

Out[176]:

exp_name	Train Acc	Test Acc	Train AUC	Test AUC	Train F1	Test F1

In [138…
```
splits = 50



finaldf = np.array_split(app_train_df, splits)
X_train = finaldf[0][selected_features]
y_train = finaldf[0]['TARGET']
```

```
# X_kaggle_test= X_kaggle_test[selected_features]

## split part of data
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, stratify=
                                                    test_size=0.3, random_state

# X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train,strati

# print(f"X train           shape: {X_train.shape}")
# print(f"X validation      shape: {X_valid.shape}")
# print(f"X test            shape: {X_test.shape}")
# print(f"X kaggle_test     shape: {X_kaggle_test.shape}")
```

In [139…
```
X_train.shape
```

Out[139]:
```
(4305, 194)
```

In [140…
```
X_test.shape
```

Out[140]:
```
(1846, 194)
```

In [141…
```
y_train.shape
```

Out[141]:
```
(4305,)
```

Logistic Regression

In [142…
```
%%time
np.random.seed(42)
full_pipeline_with_predictor = Pipeline([
        ("preparation", data_prep_pipeline),
        ("linear", LogisticRegression())
    ])
```

```
CPU times: user 297 µs, sys: 0 ns, total: 297 µs
Wall time: 304 µs
```

In [143…
```
from sklearn.model_selection import ShuffleSplit
cvSplits = ShuffleSplit(n_splits=5, test_size=0.3, random_state=42)
```

In [146…
```
import time

start = time.time()
model_lr = full_pipeline_with_predictor.fit(X_train, y_train)
np.random.seed(42)


logit_scores = cross_val_score(full_pipeline_with_predictor,X_train , y_train,c
logit_score_train = pct(logit_scores.mean())
train_time = np.round(time.time() - start, 4)


start = time.time()
logit_score_test  = full_pipeline_with_predictor.score(X_test, y_test)
test_time = np.round(time.time() - start, 4)
```

In [147…
```
# model_lr = full_pipeline_with_predictor.fit(X_train,y_train)
```

In [148…
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, log_los


np.round(accuracy_score(y_train, model_lr.predict(X_train)), 3)
```

Out[148]:
```
0.924
```

In [151…
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_train, model_lr.predict_proba(X_train)[:, 1])
```

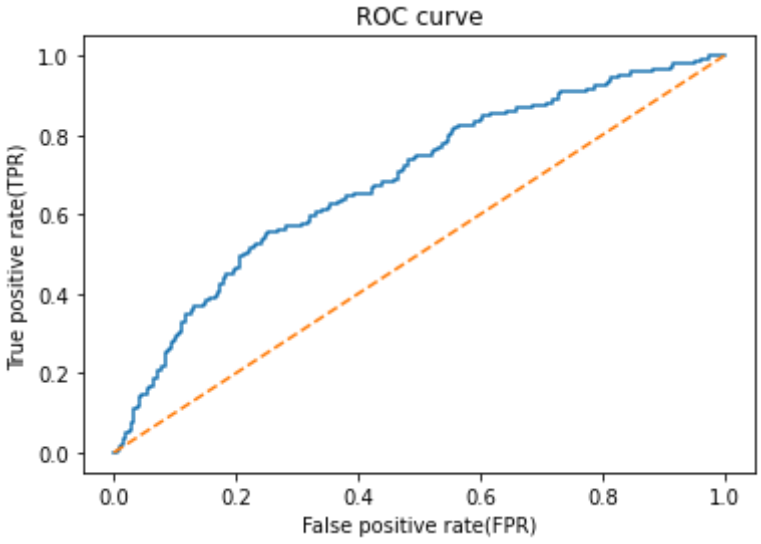
Out[151]:
```
0.846483702394827
```

In [177…
```
exp_name = "Logistic Regression"
expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
               [accuracy_score(y_train, model_lr.predict(X_train)),
                accuracy_score(y_test, model_lr.predict(X_test)),
                roc_auc_score(y_train, model_lr.predict_proba(X_train)[:, 1]),
                roc_auc_score(y_test, model_lr.predict_proba(X_test)[:, 1]),
                f1_score(y_train, model_lr.predict(X_train), average='weighted'
                f1_score(y_test, model_lr.predict(X_test), average='weighted')]
    4))
expLog
```

Out[177]:

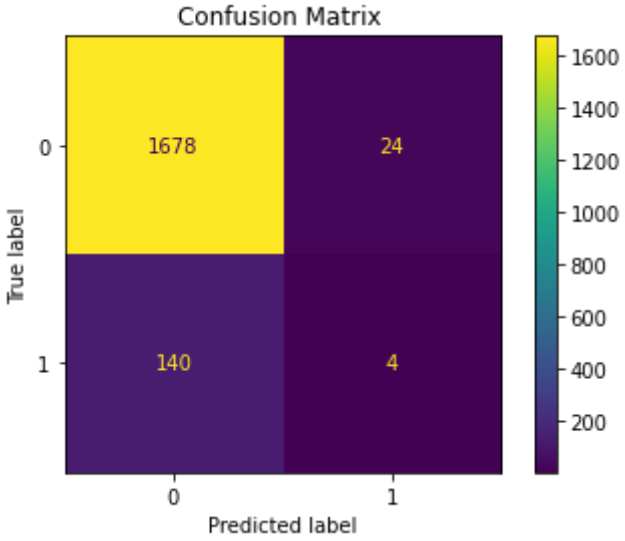
	exp_name	Train Acc	Test Acc	Train AUC	Test AUC	Train F1	Test F1
0	Logistic Regression	0.924	0.9112	0.8465	0.6855	0.8972	0.8827

In [157…
```
from sklearn import metrics
fpr,tpr,thresh = metrics.roc_curve(y_test,model_lr.predict_proba(X_test)[:,1])
plt.plot(fpr,tpr)
plt.plot(np.arange(0,1.1,0.1),np.arange(0,1.1,0.1),linestyle="dashed")
plt.xlabel("False positive rate(FPR)")
plt.ylabel("True positive rate(TPR)")
plt.title("ROC curve")
plt.show()
```



In [159…
```
from sklearn.metrics import plot_confusion_matrix
plt.clf()
plot_confusion_matrix(model_lr,X_test,y_test)
plt.title('Confusion Matrix ')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Random Forest with hyper parameter tuning

In [160…
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV


n_estimators = [100,200]
max_depth = [10,20]
max_features=[2,3]



random_grid = {'rf__n_estimators': n_estimators,
               'rf__max_depth': max_depth,
               'rf__max_features':max_features}

print(random_grid)
```

```
{'rf__n_estimators': [100, 200], 'rf__max_depth': [10, 20], 'rf__max_feature
s': [2, 3]}
```

In [161…
```
%%time
np.random.seed(42)
full_pipeline_with_predictor_1 = Pipeline([
        ("preparation", data_prep_pipeline),
        ('rf', RandomForestClassifier())
    ])
rf_random = GridSearchCV(full_pipeline_with_predictor_1, param_grid = random_gr
```

```
CPU times: user 324 µs, sys: 48 µs, total: 372 µs
Wall time: 383 µs
```

In [162…
```
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.5s
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.5s
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.5s
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.4s
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.5s
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=200; total tim
e=   0.9s
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=200; total tim
e=   0.9s
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=200; total tim
e=   0.8s
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=200; total tim
e=   0.8s
[CV] END rf__max_depth=10, rf__max_features=2, rf__n_estimators=200; total tim
e=   0.8s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.5s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.5s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.5s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.5s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.5s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.0s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.0s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.0s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.0s
[CV] END rf__max_depth=10, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.0s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.6s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.6s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.6s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.6s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=100; total tim
e=   0.6s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=200; total tim
e=   1.0s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=200; total tim
e=   1.0s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=200; total tim
e=   1.1s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=200; total tim
e=   1.0s
[CV] END rf__max_depth=20, rf__max_features=2, rf__n_estimators=200; total tim
```

```
e=   1.1s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.7s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.7s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.6s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.7s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=100; total tim
e=   0.6s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.2s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.2s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.2s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.2s
[CV] END rf__max_depth=20, rf__max_features=3, rf__n_estimators=200; total tim
e=   1.2s
```

Out[162]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('preparation',
                                        FeatureUnion(transformer_list=[('num_p
ipeline',
                                                                        Pipeli
ne(steps=[('selector',

DataFrameSelector(attribute_names=['SK_ID_CURR',

'CNT_CHILDREN',

'AMT_INCOME_TOTAL',

'AMT_CREDIT',

'AMT_ANNUITY',

'AMT_GOODS_PRICE',

'REGION_POPULATION_RELATIVE',

'DAYS_BIRTH',

'DAYS_EMPLOYED',

'DAYS_REGISTRATION',

'DAYS_ID_PU...

'OCCUPATION_TYPE',

'WEEKDAY_APPR_PROCESS_START',

'ORGANIZATION_TYPE',

'EMERGENCYSTATE_MODE'])),

('imputer',

SimpleImputer(strategy='most_frequent')),

('ohe',

OneHotEncoder(handle_unknown='ignore',

sparse=False))]))])),
                                       ('rf', RandomForestClassifier())]),
             param_grid={'rf__max_depth': [10, 20], 'rf__max_features': [2,
3],
                         'rf__n_estimators': [100, 200]},
             scoring='accuracy', verbose=2)
```

In [163…
```
rf_random.best_params_
```

Out[163]:
```
{'rf__max_depth': 10, 'rf__max_features': 2, 'rf__n_estimators': 100}
```

In [168…
```
full_pipeline_with_predictor_1 = Pipeline([
        ("preparation", data_prep_pipeline),
        ('rf', RandomForestClassifier(n_estimators=100,max_features=2,max_depth
```

```
    ])
model_rf = full_pipeline_with_predictor_1.fit(X_train,y_train)
```

In [169…
```
np.round(accuracy_score(y_train,model_rf.predict(X_train)),3)
```

Out[169]:
```
0.923
```

In [170…
```
roc_auc_score(y_train, model_rf.predict_proba(X_train)[:, 1])
```

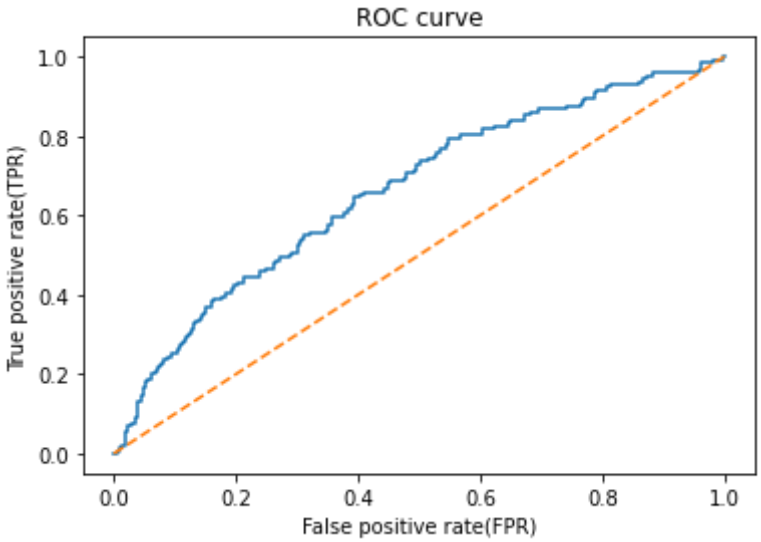
Out[170]:
```
0.9974826121282754
```

In [178…
```
exp_name = "Random Forest Model"
expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
                [accuracy_score(y_train, model_rf.predict(X_train)),
                 accuracy_score(y_test, model_rf.predict(X_test)),
                 roc_auc_score(y_train, model_rf.predict_proba(X_train)[:, 1]),
                 roc_auc_score(y_test, model_rf.predict_proba(X_test)[:, 1]),
                 f1_score(y_train, model_rf.predict(X_train), average='weighted'
                 f1_score(y_test, model_rf.predict(X_test), average='weighted')]
    4))
expLog
```

Out[178]:

	exp_name	Train Acc	Test Acc	Train AUC	Test AUC	Train F1	Test F1
0	Logistic Regression	0.9240	0.9112	0.8465	0.6855	0.8972	0.8827
1	Random Forest Model	0.9226	0.9220	0.9975	0.6616	0.8860	0.8846

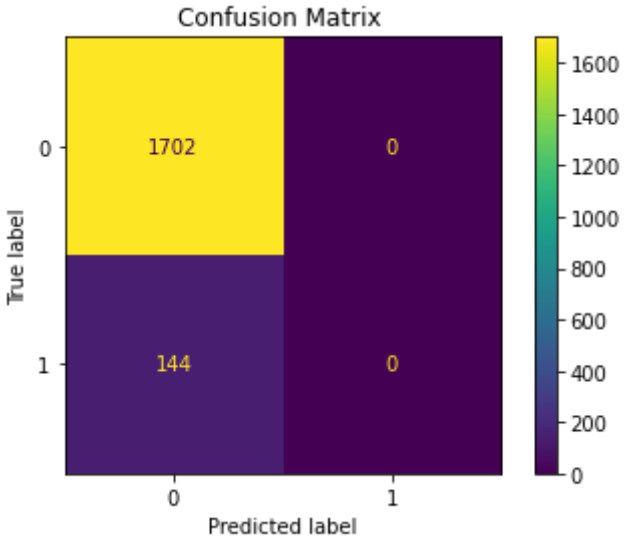
In [179…
```
from sklearn import metrics
fpr,tpr,thresh = metrics.roc_curve(y_test,model_rf.predict_proba(X_test)[:,1])
plt.plot(fpr,tpr)
plt.plot(np.arange(0,1.1,0.1),np.arange(0,1.1,0.1),linestyle="dashed")
plt.xlabel("False positive rate(FPR)")
plt.ylabel("True positive rate(TPR)")
plt.title("ROC curve")
plt.show()
```



In [180…
```
from sklearn.metrics import plot_confusion_matrix
plt.clf()
plot_confusion_matrix(model_rf,X_test,y_test)
```

```
plt.title('Confusion Matrix ')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



XGBoost with hyper parameter tuning

In [181…

```
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
import xgboost
params={
 "xgb__learning_rate"    : [0.05, 0.10] ,
 "xgb__max_depth"        : [2,3,5],
 "xgb__min_child_weight" : [ 1, 3]
}
classifier=xgboost.XGBClassifier()
full_pipeline_with_predictor_xgb = Pipeline([
        ("preparation", data_prep_pipeline),
        ("xgb", xgboost.XGBClassifier())
    ])
random_search=RandomizedSearchCV(full_pipeline_with_predictor_xgb,param_distrib
random_search.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END xgb__learning_rate=0.1, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.920 total time=   7.4s
[CV 2/5] END xgb__learning_rate=0.1, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.918 total time=   7.3s
[CV 3/5] END xgb__learning_rate=0.1, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.916 total time=   6.8s
[CV 4/5] END xgb__learning_rate=0.1, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.918 total time=   5.5s
[CV 5/5] END xgb__learning_rate=0.1, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.922 total time=   4.5s
[CV 1/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.921 total time=   2.1s
[CV 2/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.918 total time=   2.1s
[CV 3/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.919 total time=   2.1s
[CV 4/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.922 total time=   2.1s
[CV 5/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.922 total time=   2.1s
[CV 1/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
1;, score=0.920 total time=   4.5s
[CV 2/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
1;, score=0.921 total time=   4.5s
[CV 3/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
1;, score=0.919 total time=   4.5s
[CV 4/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
1;, score=0.919 total time=   4.4s
[CV 5/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
1;, score=0.923 total time=   4.5s
[CV 1/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.922 total time=   2.1s
[CV 2/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.922 total time=   2.1s
[CV 3/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.922 total time=   2.1s
[CV 4/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.922 total time=   2.1s
[CV 5/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
1;, score=0.922 total time=   2.1s
[CV 1/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
1;, score=0.921 total time=   2.8s
[CV 2/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
1;, score=0.921 total time=   2.8s
[CV 3/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
1;, score=0.920 total time=   2.8s
[CV 4/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
1;, score=0.921 total time=   2.9s
[CV 5/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
1;, score=0.922 total time=   2.9s
[CV 1/5] END xgb__learning_rate=0.1, xgb__max_depth=3, xgb__min_child_weight=
1;, score=0.921 total time=   2.9s
[CV 2/5] END xgb__learning_rate=0.1, xgb__max_depth=3, xgb__min_child_weight=
1;, score=0.914 total time=   2.9s
[CV 3/5] END xgb__learning_rate=0.1, xgb__max_depth=3, xgb__min_child_weight=
1;, score=0.918 total time=   2.9s
[CV 4/5] END xgb__learning_rate=0.1, xgb__max_depth=3, xgb__min_child_weight=
1;, score=0.921 total time=   2.9s
[CV 5/5] END xgb__learning_rate=0.1, xgb__max_depth=3, xgb__min_child_weight=
```

```
1;, score=0.925 total time=   2.9s
[CV 1/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.922 total time=   2.1s
[CV 2/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.922 total time=   2.1s
[CV 3/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.922 total time=   2.1s
[CV 4/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.922 total time=   2.1s
[CV 5/5] END xgb__learning_rate=0.05, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.922 total time=   2.1s
[CV 1/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
3;, score=0.921 total time=   3.3s
[CV 2/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
3;, score=0.921 total time=   2.8s
[CV 3/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
3;, score=0.919 total time=   2.8s
[CV 4/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
3;, score=0.921 total time=   2.8s
[CV 5/5] END xgb__learning_rate=0.05, xgb__max_depth=3, xgb__min_child_weight=
3;, score=0.922 total time=   2.8s
[CV 1/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.922 total time=   2.8s
[CV 2/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.919 total time=   2.1s
[CV 3/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.918 total time=   2.1s
[CV 4/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.922 total time=   2.1s
[CV 5/5] END xgb__learning_rate=0.1, xgb__max_depth=2, xgb__min_child_weight=
3;, score=0.922 total time=   2.1s
[CV 1/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.920 total time=   4.4s
[CV 2/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.920 total time=   4.5s
[CV 3/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.920 total time=   4.5s
[CV 4/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.922 total time=   4.9s
[CV 5/5] END xgb__learning_rate=0.05, xgb__max_depth=5, xgb__min_child_weight=
3;, score=0.923 total time=   4.9s
```

Out[181]:
```
RandomizedSearchCV(cv=5,
                   estimator=Pipeline(steps=[('preparation',
                                              FeatureUnion(transformer_list=
[('num_pipeline',

Pipeline(steps=[('selector',

DataFrameSelector(attribute_names=['SK_ID_CURR',

'CNT_CHILDREN',

'AMT_INCOME_TOTAL',

'AMT_CREDIT',

'AMT_ANNUITY',

'AMT_GOODS_PRICE',

'REGION_POPULATION_RELATIVE',

'DAYS_BIRTH',

'DAYS_EMPLOYED',

'DAYS_REGISTRATION',

'DAYS...

'WEEKDAY_APPR_PROCESS_START',

'ORGANIZATION_TYPE',

'EMERGENCYSTATE_MODE'])),

('imputer',

SimpleImputer(strategy='most_frequent')),

('ohe',

OneHotEncoder(handle_unknown='ignore',

sparse=False))]))])),
                                             ('xgb', XGBClassifier())]),
                   param_distributions={'xgb__learning_rate': [0.05, 0.1],
                                        'xgb__max_depth': [2, 3, 5],
                                        'xgb__min_child_weight': [1, 3]},
                   scoring='accuracy', verbose=3)
```

In [182…
```
print("best params",random_search.best_params_)
```

```
best params {'xgb__min_child_weight': 1, 'xgb__max_depth': 2, 'xgb__learning_r
ate': 0.05}
```

In [183…
```
np.random.seed(42)
full_pipeline_with_predictor_1 = Pipeline([
        ("preparation", data_prep_pipeline),
        ("xgb", xgboost.XGBClassifier(min_child_weight=1,max_depth=2,learning_r
```

```
    ])
model_xgb = full_pipeline_with_predictor_1.fit(X_train,y_train)
```

In [185…
```
np.round(accuracy_score(y_train, model_xgb.predict(X_train)), 3)
```

Out[185]:
```
0.922
```

In [186…
```
roc_auc_score(y_train, model_xgb.predict_proba(X_train)[:, 1])
```

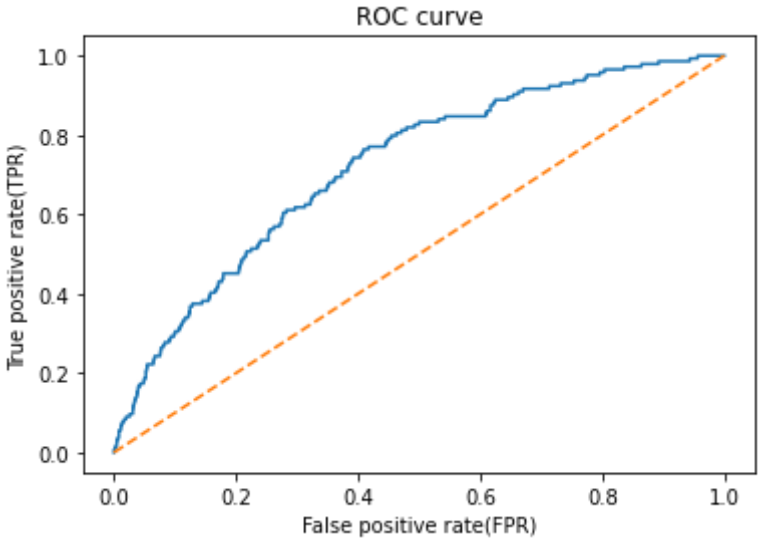
Out[186]:
```
0.8272814767472462
```

In [187…
```
from sklearn.metrics import f1_score
exp_name = "XG Boost Model"
expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
               [accuracy_score(y_train, model_xgb.predict(X_train)),
                accuracy_score(y_test, model_xgb.predict(X_test)),
                roc_auc_score(y_train, model_xgb.predict_proba(X_train)[:, 1]),
                roc_auc_score(y_test, model_xgb.predict_proba(X_test)[:, 1]),
                f1_score(y_train, model_xgb.predict(X_train), average='weighted
                f1_score(y_test, model_xgb.predict(X_test), average='weighted')
    4))
expLog
```

Out[187]:

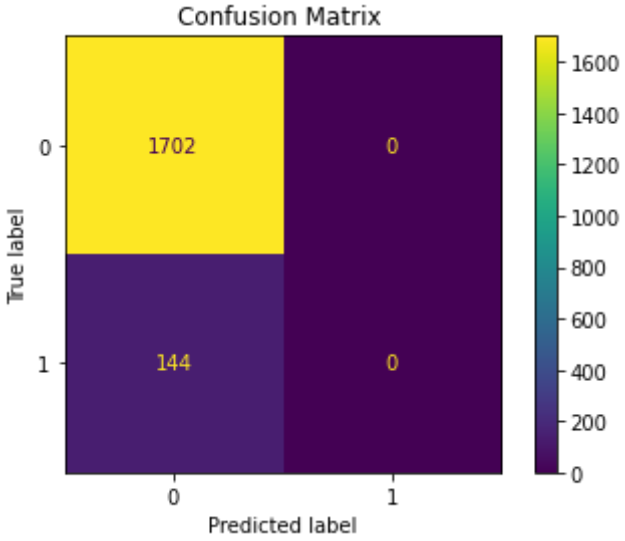
	exp_name	Train Acc	Test Acc	Train AUC	Test AUC	Train F1	Test F1
0	Logistic Regression	0.9240	0.9112	0.8465	0.6855	0.8972	0.8827
1	Random Forest Model	0.9226	0.9220	0.9975	0.6616	0.8860	0.8846
2	XG Boost Model	0.9222	0.9220	0.8273	0.7207	0.8849	0.8846

In [188…
```
from sklearn import metrics
fpr,tpr,thresh = metrics.roc_curve(y_test,model_xgb.predict_proba(X_test)[:,1])
plt.plot(fpr,tpr)
plt.plot(np.arange(0,1.1,0.1),np.arange(0,1.1,0.1),linestyle="dashed")
plt.xlabel("False positive rate(FPR)")
plt.ylabel("True positive rate(TPR)")
plt.title("ROC curve")
plt.show()
```



In [189…
```
from sklearn.metrics import plot_confusion_matrix
plt.clf()
plot_confusion_matrix(model_xgb,X_test,y_test)
plt.title('Confusion Matrix ')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Multilayer Perceptron

In [1]:
```
import pandas as pd
import numpy as np
import torch
from torch.utils.data import DataLoader,Dataset
import torch.optim as optim
from tqdm.notebook import tqdm
import torch.nn.functional as F

import torch.nn as nn
```

In [2]:
```
#Mount Google Drive

from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
drive.mount("/content/gdrive", force_remount=True).
```

In [3]:
```
test_data = pd.read_csv('/content/gdrive/MyDrive/AML/home-credit-default-risk/T
```

In [4]:
```
test_data
```

Out[4]:

	0	1	2	3	4	5	6	7
0	-0.486932	2.106840	-0.475346	0.439945	-0.161526	-0.123619	-0.12402	-0.124018
1	0.386101	-0.337285	0.397462	0.803713	-0.845412	-0.123619	-0.12402	-0.124018
2	0.095090	-0.337285	0.106526	-0.159637	0.714822	-0.123619	-0.12402	-0.124018
3	-0.777942	-0.337285	-0.766283	-0.503642	0.563570	-0.123619	-0.12402	-0.124018
4	-0.486932	-0.337285	-0.475346	0.939037	0.310593	-0.123619	-0.12402	-0.124018
...	...	...	...	...	...	...	...	...
4960	-0.632437	-0.337285	-0.620815	-0.445024	1.078997	-0.123619	-0.12402	-0.124018
4961	0.386101	-0.337285	0.397462	-0.184759	0.756322	-0.123619	-0.12402	-0.124018
4962	-0.195921	-0.337285	-0.184410	0.715283	-0.161526	-0.123619	-0.12402	-0.124018
4963	0.095090	-0.337285	0.106526	-0.487899	-1.001219	-0.123619	-0.12402	-0.124018
4964	-1.359964	-0.337285	-1.348155	0.986602	-0.127762	-0.123619	-0.12402	-0.124018

4965 rows × 167 columns

In [5]:
```
class HCDR_dataset(Dataset):
    def __init__(self, data_csv_path:str):
        self.data = pd.read_csv(data_csv_path)

    def __getitem__(self,idx):
        return torch.Tensor(self.data.iloc[idx,:-1]),self.data.iloc[idx,-1].ast
    def __len__(self):
        return len(self.data)
```

In [6]:
```
train_dataset = HCDR_dataset("/content/gdrive/MyDrive/AML/home-credit-default-r
test_dataset = HCDR_dataset("/content/gdrive/MyDrive/AML/home-credit-default-ri
valid_dataset = HCDR_dataset("/content/gdrive/MyDrive/AML/home-credit-default-r
```

In [7]:
```
from tensorflow.keras.callbacks import TensorBoard
from torch.utils.tensorboard import SummaryWriter
writer = SummaryWriter("/content/gdrive/MyDrive/AML/home-credit-default-risk/ru
```

In [8]:
```
class Linear_model(nn.Module):
    def __init__(self,n_hidden:int=3,n_neurons=[ 80,60,10],input_dim:int = 40,r
        super().__init__()

        self.n_hidden = n_hidden
        self.n_neurons  = n_neurons
        self.input_dim = input_dim
        self.n_classes = n_classes

        self.layer1 =  torch.nn.Linear(self.input_dim, self.n_neurons[0])
        self.layer_stack = nn.ModuleList([])
        for layers in range(len(self.n_neurons)):
            if layers+1>=len(self.n_neurons):
                self.layer_stack.append(torch.nn.Linear(self.n_neurons[layers],
                break
            self.layer_stack.append(torch.nn.Linear(self.n_neurons[layers], sel
```

```
    def forward(self,x):

        x = F.relu(self.layer1(x))
        for d in range(len(self.n_neurons[:-1])):
            x = F.relu(self.layer_stack[d](x))
        x = F.relu(self.layer_stack[-1](x))
        return x
```

In [9]:
```
len(valid_dataset[0][0])
```

Out[9]:
```
166
```

In [48]:
```
class MLP(nn.Module):
    def __init__(self,input_features=14,hidden1=20,hidden2=20,out_features=2):
        super().__init__()
        self.f_connected1 = nn.Linear(input_features,hidden1)
        self.f_connected2 = nn.Linear(hidden1,hidden2)
        self.out = nn.Linear(hidden2,out_features)
    def forward(self,x):
        x = F.leaky_relu(self.f_connected1(x))
        x = F.leaky_relu(self.f_connected2(x))
        x = self.out(x)
        return x
```

In [49]:
```
net = Linear_model(n_hidden=2,n_neurons=[20,20],input_dim=166)
```

In [50]:
```
net
```

Out[50]:
```
<bound method Module.parameters of MLP(
  (f_connected1): Linear(in_features=14, out_features=20, bias=True)
  (f_connected2): Linear(in_features=20, out_features=20, bias=True)
  (out): Linear(in_features=20, out_features=2, bias=True)
)>
```

In [13]:
```
from torchsummary import summary
```

In [14]:
```
summary(net.cuda(),(1,166))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                [-1, 1, 8]           1,336
            Linear-2                [-1, 1, 8]              72
            Linear-3                [-1, 1, 1]               9
================================================================
Total params: 1,417
Trainable params: 1,417
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.01
Estimated Total Size (MB): 0.01
----------------------------------------------------------------
```

In [15]:
```
train_loader = DataLoader(train_dataset,batch_size=2048,shuffle=True)
valid_loader = DataLoader(valid_dataset,batch_size=2048,shuffle=False)
test_loader = DataLoader(test_dataset,batch_size=2048,shuffle=False)
```

In [17]:
```
criterion = nn.BCEWithLogitsLoss()
opt = optim.Adam(net.parameters(),lr=0.001)
loss_history = []
acc_history = []

def train_epoch(epoch, clf, criterion, opt, train_loader, num_of_epochs):
    clf.train() # set model in training mode (need this because of dropout)
    running_loss = 0.0

    # dataset API gives us pythonic batching
    for batch_id, data in enumerate(train_loader):
        inputs, target = data[0].to("cuda"), data[1].to("cuda")
        opt.zero_grad()
        preds = clf(inputs) #prediction over the input data
        loss = criterion(preds.double(), target.double())
        loss.backward()
        loss_history.append(loss.item())
        opt.step()
        running_loss += loss.item()
        if batch_id % 100 == 0:    # print every 100 mini-batches
            print(f"Epoch {epoch} of {num_of_epochs}, batch {batch_id+1}, batch
            running_loss = 0.0

    writer.add_scalar('Training loss', running_loss, epoch+1)

    return clf


def evaluate_model(epoch, clf, criterion, opt, data_loader, tag = "Test"):
    probas = []
    clf.eval() # set model in inference mode (need this because of dropout)
    correct = 0
    count = 0
    overall_loss = 0.0

    for i,data in enumerate(data_loader):
        inputs, targets = data[0].to("cuda"), data[1].to("cuda")
        outputs = clf(inputs)
        predicted_classes = torch.sigmoid(outputs.data)
        probas.append(predicted_classes)
        predicted_classes = (predicted_classes>0.5)*1
        correct += (predicted_classes.flatten().flatten() == targets.flatten())


        loss = criterion(outputs.double(), targets.double())
        loss_this_iter = loss.cpu().detach().numpy()
        overall_loss += (loss_this_iter)
        count += inputs.size(0)

    overall_loss /= len(data_loader.dataset)
    writer.add_scalar(f'{tag} loss', overall_loss, epoch+1)

    accuracy = 100. * correct / count
    acc_history.append(accuracy)
```

```
    print(f"{tag} {epoch} set: Average loss: {overall_loss:.6f}, Accuracy: {cor
    return accuracy,probas
```

In [18]:

```
num_of_epochs = 30
for epoch in range(num_of_epochs):
    print("Epoch %d" % epoch)
    clf = train_epoch(epoch, net, criterion, opt, train_loader, num_of_epochs)
    evaluate_model(epoch, net, criterion, opt, valid_loader, tag = "Validation"
print("-"*50)
_,pred_proba = evaluate_model(epoch, net, criterion, opt, test_loader, tag="Tes


writer.flush()
writer.close()
```

```
Epoch 0
Epoch 0 of 30, batch 1, batch loss: 0.706397
Validation 0 set: Average loss: 0.000467, Accuracy: 2235/4469 (50%)
Epoch 1
Epoch 1 of 30, batch 1, batch loss: 0.694368
Validation 1 set: Average loss: 0.000456, Accuracy: 2504/4469 (56%)
Epoch 2
Epoch 2 of 30, batch 1, batch loss: 0.673597
Validation 2 set: Average loss: 0.000451, Accuracy: 2774/4469 (62%)
Epoch 3
Epoch 3 of 30, batch 1, batch loss: 0.66576
Validation 3 set: Average loss: 0.000447, Accuracy: 2844/4469 (64%)
Epoch 4
Epoch 4 of 30, batch 1, batch loss: 0.660621
Validation 4 set: Average loss: 0.000444, Accuracy: 2888/4469 (65%)
Epoch 5
Epoch 5 of 30, batch 1, batch loss: 0.659779
Validation 5 set: Average loss: 0.000442, Accuracy: 2918/4469 (65%)
Epoch 6
Epoch 6 of 30, batch 1, batch loss: 0.647244
Validation 6 set: Average loss: 0.000439, Accuracy: 2953/4469 (66%)
Epoch 7
Epoch 7 of 30, batch 1, batch loss: 0.652207
Validation 7 set: Average loss: 0.000438, Accuracy: 2961/4469 (66%)
Epoch 8
Epoch 8 of 30, batch 1, batch loss: 0.652405
Validation 8 set: Average loss: 0.000437, Accuracy: 2975/4469 (67%)
Epoch 9
Epoch 9 of 30, batch 1, batch loss: 0.642659
Validation 9 set: Average loss: 0.000437, Accuracy: 3003/4469 (67%)
Epoch 10
Epoch 10 of 30, batch 1, batch loss: 0.645902
Validation 10 set: Average loss: 0.000436, Accuracy: 3009/4469 (67%)
Epoch 11
Epoch 11 of 30, batch 1, batch loss: 0.651531
Validation 11 set: Average loss: 0.000436, Accuracy: 2997/4469 (67%)
Epoch 12
Epoch 12 of 30, batch 1, batch loss: 0.642409
Validation 12 set: Average loss: 0.000436, Accuracy: 3005/4469 (67%)
Epoch 13
Epoch 13 of 30, batch 1, batch loss: 0.637637
Validation 13 set: Average loss: 0.000436, Accuracy: 3001/4469 (67%)
Epoch 14
Epoch 14 of 30, batch 1, batch loss: 0.6454
Validation 14 set: Average loss: 0.000436, Accuracy: 3006/4469 (67%)
Epoch 15
Epoch 15 of 30, batch 1, batch loss: 0.640433
Validation 15 set: Average loss: 0.000436, Accuracy: 3012/4469 (67%)
Epoch 16
Epoch 16 of 30, batch 1, batch loss: 0.636801
Validation 16 set: Average loss: 0.000436, Accuracy: 2991/4469 (67%)
Epoch 17
Epoch 17 of 30, batch 1, batch loss: 0.649407
Validation 17 set: Average loss: 0.000436, Accuracy: 3005/4469 (67%)
Epoch 18
Epoch 18 of 30, batch 1, batch loss: 0.641722
Validation 18 set: Average loss: 0.000436, Accuracy: 2999/4469 (67%)
Epoch 19
Epoch 19 of 30, batch 1, batch loss: 0.646735
Validation 19 set: Average loss: 0.000436, Accuracy: 2987/4469 (67%)
```

```
Epoch 20
Epoch 20 of 30, batch 1, batch loss: 0.64471
Validation 20 set: Average loss: 0.000436, Accuracy: 2996/4469 (67%)
Epoch 21
Epoch 21 of 30, batch 1, batch loss: 0.631068
Validation 21 set: Average loss: 0.000436, Accuracy: 2996/4469 (67%)
Epoch 22
Epoch 22 of 30, batch 1, batch loss: 0.651334
Validation 22 set: Average loss: 0.000436, Accuracy: 3004/4469 (67%)
Epoch 23
Epoch 23 of 30, batch 1, batch loss: 0.65699
Validation 23 set: Average loss: 0.000436, Accuracy: 2996/4469 (67%)
Epoch 24
Epoch 24 of 30, batch 1, batch loss: 0.632455
Validation 24 set: Average loss: 0.000436, Accuracy: 3003/4469 (67%)
Epoch 25
Epoch 25 of 30, batch 1, batch loss: 0.64181
Validation 25 set: Average loss: 0.000436, Accuracy: 2994/4469 (67%)
Epoch 26
Epoch 26 of 30, batch 1, batch loss: 0.637091
Validation 26 set: Average loss: 0.000436, Accuracy: 2993/4469 (67%)
Epoch 27
Epoch 27 of 30, batch 1, batch loss: 0.649141
Validation 27 set: Average loss: 0.000436, Accuracy: 2989/4469 (67%)
Epoch 28
Epoch 28 of 30, batch 1, batch loss: 0.639203
Validation 28 set: Average loss: 0.000436, Accuracy: 2994/4469 (67%)
Epoch 29
Epoch 29 of 30, batch 1, batch loss: 0.646501
Validation 29 set: Average loss: 0.000436, Accuracy: 2993/4469 (67%)
-------------------------------------------------
Test 29 set: Average loss: 0.000392, Accuracy: 3373/4965 (68%)
```

In [18]:

In [19]:
```
proba_Scores = []
for val in pred_proba:
  for x in val:
    proba_Scores.append(x.cpu().numpy()[0])
```

In [20]:
```
import sklearn.metrics as metrics
```

In [21]:
```
fpr, tpr, thresholds = metrics.roc_curve(y_true = test_data['Target'], y_score
auroc = metrics.auc(fpr, tpr)
```

In [22]:
```
print(auroc)
```

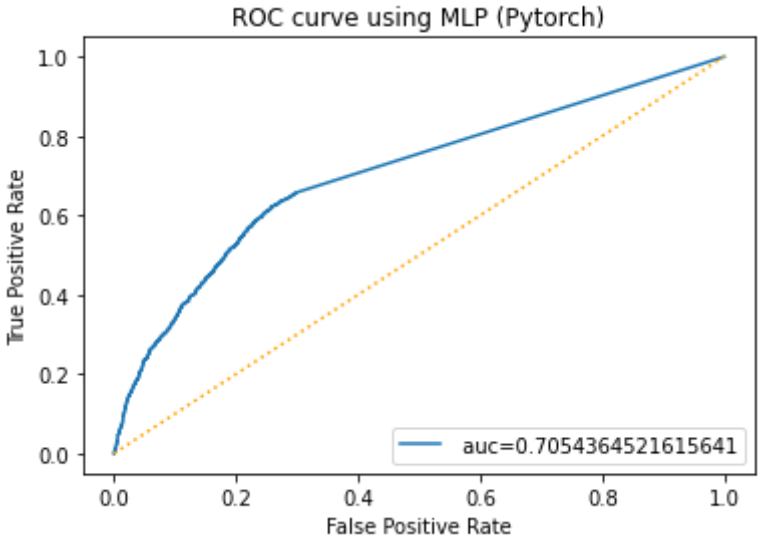
```
0.7054364521615641
```

AUROC plotting

In [23]:
```
import matplotlib.pyplot as plt
```

In [24]:
```
auc = metrics.roc_auc_score( test_data['Target'], proba_Scores)
plt.plot(fpr,tpr,label=" auc="+str(auc))
plt.plot(np.arange(0,2),np.arange(0,2),linestyle="dotted",color="orange")
plt.xlabel("False Positive Rate")
```

```
plt.ylabel("True Positive Rate")
plt.title("ROC curve using MLP (Pytorch)")
plt.legend(loc=4)
plt.show()
```



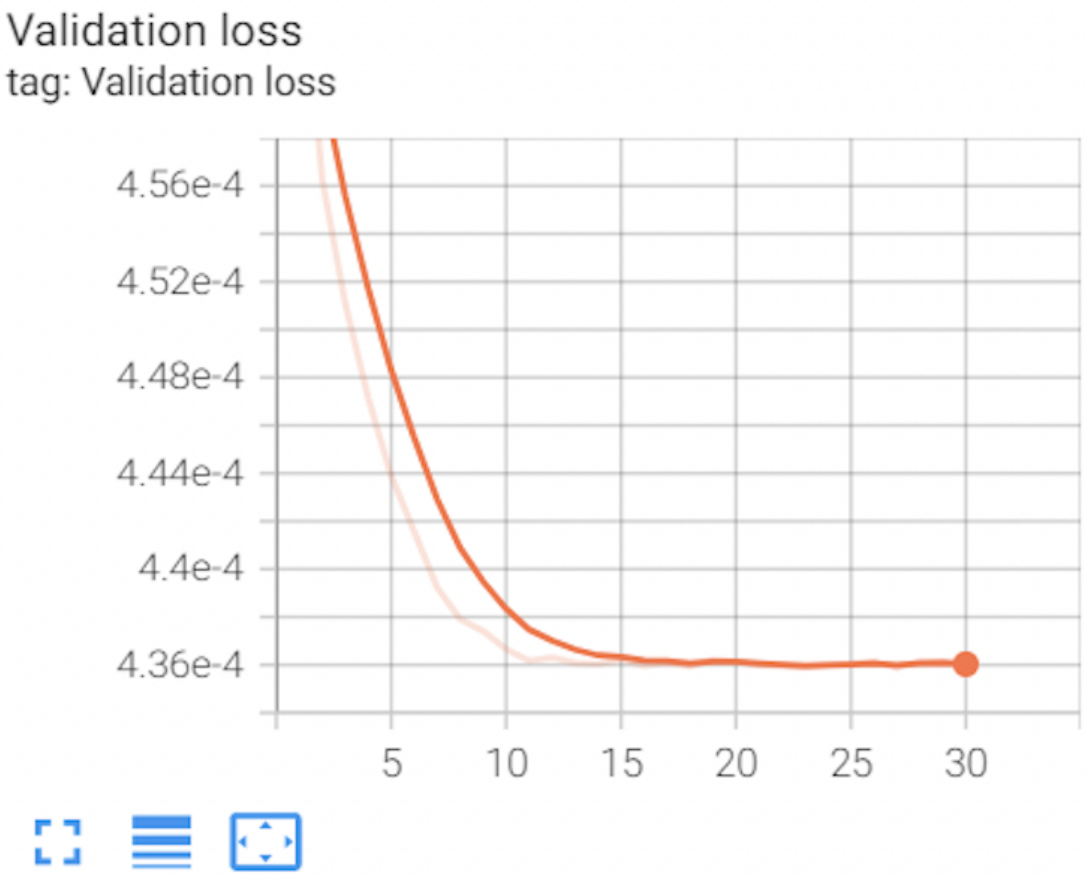
In [25]:
```
%load_ext tensorboard
```

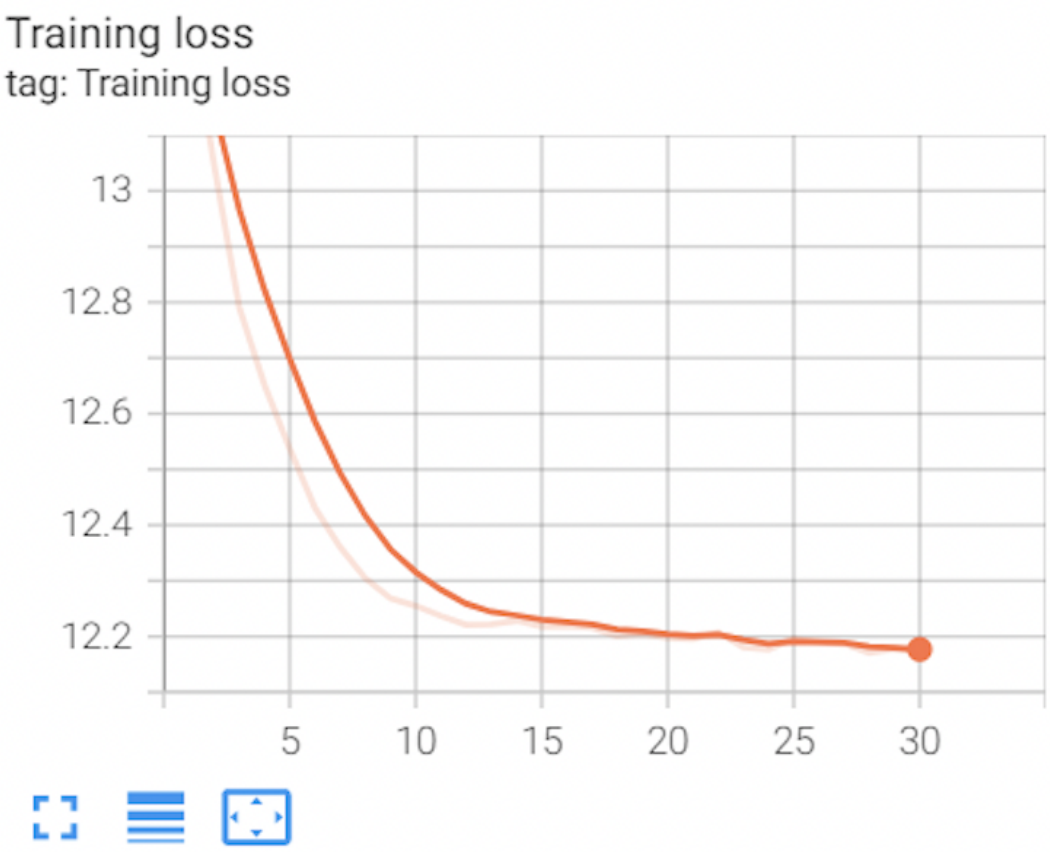
In [26]:
```
%matplotlib inline
```

In [27]:
```
%tensorboard --logdir /content/gdrive/MyDrive/AML/home-credit-default-risk/ Viz
```

```
Reusing TensorBoard on port 6007 (pid 4531), started 0:23:07 ago. (Use '!kill
4531' to kill it.)
```

Tensor board visualizations


Validation loss
tag: Validation loss
4.56e-4
4.52e-4
4.48e-4
4.44e-4
4.4e-4
4.36e-4
5
10
15
20
25
30



Submission File Prep

For each SK_ID_CURR in the test set, you must predict a probability for the TARGET variable. The file should contain a header and have the following format:

```
SK_ID_CURR,TARGET
100001,0.1
100005,0.9
100013,0.2
etc.
```

Kaggle test preprocessing

Kaggle submission via the command line API

In [182…
```
X_kaggle_test.shape
```

Out[182]:
```
(48744, 80)
```

In [184…
```
app_train.shape
```

Out[184]:
```
(307511, 195)
```

In [185…
```
X_kaggle_test = X_kaggle_test.merge(pa_agg, how='left', on='SK_ID_CURR')
```

In [186…
```
X_kaggle_test = X_kaggle_test.merge(bureau_agg, how='left', on="SK_ID_CURR")
```

In [187…
```
X_kaggle_test['CREDIT_INCOME_PCT'] = X_kaggle_test['AMT_CREDIT'] / X_kaggle_tes
X_kaggle_test['CREDIT_TERM'] = X_kaggle_test['AMT_ANNUITY'] / X_kaggle_test['AM
```

In [188…
```
X_kaggle_test[pa_agg.columns] = X_kaggle_test[pa_agg.columns].fillna(0)
```

In [189…
```
X_kaggle_test[bureau_agg.columns] = X_kaggle_test[bureau_agg.columns].fillna(0)
```

In [190…
```
X_kaggle_test.shape
```

Out[190]:
```
(48744, 194)
```

In [191…
```
app_train.shape
```

Out[191]:
```
(307511, 195)
```

In [206…
```
test_class_scores = model_rf.predict(X_kaggle_test)
```

In [207…
```
submit_df = pd.DataFrame({'SK_ID_CURR' : list(X_kaggle_test['SK_ID_CURR']), 'TA
```

In [208…
```
submit_df.head()
```

Out[208]:

	SK_ID_CURR	TARGET
0	100001	0
1	100005	0
2	100013	0
3	100028	0
4	100038	0

In [209…
```
submit_df.to_csv("submission.csv",index=False)
```

In [210…
```
!pwd
```

```
/content
```

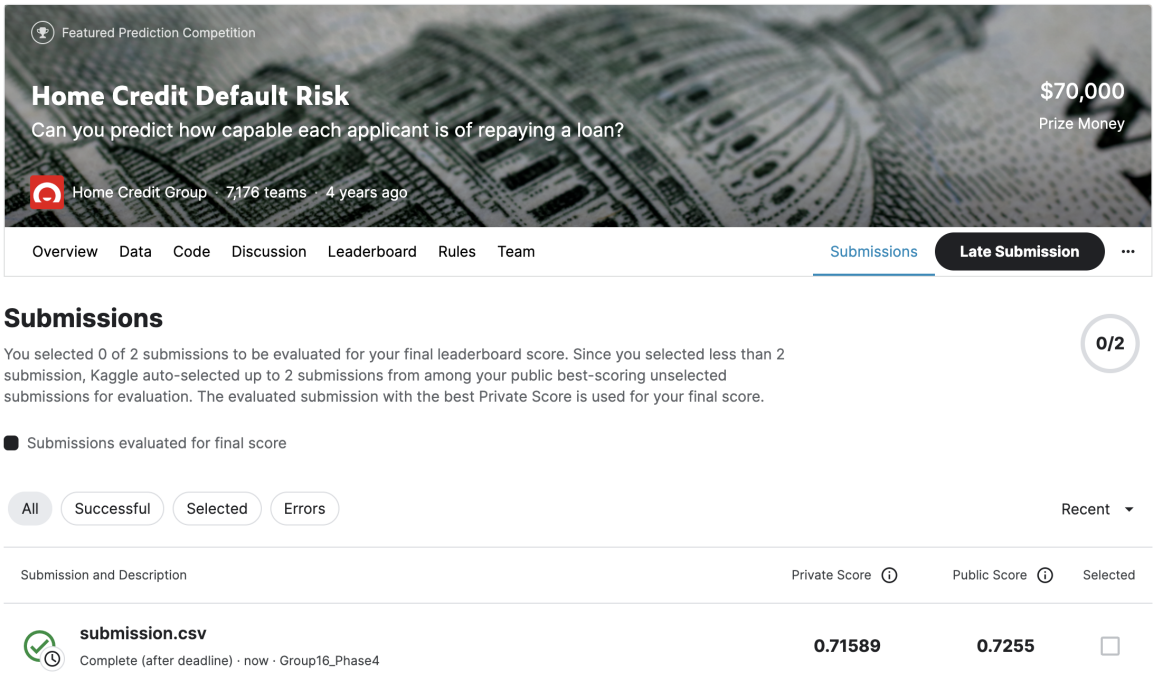
In [211…
```
from google.colab import files
files.download("submission.csv")
```

In [7]:
```
# ! kaggle competitions submit -c home-credit-default-risk -f submission.csv -m
#uploaded directly via web
! kaggle competitions submit -c home-credit-default-risk -f submission.csv -m '
```

```
100%|████████████████████████████████████████| 1.26M/1.26M [00:01<00:00, 918kB/
s]
Successfully submitted to Home Credit Default Risk
```

report submission

Click on this link



Write-up

For this phase of the project, you will need to submit a write-up summarizing the work you did. The write-up form is available on Canvas (Modules-> Module 12.1 - Course Project - Home Credit Default Risk (HCDR)-> FP Phase 2 (HCDR) : write-up form). It has the following sections:

Abstract

Home Credit is a prominent developing market consumer financing specialist that has developed a platform that manages its core technology, product, and funding activities keeping in mind the local market needs. Their target market is underserved borrowers in the blue-collar and junior white-collar segments who have a steady source of income from their jobs or micro-businesses but are less likely to get loans from banks and other traditional lenders. It is vital for Financial Organizations to observe whether their loan applicants will be able to repay their loans. There are multiple parameters that need to be considered for optimally predicting if the client will be defaulting. The parameters that can be used are occupation, credit history, age, location, credit card usage, cash balance and others. We will thoroughly visualize and study these parameters before implementing them. Our goal is to make use of all the parameters to predict with the best possible efficacy which will aid financial organizations to make better decisions for their loan applicants.

To perform the Home Credit Default Risk project, we have extracted the data from the Kaggle competition 'Home Credit Risk Analysis' and performed Exploratory Data Analysis to understand and explore the data. Various visualizations were performed on most of the input features to the 'Target' variable to find the people at maximum risk. In the second phase of the project, the models built were overfitting. So, we have remodeled the old feature engineering on all the tables and added features such as AMT_DRAWING_ratio, DAYS_INSTALLMENTS_diff, and AMT_ANNUITY_ratio. Data leakage was taken into consideration throughout the project to retain the efficiency of the project. Modeled a neural network model and implemented Multi_layer Perceptron with the help of Pytorch. A Tensor board was used to visualize and track the training and validation loss of the models. The Machine Learning Classifiers such as Logistic Regression, Random Forest, and XGBoost have performed better. The accuracies of our models improved compared to phase 3. Logistic Regression achieved a test accuracy of 91.1% and a test ROC_AUC score of 0.6855, whereas XGBoost has a better ROC_AUC score, which is 0.7207. We have also implemented Multi-Layer Perceptron and achieved a test accuracy of 68% and a ROC_AUC score of 0.705. A Kaggle submission was made using the best performing model XGBoost and obtained a score of 0.7255.

For Kaggle submission, we obtained a public score of 0.7158 and a private score of 0.7255

Introduction

Data Description

Data Description

In this project, seven different datasets are utilized.

● Application_{train|test}.csv:

This is the primary dataset, it consists of 307,511 observations and 122 different variables which provide data on all the applicants. In this dataset, each applicant has a row with the Target variables 0 and

1. Here, 0 indicates that the loan was repaid and 1 indicates that the loan was not paid. All the data points in this dataset are static for all applicants.

● Bureau.csv:

Here, the data consists of the customer's previous credits from different financial companies that were reported to the Credit Bureau. Every previous credit has its own row, unlike application data where one loan can have multiple previous credits.

● Bureau_Balance.csv:

This data has monthly balances of previous credit. Single previous credit can have multiple rows, one for every month in the period of credit length.

● Credit_card_balance.csv:

All the monthly balances of previous credits taken by an individual with the Home Credit. A single credit card can have multiple rows with each row consisting of every month's balance.

● Previous_application.csv:

Data consists of each person's previous loan application at Home Credit. Current loan applicants can have multiple previous loan applications.

● POS_CASH_balance.csv:

In this dataset, monthly data of the previous point of sale or cash loan clients had with Home Credit are present. A single previous loan can have multiple rows with each row representing one-month data of the previous point of sale or cash loan.

● Installments_payments.csv:

The previous loan payment data is present with Home Credit. The data has both information on every missed and made payment.

Tasks to be handled

- Building a Multi Layer Perceptron using PyTorch
- Using Tensorboard to visualize the results of training in real-time.
- We have increased number of features selected from 14(in Phase 3) to 50(in phase 4) and ran all the ML Classifiers.

Neural Networks

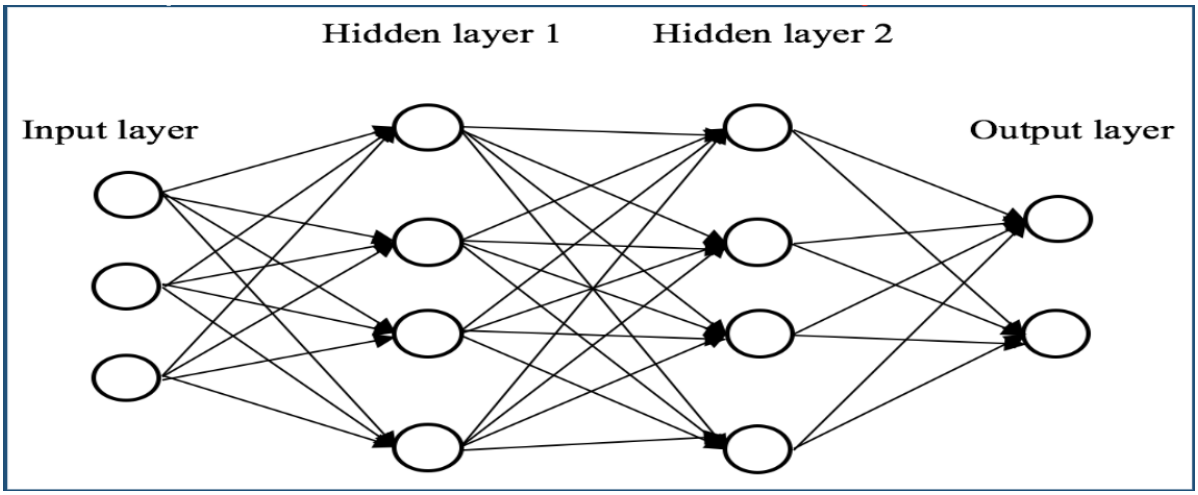
We have built Multilayer perceptron using 3 hidden layers and {80,60,10} neurons with 40 input dimensions in the phase 4 of HCDR project. And also we implemented Multilayer perceptron using 2 hidden layers and 20 neurons with 166 input dimensions. We have acquired the AUC score - 0.705 and accuracy - 68%

Neural network architecture in string form for 3 hidden layers is

- 80-Relu-BCE-60-Relu-BCE-10-Relu-BCE-1.

Neural network architecture in string form for 2 hidden layers is

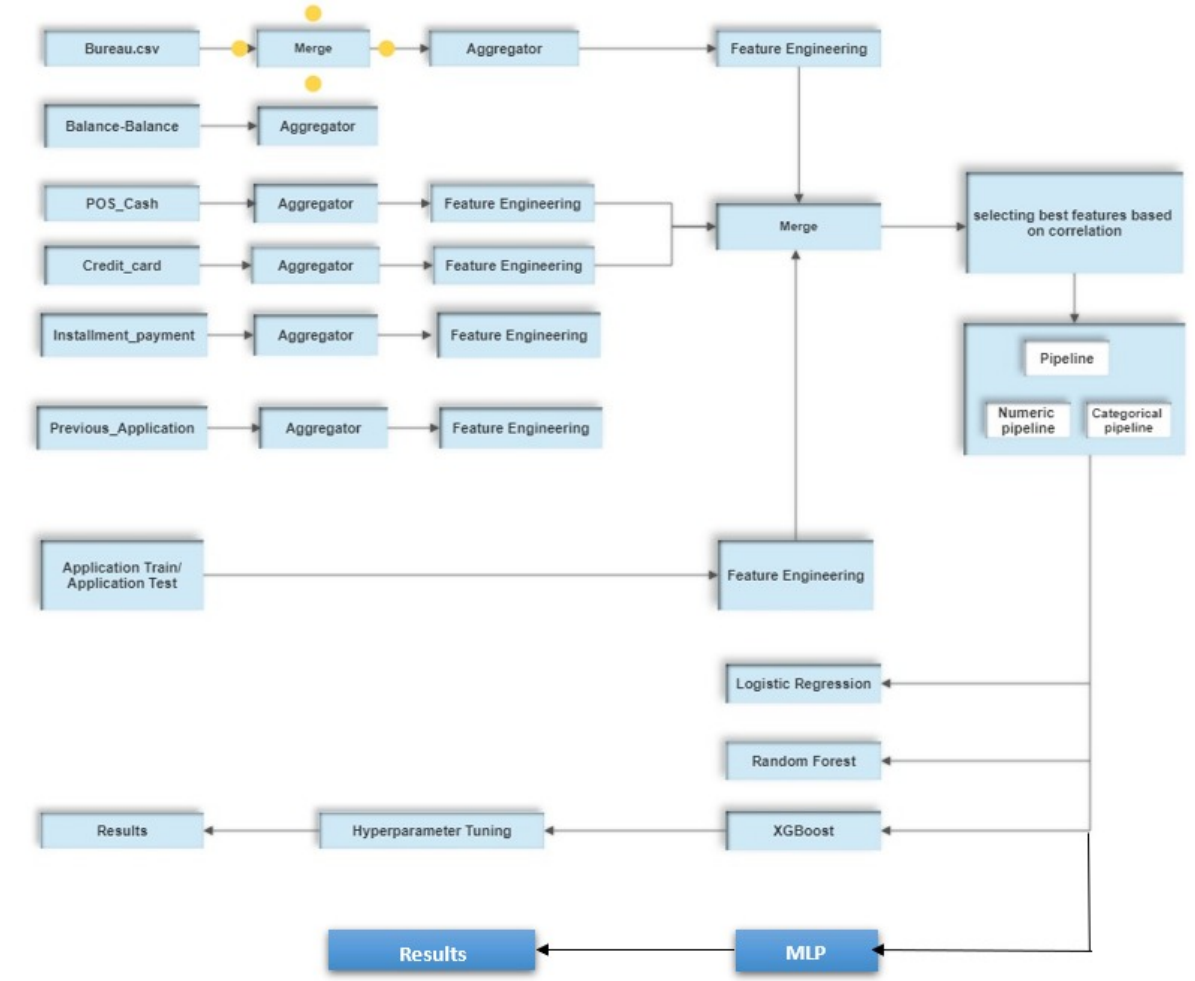
- 20-Relu-BCE-20-Relu-BCE-1.



Pipelines

Families of input features: count of numerical features: 106(int64, float64) count of categorical features: 16(object) The total number of input features: 122 input features. We have trained four models:

1. Logistic Classifier
2. RandomForestClassifier
3. XGBoost
4. Multilayer Perceptron



In [ ]:
```
from sklearn.preprocessing import OrdinalEncoder

# Identify the numeric features we wish to consider.
num_attribs = imp_features

 # [
 #    'AMT_INCOME_TOTAL',  'AMT_CREDIT','DAYS_EMPLOYED','DAYS_BIRTH','EXT_SOURC
 #    'EXT_SOURCE_2','EXT_SOURCE_3']

num_pipeline = Pipeline([
        ('selector', DataFrameSelector(num_attribs)),
        ('imputer', SimpleImputer(strategy='mean')),
        ('std_scaler', StandardScaler()),
    ])
# Identify the categorical features we wish to consider.
cat_attribs = cat_cols2


# ['CODE_GENDER', 'FLAG_OWN_REALTY','FLAG_OWN_CAR','NAME_CONTRACT_TYPE',
#                 'NAME_EDUCATION_TYPE','OCCUPATION_TYPE','NAME_INCOME_TYPE']

# Notice handle_unknown="ignore" in OHE which ignore values from the validatio
# do NOT occur in the training set
cat_pipeline = Pipeline([
        ('selector', DataFrameSelector(cat_attribs)),
        #('imputer', SimpleImputer(strategy='most_frequent')),
        ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
```

```
        ('ohe', OneHotEncoder(sparse=False, handle_unknown="ignore"))
    ])

data_prep_pipeline = FeatureUnion(transformer_list=[
        ("num_pipeline", num_pipeline),
        ("cat_pipeline", cat_pipeline),
    ])
```

In [ ]:
```
%%time
np.random.seed(42)
full_pipeline_with_predictor = Pipeline([
        ("preparation", data_prep_pipeline),
        ("linear", LogisticRegression())
    ])
model_lr = full_pipeline_with_predictor.fit(X_train, y_train)
```

In [ ]:
```
%%time
np.random.seed(42)
full_pipeline_with_predictor_1 = Pipeline([
        ("preparation", data_prep_pipeline),
        ('rf', RandomForestClassifier(n_estimators=100,max_features=2,max_depth
    ])
model_rf = full_pipeline_with_predictor_1.fit(X_train,y_train)
```

In [ ]:
```
%%time
np.random.seed(42)
full_pipeline_with_predictor_1 = Pipeline([
        ("preparation", data_prep_pipeline),
        ("xgb", xgboost.XGBClassifier(min_child_weight=1,max_depth=2,learning_r
    ])
model_xgb = full_pipeline_with_predictor_1.fit(X_train,y_train)
```

Data Leakage

Data leakage (or leakage) happens when your training data contains information about the target, but similar data will not be available when the model is used for prediction. We have split the dataset into training and test data. We are removing missing values in our data by replacing them with mean. We scaled the data using StandardScaler. Because of these factors, there will not be any significant data leakage in our modeled pipelines.

Cardinal sins avoided:

To avoid overfitting we have splitted our data into train and test. The test dataset is only available when we train the model on the train set and once the evaluation is done. We can observe that the accuracy for the train and test are very close which avoids the overfitting the model.

Loss Functions

Log Loss

Log loss is indictive of how close is the prediction probability is to the corresponding value/true value. The more the predicted probability diverges from the actual value, the higher is the log loss value.

In [12]:
```
import latexify
@latexify.function(use_math_symbols=True)
def logLoss():
    return (-1 / N) * Sigma(_i**N) (y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y
logLoss
```

Out[12]:
$$ \displaystyle \mathrm{logLoss}() = \frac{-{1}}{N} \mathrm{\mathrm{{\Sigma}}\left(_i^{N}\right)}\left(y_i \log{\left({\mathrm{p}\left(y_i\right)}\right)} + \left({1} - y_i \right) \log{\left({{1} - \mathrm{p}\left(y_i\right)}\right)}\right) $$

Gini Impurity

Gini Impurity is a measure of variance across the different classes

In [13]:
```
import latexify
@latexify.function(use_math_symbols=True)
def Gini(node):
    return Sigma(_i**c) (p_i * (1-p_i))
Gini
```

Out[13]:
$$ \displaystyle \mathrm{Gini}(node) = \mathrm{\mathrm{{\Sigma}}\left(_i^{c}\right)}\left(p_i \left({1} - p_i \right)\right) $$

Experimental results

In [187…
```
expLog
```

Out[187]:

	exp_name	Train Acc	Test Acc	Train AUC	Test AUC	Train F1	Test F1
0	Logistic Regression	0.9240	0.9112	0.8465	0.6855	0.8972	0.8827
1	Random Forest Model	0.9226	0.9220	0.9975	0.6616	0.8860	0.8846
2	XG Boost Model	0.9222	0.9220	0.8273	0.7207	0.8849	0.8846

Discussion of Results

The models were overfitting in phase 3 and had an accuracy of 1. We improved this, and the models performed better than in the previous phase after increasing the number of features to 50. Following are the results we have got in Phase 4:

Logistic Regression

- Test accuracy - 91%
- Test ROC_AUC - 0.685

Random Forest

- Test accuracy - 92%
- Test ROC_AUC - 0.661

XG Boost

- Test accuracy - 92%
- Test ROC_AUC- 0.720

MLP

- Test accuracy - 68%
- Test ROC_AUC- 0.705

Overall, XGBoost performed better among all the models.

Conclusion

After exploring new features from EDA in phase 2. We have focused much on improving the test accuracy, additional Feature Engineering, HyperParameter Tuning, Feature selection, analysis of Feature importance, and other ensemble methods in this phase.

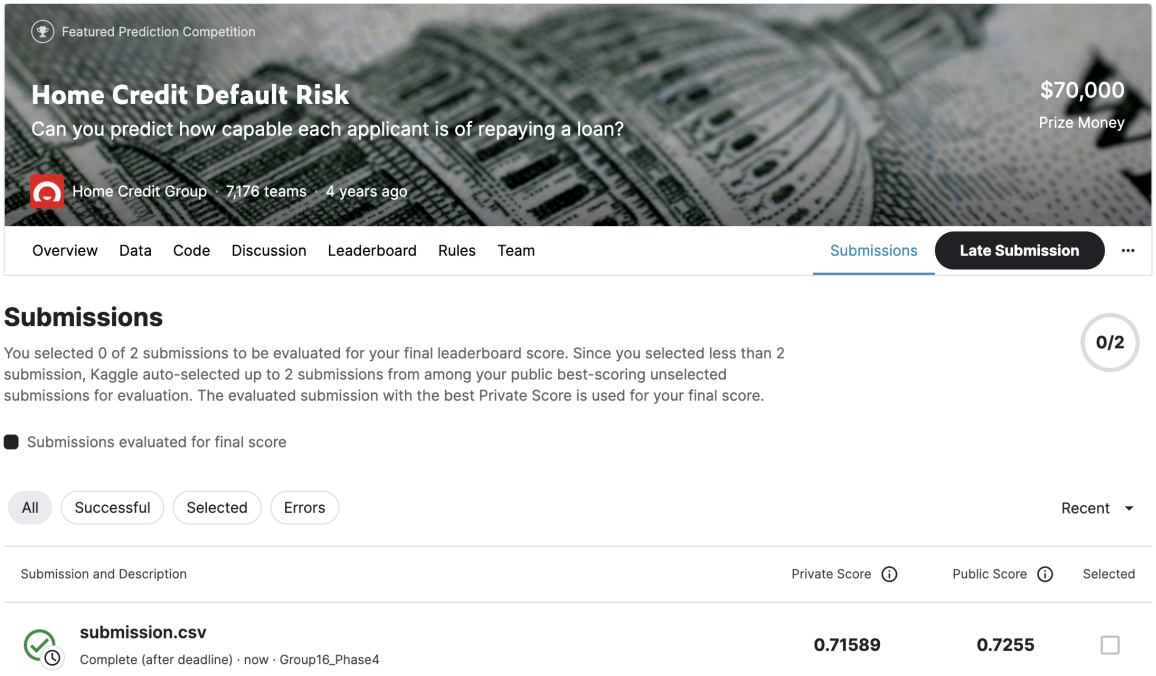
In Phase 3, the data was trained and tested with the help of ML Classifiers Logistic Regression, Random Forest, and XGBoost. Accuracies and areas under the ROC_AUC curve were found and compared. We have improved our Logistic Regression and Random Forest compared to our phase 2 submission. Logistic Regression has achieved an accuracy of 0.808. We have achieved this by doing feature engineering on secondary and tertiary tables. Later we chose the 50 best-correlated columns and dropped all null values. We then performed hyper-parameter tuning. XGBoost gave an accuracy of 0.996.

In the final phase: This project's main goal is to develop a Machine Learning model that can predict whether or not a loan applicant will be able to repay the loan. Without any statistical analysis, many deserving applicants with no credit history or default history get approved. The HCDR dataset is used to train the machine learning model. Based on the history of comparable applicants in the past, it will be able to estimate whether or not an applicant would be able to repay their loan. This model would help in the screening of applications by providing statistical support resulting from numerous aspects taken into account.

In the previous phase of the project the models were overfitting the data and hence got an accuracy score of 1, to overcome this we added new features using the pipelines, and tables were merged considering inherent hierarchy. A neural network model, Multi-layer

Perceptron was implemented using PyTorch in the final phase, and MLP was modeled with two hidden layers, twenty neurons, and 166 input dimensions. This resulted in an accuracy of 68% with an AUC score of 0.705. Out of all models implemented, XGboost gave the best results resulting in an accuracy of 92.2 and an AUC score of 0.720. We have also made a Kaggle submission and obtained a score of 0.7255.

Kaggle Submission



References

Some of the material in this notebook has been adopted from here

In [ ]:

In [ ]:

TODO: Predicting Loan Repayment with Automated Feature Engineering in Featuretools

Read the following:

- feature engineering via Featuretools library:
  - https://github.com/Featuretools/predict-loan-repayment/blob/master/Automated%20Loan%20Repayment.ipynb

- https://www.analyticsvidhya.com/blog/2018/08/guide-automated-feature-engineering-featuretools-python/
- feature engineering paper: https://dai.lids.mit.edu/wp-content/uploads/2017/10/DSAA_DSM_2015.pdf
- https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/

In [ ]: