Credit Risk Analysis_Solution

October 27, 2024

[102]:
```
# Import the Dataset

# Import necessary libraries for data analysis and visualization
import pandas as pd  # For data manipulation and analysis
import numpy as np   # For numerical computation
import matplotlib.pyplot as plt  # For plotting
import seaborn as sns
import xlrd # For advanced statistical visualizations
sns.set(color_codes=True)  # Set seaborn color palette to default
```

[103]:
```
import warnings

# Filter out warnings to ignore them
warnings.filterwarnings('ignore')
```

[104]:
```
# Reading the Dataset

# Read the CSV file 'application_data.csv' into a DataFrame
application_train = pd.read_csv('application_data.csv')
```

[105]:
```
# Task-1 - Exploring the Dataset

# Display the first few rows of the DataFrame to inspect its structure and␣
 ↪contents
application_train.head()
```

[105]:

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

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

```
4               Y             0          121500.0    513000.0      21865.5

   …  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20 FLAG_DOCUMENT_21  \
0  …                 0                0                 0                 0
1  …                 0                0                 0                 0
2  …                 0                0                 0                 0
3  …                 0                0                 0                 0
4  …                 0                0                 0                 0

  AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
0                        0.0                       0.0
1                        0.0                       0.0
2                        0.0                       0.0
3                        NaN                       NaN
4                        0.0                       0.0

   AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                         0.0                       0.0
1                         0.0                       0.0
2                         0.0                       0.0
3                         NaN                       NaN
4                         0.0                       0.0

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                        1.0
1                        0.0                        0.0
2                        0.0                        0.0
3                        NaN                        NaN
4                        0.0                        0.0

[5 rows x 122 columns]
```

[106]:
```
# Get the dimensions of the DataFrame (number of rows and columns)
application_train.shape
```

[106]:
```
(307511, 122)
```

[107]:
```
# Get concise summary information about the DataFrame, including column data␣
 ↪types and non-null counts
application_train.info()
```

```
<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
```

[108]:
```
application_train.iloc[:,:100].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 100 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
dtypes: float64(59), int64(25), object(16)
memory usage: 234.6+ MB
```

[109]:
```
application_train.iloc[:,100:122].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 22 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   FLAG_DOCUMENT_6             307511 non-null  int64
 1   FLAG_DOCUMENT_7             307511 non-null  int64
 2   FLAG_DOCUMENT_8             307511 non-null  int64
 3   FLAG_DOCUMENT_9             307511 non-null  int64
 4   FLAG_DOCUMENT_10            307511 non-null  int64
 5   FLAG_DOCUMENT_11            307511 non-null  int64
 6   FLAG_DOCUMENT_12            307511 non-null  int64
 7   FLAG_DOCUMENT_13            307511 non-null  int64
 8   FLAG_DOCUMENT_14            307511 non-null  int64
 9   FLAG_DOCUMENT_15            307511 non-null  int64
 10  FLAG_DOCUMENT_16            307511 non-null  int64
 11  FLAG_DOCUMENT_17            307511 non-null  int64
 12  FLAG_DOCUMENT_18            307511 non-null  int64
 13  FLAG_DOCUMENT_19            307511 non-null  int64
 14  FLAG_DOCUMENT_20            307511 non-null  int64
 15  FLAG_DOCUMENT_21            307511 non-null  int64
 16  AMT_REQ_CREDIT_BUREAU_HOUR  265992 non-null  float64
 17  AMT_REQ_CREDIT_BUREAU_DAY   265992 non-null  float64
 18  AMT_REQ_CREDIT_BUREAU_WEEK  265992 non-null  float64
 19  AMT_REQ_CREDIT_BUREAU_MON   265992 non-null  float64
 20  AMT_REQ_CREDIT_BUREAU_QRT   265992 non-null  float64
 21  AMT_REQ_CREDIT_BUREAU_YEAR  265992 non-null  float64
dtypes: float64(6), int64(16)
memory usage: 51.6 MB
```

[110]:
```
# Task-2 - Missing Values Analysis

# Calculating the Percentage of Missing values
```

[111]:
```
data = application_train.isnull().mean()
result = data[data > 0].sort_values(ascending=False).head(50)*100
result.apply(lambda x: f"{x:.2f}%")
```

[111]:

COMMONAREA_MEDI	69.87%
COMMONAREA_AVG	69.87%
COMMONAREA_MODE	69.87%
NONLIVINGAPARTMENTS_MEDI	69.43%
NONLIVINGAPARTMENTS_MODE	69.43%
NONLIVINGAPARTMENTS_AVG	69.43%
FONDKAPREMONT_MODE	68.39%
LIVINGAPARTMENTS_MODE	68.35%
LIVINGAPARTMENTS_MEDI	68.35%
LIVINGAPARTMENTS_AVG	68.35%
FLOORSMIN_MODE	67.85%
FLOORSMIN_MEDI	67.85%
FLOORSMIN_AVG	67.85%
YEARS_BUILD_MODE	66.50%
YEARS_BUILD_MEDI	66.50%
YEARS_BUILD_AVG	66.50%
OWN_CAR_AGE	65.99%
LANDAREA_AVG	59.38%
LANDAREA_MEDI	59.38%
LANDAREA_MODE	59.38%
BASEMENTAREA_MEDI	58.52%
BASEMENTAREA_AVG	58.52%
BASEMENTAREA_MODE	58.52%
EXT_SOURCE_1	56.38%
NONLIVINGAREA_MEDI	55.18%
NONLIVINGAREA_MODE	55.18%
NONLIVINGAREA_AVG	55.18%
ELEVATORS_MEDI	53.30%
ELEVATORS_MODE	53.30%
ELEVATORS_AVG	53.30%
WALLSMATERIAL_MODE	50.84%
APARTMENTS_MODE	50.75%
APARTMENTS_MEDI	50.75%
APARTMENTS_AVG	50.75%
ENTRANCES_MODE	50.35%
ENTRANCES_AVG	50.35%
ENTRANCES_MEDI	50.35%
LIVINGAREA_MEDI	50.19%
LIVINGAREA_MODE	50.19%
LIVINGAREA_AVG	50.19%
HOUSETYPE_MODE	50.18%
FLOORSMAX_MEDI	49.76%
FLOORSMAX_AVG	49.76%

```
FLOORSMAX_MODE                  49.76%
YEARS_BEGINEXPLUATATION_AVG     48.78%
YEARS_BEGINEXPLUATATION_MEDI    48.78%
YEARS_BEGINEXPLUATATION_MODE    48.78%
TOTALAREA_MODE                  48.27%
EMERGENCYSTATE_MODE             47.40%
OCCUPATION_TYPE                 31.35%
dtype: object
```

[112]:
```
# Calculate the proportion of non-missing values for each column
train_missing = application_train.count() / len(application_train)

# Convert the proportions to percentages and calculate the percentage of␣
 ↪missing values for each column
train_missing = (1 - train_missing) * 100

# Sort the missing percentages in descending order and display the top 60␣
 ↪columns
train_missing.sort_values(ascending=False).head(60)
```

[112]:
```
COMMONAREA_MEDI                 69.872297
COMMONAREA_AVG                  69.872297
COMMONAREA_MODE                 69.872297
NONLIVINGAPARTMENTS_MODE        69.432963
NONLIVINGAPARTMENTS_AVG         69.432963
NONLIVINGAPARTMENTS_MEDI        69.432963
FONDKAPREMONT_MODE              68.386172
LIVINGAPARTMENTS_MODE           68.354953
LIVINGAPARTMENTS_AVG            68.354953
LIVINGAPARTMENTS_MEDI           68.354953
FLOORSMIN_AVG                   67.848630
FLOORSMIN_MODE                  67.848630
FLOORSMIN_MEDI                  67.848630
YEARS_BUILD_MEDI                66.497784
YEARS_BUILD_MODE                66.497784
YEARS_BUILD_AVG                 66.497784
OWN_CAR_AGE                     65.990810
LANDAREA_MEDI                   59.376738
LANDAREA_MODE                   59.376738
LANDAREA_AVG                    59.376738
BASEMENTAREA_MEDI               58.515956
BASEMENTAREA_AVG                58.515956
BASEMENTAREA_MODE               58.515956
EXT_SOURCE_1                    56.381073
NONLIVINGAREA_MODE              55.179164
NONLIVINGAREA_AVG               55.179164
NONLIVINGAREA_MEDI              55.179164
```

```
ELEVATORS_MEDI                  53.295980
ELEVATORS_AVG                   53.295980
ELEVATORS_MODE                  53.295980
WALLSMATERIAL_MODE              50.840783
APARTMENTS_MEDI                 50.749729
APARTMENTS_AVG                  50.749729
APARTMENTS_MODE                 50.749729
ENTRANCES_MEDI                  50.348768
ENTRANCES_AVG                   50.348768
ENTRANCES_MODE                  50.348768
LIVINGAREA_AVG                  50.193326
LIVINGAREA_MODE                 50.193326
LIVINGAREA_MEDI                 50.193326
HOUSETYPE_MODE                  50.176091
FLOORSMAX_MODE                  49.760822
FLOORSMAX_MEDI                  49.760822
FLOORSMAX_AVG                   49.760822
YEARS_BEGINEXPLUATATION_MODE    48.781019
YEARS_BEGINEXPLUATATION_MEDI    48.781019
YEARS_BEGINEXPLUATATION_AVG     48.781019
TOTALAREA_MODE                  48.268517
EMERGENCYSTATE_MODE             47.398304
OCCUPATION_TYPE                 31.345545
EXT_SOURCE_3                    19.825307
AMT_REQ_CREDIT_BUREAU_HOUR      13.501631
AMT_REQ_CREDIT_BUREAU_DAY       13.501631
AMT_REQ_CREDIT_BUREAU_WEEK      13.501631
AMT_REQ_CREDIT_BUREAU_MON       13.501631
AMT_REQ_CREDIT_BUREAU_QRT       13.501631
AMT_REQ_CREDIT_BUREAU_YEAR      13.501631
NAME_TYPE_SUITE                  0.420148
OBS_30_CNT_SOCIAL_CIRCLE         0.332021
DEF_30_CNT_SOCIAL_CIRCLE         0.332021
dtype: float64
```

[113]:
```
# Removing the Columns with more the 50% missing values

# Filter the DataFrame 'application_train' to include only columns with less than 50% missing values
train = application_train.loc[:, train_missing < 50]
train.head(1)
```

[113]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5

```
   …  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20 FLAG_DOCUMENT_21  \
0  …                 0                 0                 0                 0

  AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
0                        0.0                       0.0

   AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                         0.0                        0.0

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                        1.0

[1 rows x 81 columns]
```

[114]:
```
# Calculate the proportion of non-missing values for each column
train_missing_2 = train.count() / len(train)

# Convert the proportions to percentages and calculate the percentage of␣
 ↪missing values for each column
train_missing_2 = (1 - train_missing_2) * 100

# Sort the missing percentages in descending order
train_missing_2[train_missing_2 > 0].sort_values(ascending=False)
```

[114]:
```
FLOORSMAX_AVG                 49.760822
FLOORSMAX_MODE                49.760822
FLOORSMAX_MEDI                49.760822
YEARS_BEGINEXPLUATATION_AVG   48.781019
YEARS_BEGINEXPLUATATION_MODE  48.781019
YEARS_BEGINEXPLUATATION_MEDI  48.781019
TOTALAREA_MODE                48.268517
EMERGENCYSTATE_MODE           47.398304
OCCUPATION_TYPE               31.345545
EXT_SOURCE_3                  19.825307
AMT_REQ_CREDIT_BUREAU_HOUR    13.501631
AMT_REQ_CREDIT_BUREAU_QRT     13.501631
AMT_REQ_CREDIT_BUREAU_MON     13.501631
AMT_REQ_CREDIT_BUREAU_WEEK    13.501631
AMT_REQ_CREDIT_BUREAU_DAY     13.501631
AMT_REQ_CREDIT_BUREAU_YEAR    13.501631
NAME_TYPE_SUITE                0.420148
DEF_30_CNT_SOCIAL_CIRCLE       0.332021
OBS_60_CNT_SOCIAL_CIRCLE       0.332021
DEF_60_CNT_SOCIAL_CIRCLE       0.332021
OBS_30_CNT_SOCIAL_CIRCLE       0.332021
EXT_SOURCE_2                   0.214626
```

```
AMT_GOODS_PRICE                 0.090403
AMT_ANNUITY                     0.003902
CNT_FAM_MEMBERS                 0.000650
DAYS_LAST_PHONE_CHANGE          0.000325
dtype: float64
```

[115]:
```
# OCCUPATION_TYPE                31.345545
# EXT_SOURCE_3                   19.825307
# AMT_REQ_CREDIT_BUREAU_HOUR     13.501631
# AMT_REQ_CREDIT_BUREAU_QRT      13.501631
# AMT_REQ_CREDIT_BUREAU_MON      13.501631
# AMT_REQ_CREDIT_BUREAU_WEEK     13.501631
# AMT_REQ_CREDIT_BUREAU_DAY      13.501631
# AMT_REQ_CREDIT_BUREAU_YEAR     13.501631
# NAME_TYPE_SUITE                 0.420148
# DEF_30_CNT_SOCIAL_CIRCLE        0.332021
# OBS_60_CNT_SOCIAL_CIRCLE        0.332021
# DEF_60_CNT_SOCIAL_CIRCLE        0.332021
# OBS_30_CNT_SOCIAL_CIRCLE        0.332021
# EXT_SOURCE_2                    0.214626
# AMT_GOODS_PRICE                 0.090403
# AMT_ANNUITY                     0.003902
# CNT_FAM_MEMBERS                 0.000650
# DAYS_LAST_PHONE_CHANGE          0.000325
```

[116]:
```
# Column: DAYS_BIRTH

# Display the first few rows of the 'DAYS_BIRTH' column in the DataFrame 'train'
train['DAYS_BIRTH'].head()
```

[116]:
```
0    -9461
1   -16765
2   -19046
3   -19005
4   -19932
Name: DAYS_BIRTH, dtype: int64
```

[117]:
```
abs(train['DAYS_BIRTH']).head()
```

[117]:
```
0     9461
1    16765
2    19046
3    19005
4    19932
Name: DAYS_BIRTH, dtype: int64
```

[118]:
```
# Convert the values in the 'DAYS_BIRTH' column from days to years and display␣
↪the first few rows
train['DAYS_BIRTH'] = -round(train['DAYS_BIRTH'] / 365,0)
train['DAYS_BIRTH'].head()
```

[118]:
```
0    26.0
1    46.0
2    52.0
3    52.0
4    55.0
Name: DAYS_BIRTH, dtype: float64
```

[119]:
```
# Convert the values in the 'DAYS_REGISTRATION' column from days to years and␣
↪display the first few rows
train['DAYS_REGISTRATION'] = -round(train['DAYS_REGISTRATION'] / 365, 0)
train['DAYS_REGISTRATION'].head()
```

[119]:
```
0    10.0
1     3.0
2    12.0
3    27.0
4    12.0
Name: DAYS_REGISTRATION, dtype: float64
```

[120]:
```
# Convert the values in the 'DAYS_ID_PUBLISH' column from days to years and␣
↪display the first few rows
train['DAYS_ID_PUBLISH'] = -round(train['DAYS_ID_PUBLISH'] / 365, 0)
train['DAYS_ID_PUBLISH'].head()
```

[120]:
```
0    6.0
1    1.0
2    7.0
3    7.0
4    9.0
Name: DAYS_ID_PUBLISH, dtype: float64
```

[121]:
```
# Get the data types of each column in the DataFrame 'train'
train.dtypes
```

[121]:
```
SK_ID_CURR                    int64
TARGET                        int64
NAME_CONTRACT_TYPE           object
CODE_GENDER                  object
FLAG_OWN_CAR                 object
                              ...
AMT_REQ_CREDIT_BUREAU_DAY    float64
AMT_REQ_CREDIT_BUREAU_WEEK   float64
```

```
AMT_REQ_CREDIT_BUREAU_MON     float64
AMT_REQ_CREDIT_BUREAU_QRT     float64
AMT_REQ_CREDIT_BUREAU_YEAR    float64
Length: 81, dtype: object
```

[122]:
```
# Calculating the Ratio:

# Calculate the ratio of records with 'TARGET' value equal to 0 to records with␣
 ↪'TARGET' value equal to 1
(train['TARGET'] == 0).sum() / (train['TARGET'] == 1).sum()
```

[122]:
```
11.387150050352467
```

[123]:
```
# Task-3 - Analysing Categorical and Numerical data
# Analysing categorical data

# Create a subset of the DataFrame 'train' containing records
# where the 'TARGET' column is equal to 0
train_0 = train.loc[train['TARGET'] == 0]

# Create a subset of the DataFrame 'train' containing records
# where the 'TARGET' column is equal to 1
train_1 = train.loc[train['TARGET'] == 1]
```

[124]:
```
train['TARGET'].unique()
```

[124]:
```
array([1, 0], dtype=int64)
```

[125]:
```
def plotting(train, train0, train1, column):
    """
    Plots three types of visualizations for a given column in the dataset:
    a pie chart of overall distribution, a countplot by category, and a bar␣
 ↪plot of percentage distribution by target variable.

    Parameters:
    - train: DataFrame containing the entire dataset.
    - train0: DataFrame filtered by the target variable with value 0.
    - train1: DataFrame filtered by the target variable with value 1.
    - column: The name of the column to be visualized.
    """

    # Assigning dataframes to local variables (This step might be redundant as␣
 ↪we can directly use the function arguments)
    train = train
    train_0 = train0
    train_1 = train1
    col = column
```

```
    # Initialize figure with a specific size
    fig = plt.figure(figsize=(13,10))

    # Create a subplot for the pie chart
    ax1 = plt.subplot(221)
    # Plotting pie chart for overall distribution of the column
    train[col].value_counts().plot.pie(autopct="%1.0f%%", ax=ax1)
    plt.title('Plotting data for the column: ' + column)

    # Create a subplot for the countplot
    ax2 = plt.subplot(222)
    # Plotting count plot by category with hue as TARGET
    sns.countplot(x=column, hue='TARGET', data=train, ax=ax2)
    plt.xticks(rotation=90)
    plt.title('Plotting data for target in terms of total count')

    # Create a subplot for the bar plot
    ax3 = plt.subplot(223)
    # Preparing data for percentage distribution by target variable
    df = pd.DataFrame()
    df['0'] = ((train_0[col].value_counts())/len(train_0))
    df['1'] = ((train_1[col].value_counts())/len(train_1))
    # Plotting bar plot for percentage distribution
    df.plot.bar(ax=ax3)
    plt.title('Plotting data for target in terms of percentage')

    # Adjust layout to prevent overlap
    fig.tight_layout()

    # Display the plots
    plt.show()
```

[126]:
```
# Create a list to get all the Categorical columns
train_categorical = train.select_dtypes(include=['object']).columns
```

[127]:
```
train_categorical
```

[127]:
```
Index(['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
       'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE',
       'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE',
       'EMERGENCYSTATE_MODE'],
      dtype='object')
```

[128]:
```
# Task-4.1 - Univariate Analysis of the Categorical data
# Visualising Data Distribution
```

[129]:
```
train['NAME_CONTRACT_TYPE'].unique()
```

[129]:
```
array(['Cash loans', 'Revolving loans'], dtype=object)
```

[130]:
```
train.CODE_GENDER.unique()
```

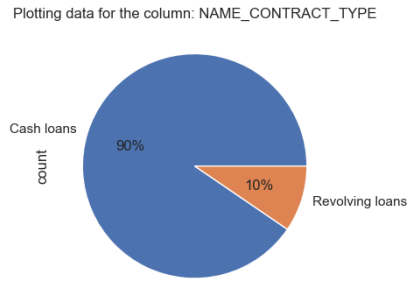
[130]:
```
array(['M', 'F', 'XNA'], dtype=object)
```

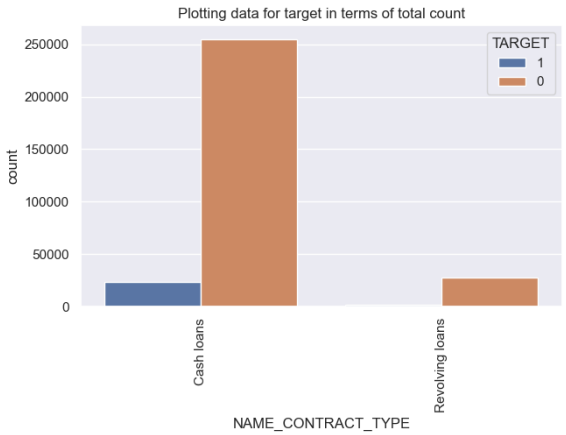
[131]:
```
train_categorical
```

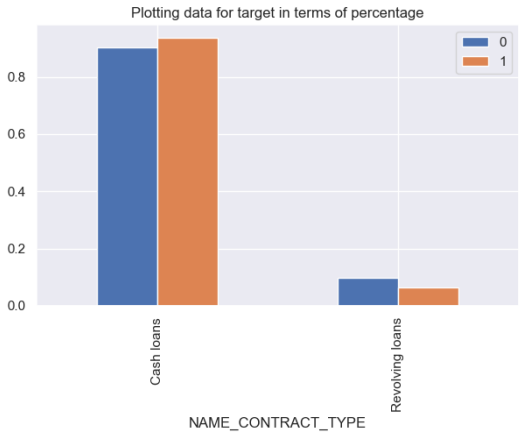
[131]:
```
Index(['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
       'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE',
       'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE',
       'EMERGENCYSTATE_MODE'],
      dtype='object')
```

[132]:
```
# Convert 'TARGET' to a string data type for categorical processing.
train['TARGET'] = train['TARGET'].astype(str)

# Change 'NAME_CONTRACT_TYPE' to a categorical type for optimized storage and␣
 ↪plotting.
train['NAME_CONTRACT_TYPE'] = train['NAME_CONTRACT_TYPE'].astype('category')

# Iterate through a list of categorical column names, creating plots for each.
for column in train_categorical:
    # Indicates the column currently being plotted.
    print("Plotting ", column)

    # Generate visualizations for the current column against the TARGET␣
 ↪variable.
    plotting(train, train_0, train_1, column)

    # Prints a separator line for readability between plots of different␣
 ↪columns.
  ␣
 ↪print('------------------------------------------------------------------------')
```

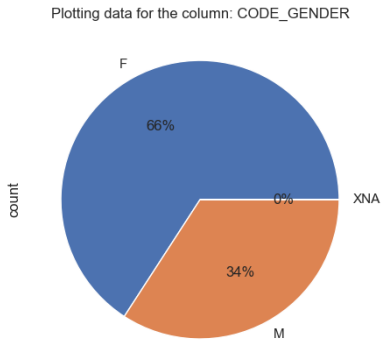
```
Plotting  NAME_CONTRACT_TYPE
```

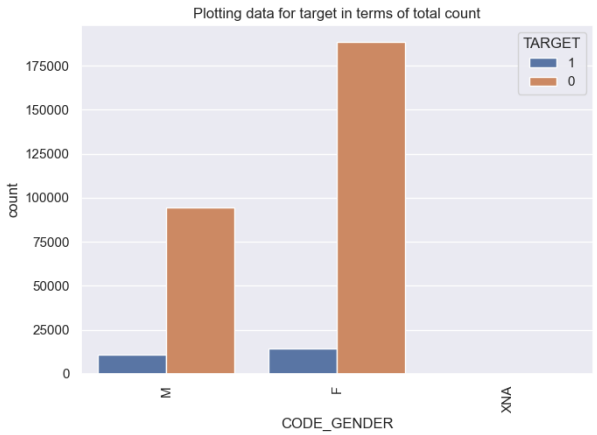


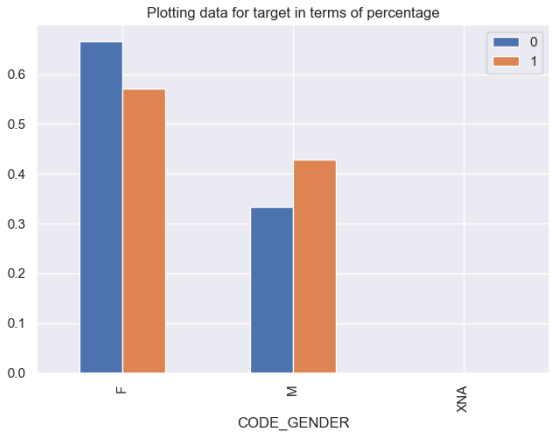




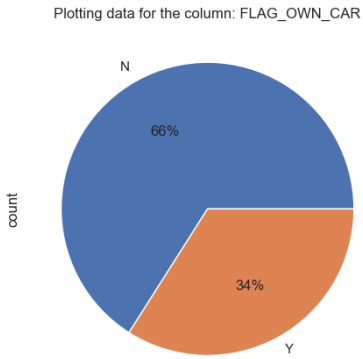
Plotting CODE_GENDER

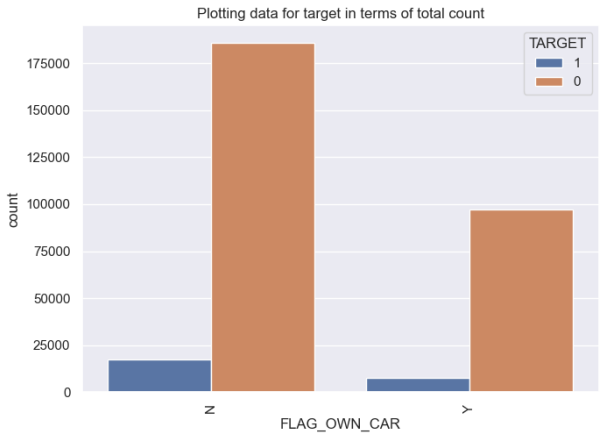


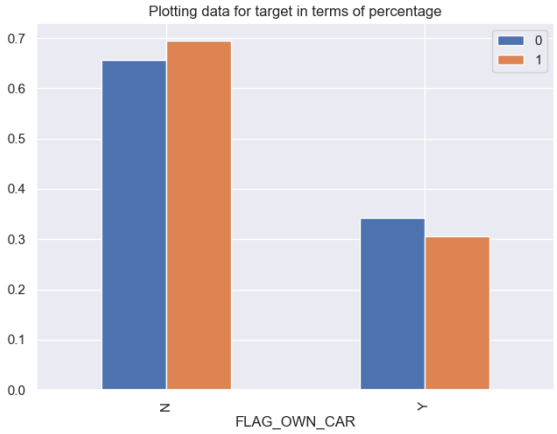




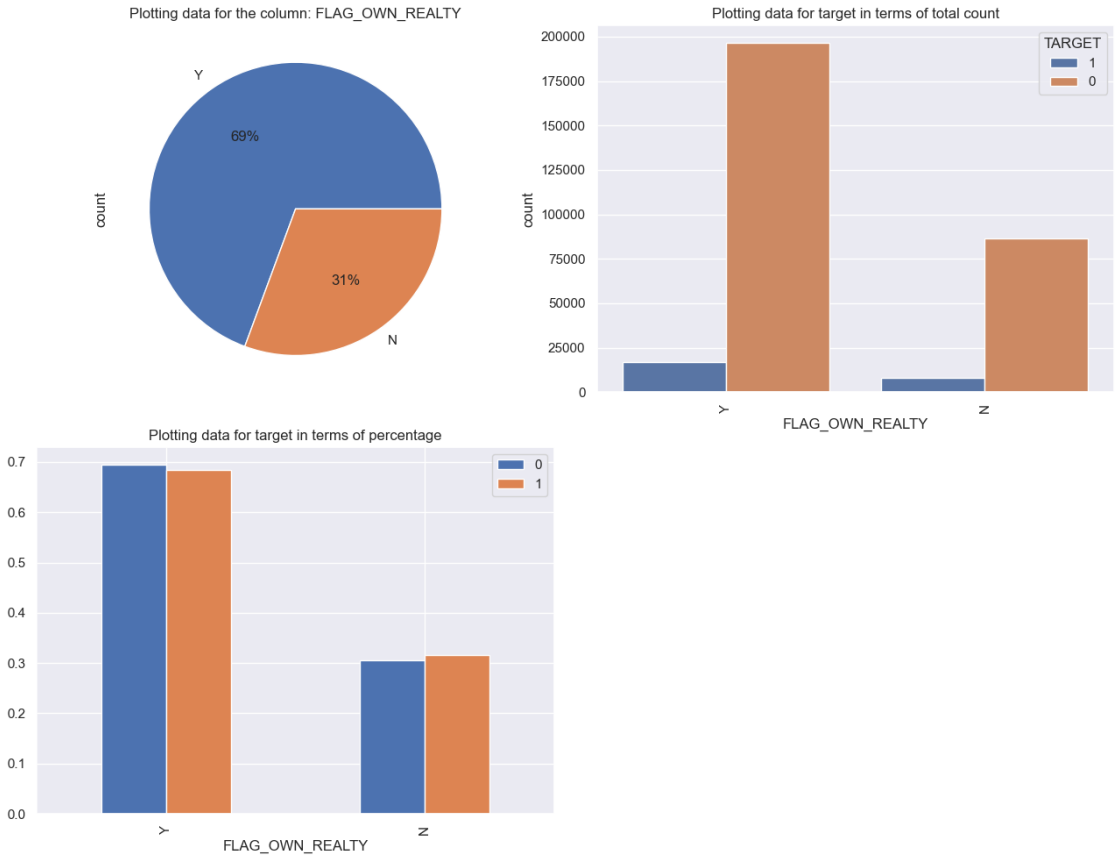
Plotting FLAG_OWN_CAR



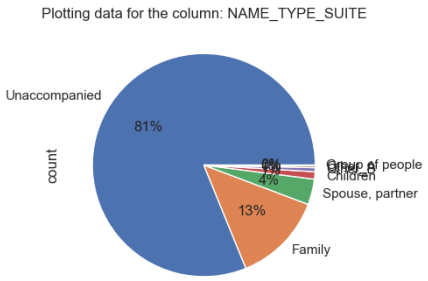


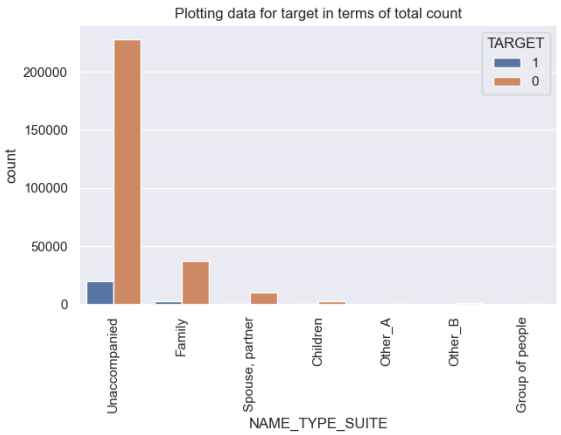


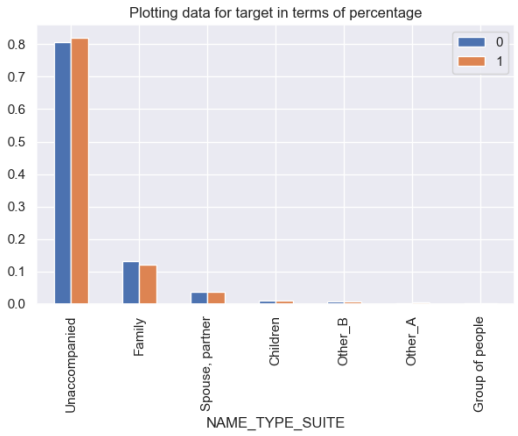
Plotting FLAG_OWN_REALTY



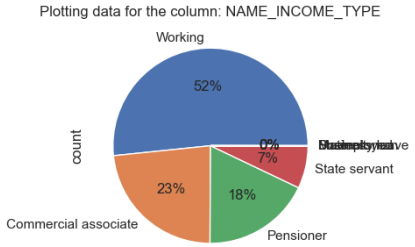
Plotting NAME_TYPE_SUITE

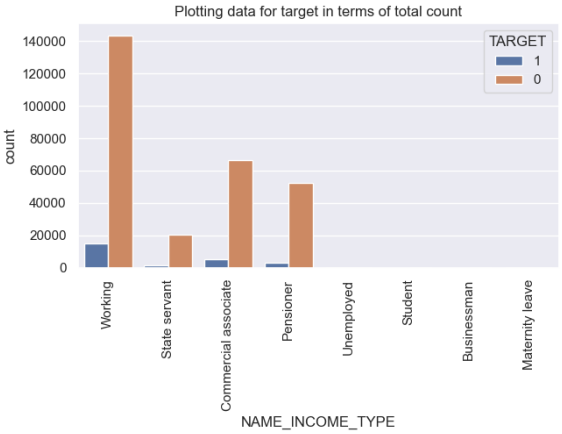


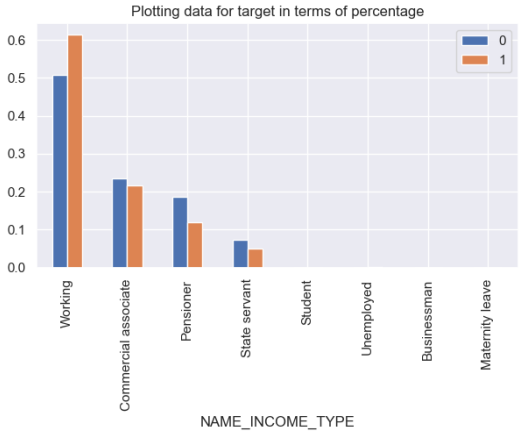




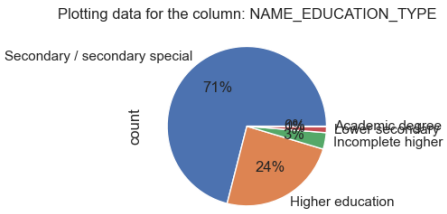
Plotting NAME_INCOME_TYPE

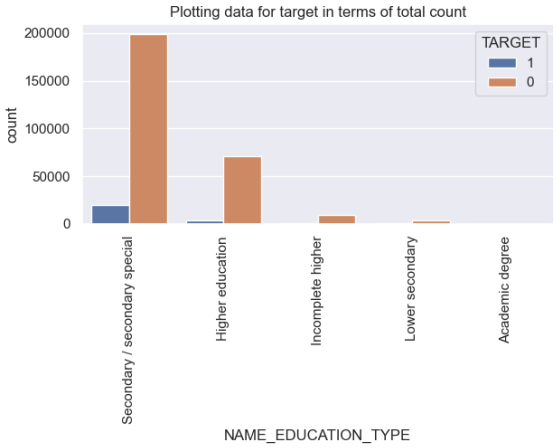


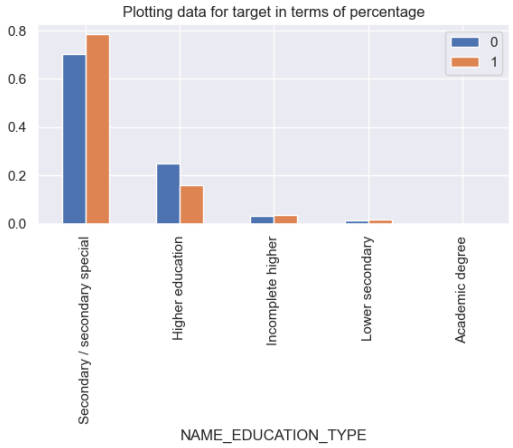




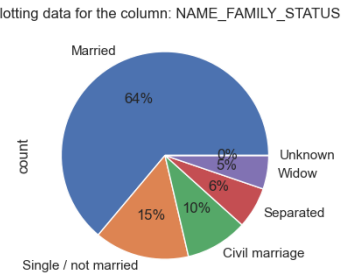
Plotting NAME_EDUCATION_TYPE

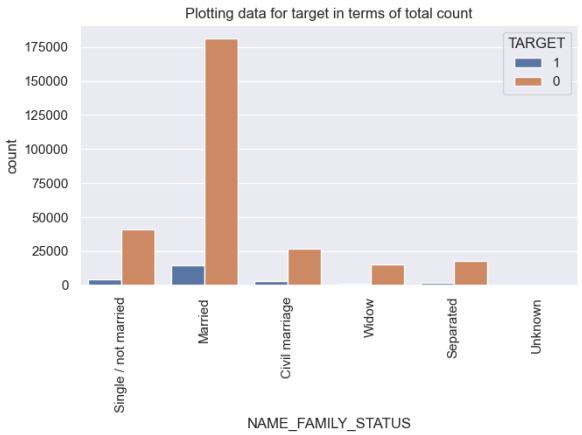


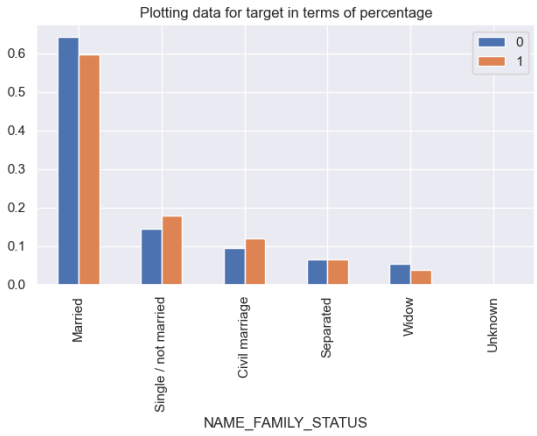




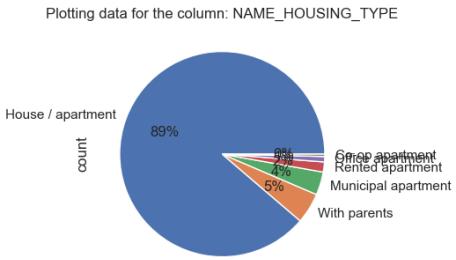
Plotting NAME_FAMILY_STATUS

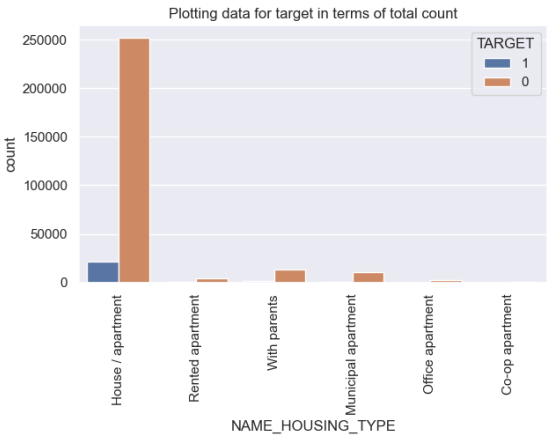


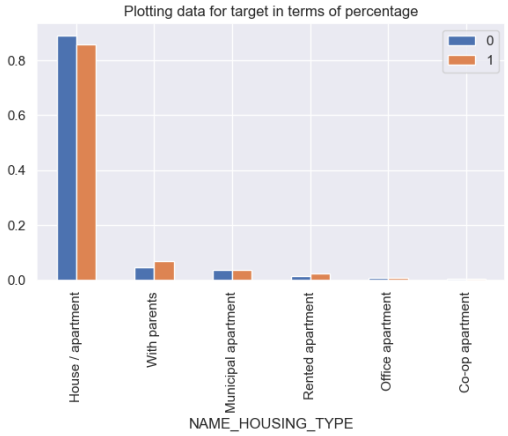




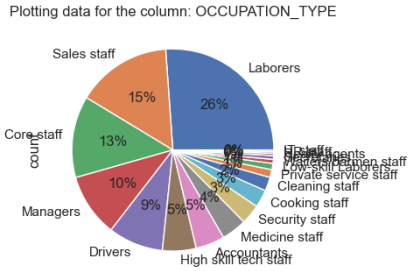
Plotting NAME_HOUSING_TYPE

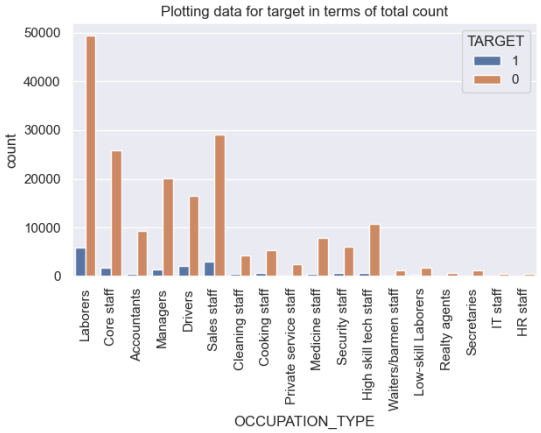


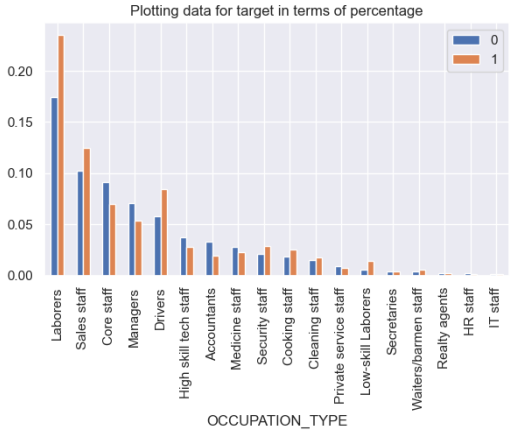




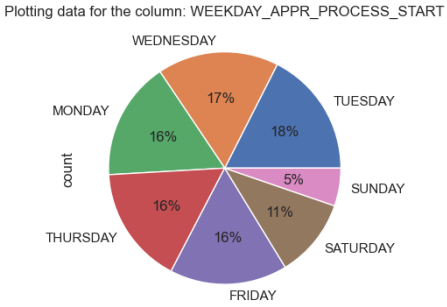
Plotting OCCUPATION_TYPE

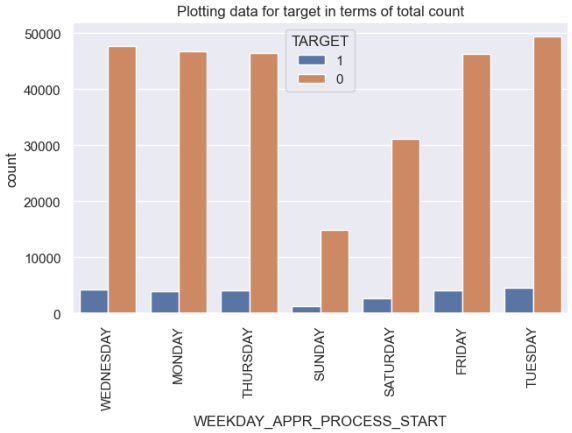


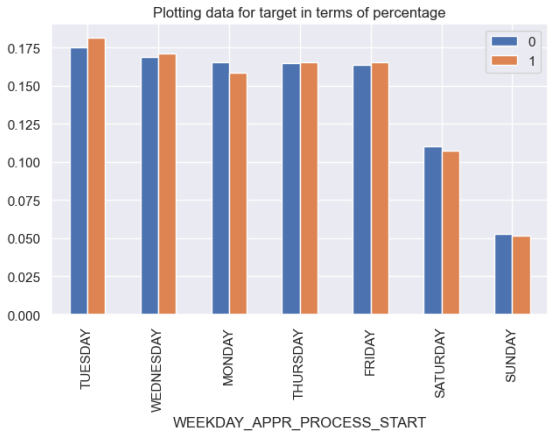




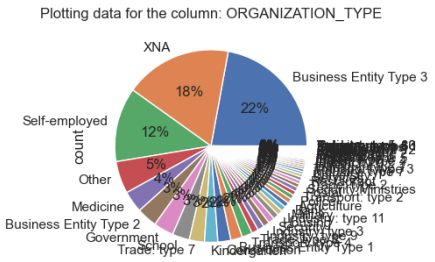
Plotting WEEKDAY_APPR_PROCESS_START

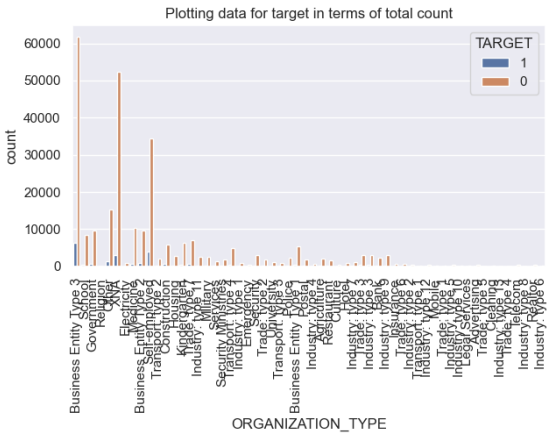


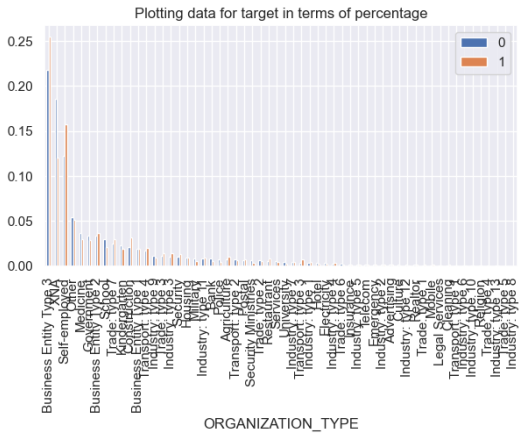




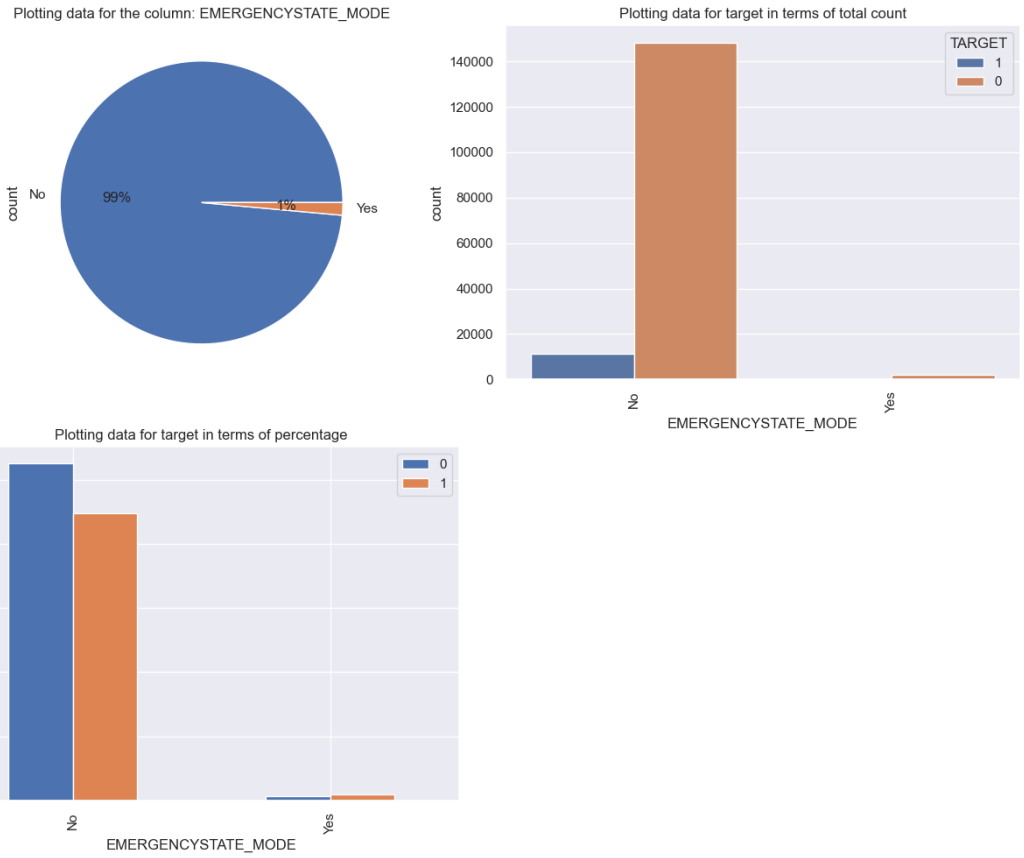
Plotting ORGANIZATION_TYPE







Plotting EMERGENCYSTATE_MODE



---

[133]:
```
# Select columns with numerical data types ('int64' and 'float64') from the␣
 ↪DataFrame 'train'
train_categorical = train.select_dtypes(include=['int64', 'float64']).columns
```

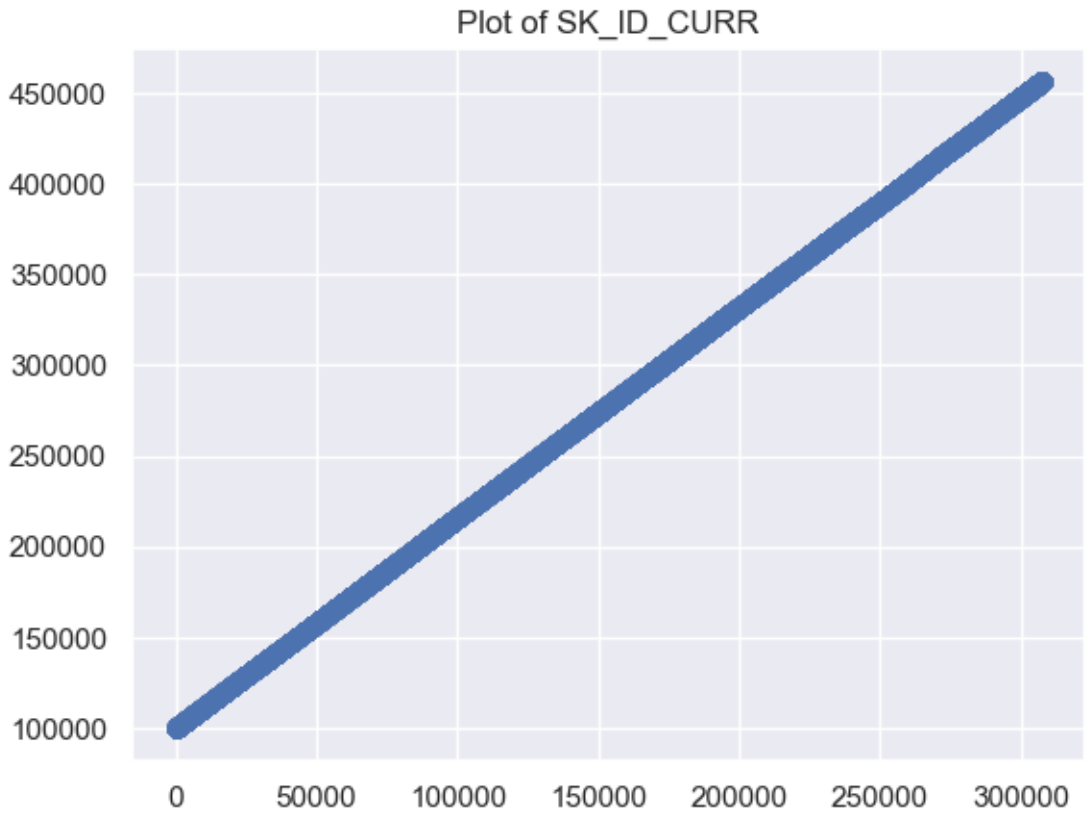
[134]:
```
# Task-5 - Analysis for the outliers
# Plotting the numerial data based on the index and analysing if there are␣
 ↪outliers in any of the columns:
```

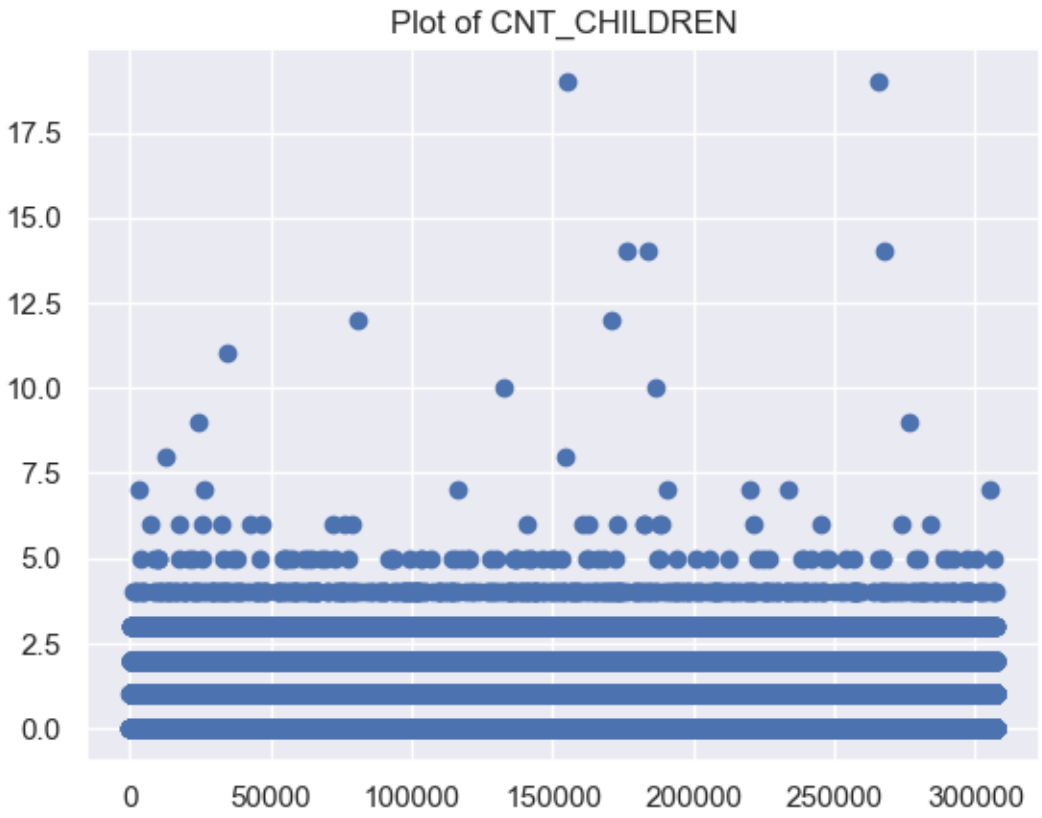
[135]:
```
train_categorical
```

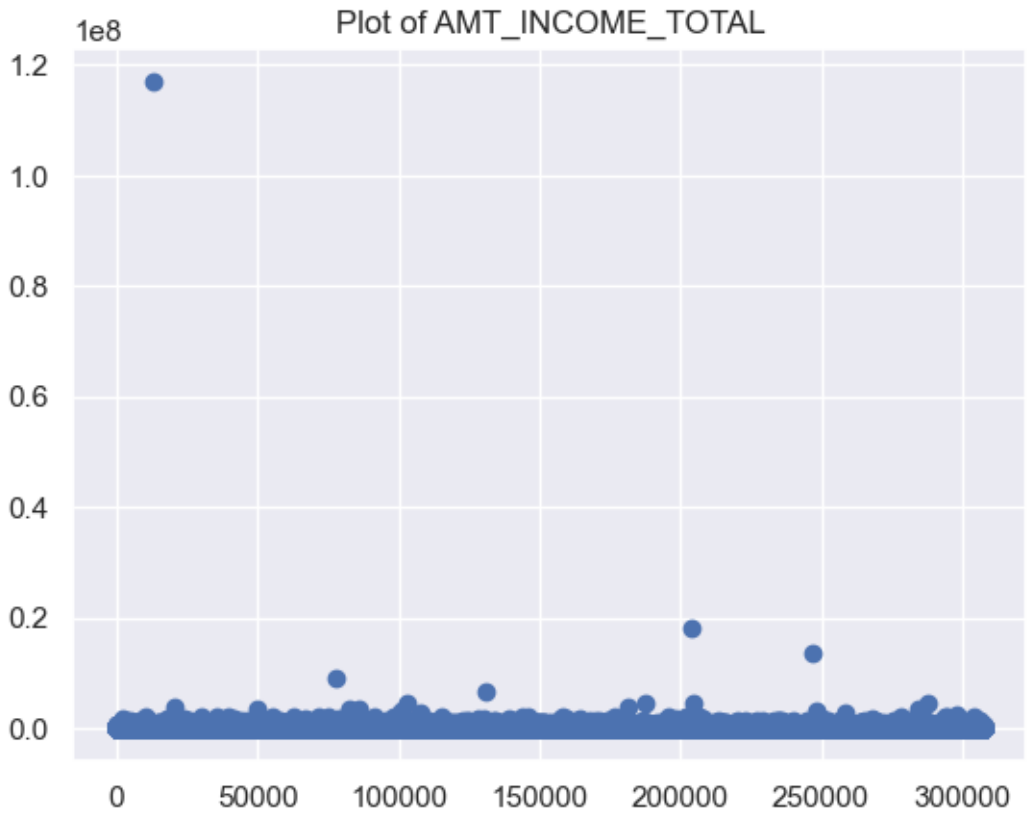
[135]:
```
Index(['SK_ID_CURR', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT',
       'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'REGION_POPULATION_RELATIVE',
       'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH',
       'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE',
       'FLAG_PHONE', 'FLAG_EMAIL', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'HOUR_APPR_PROCESS_START',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
```

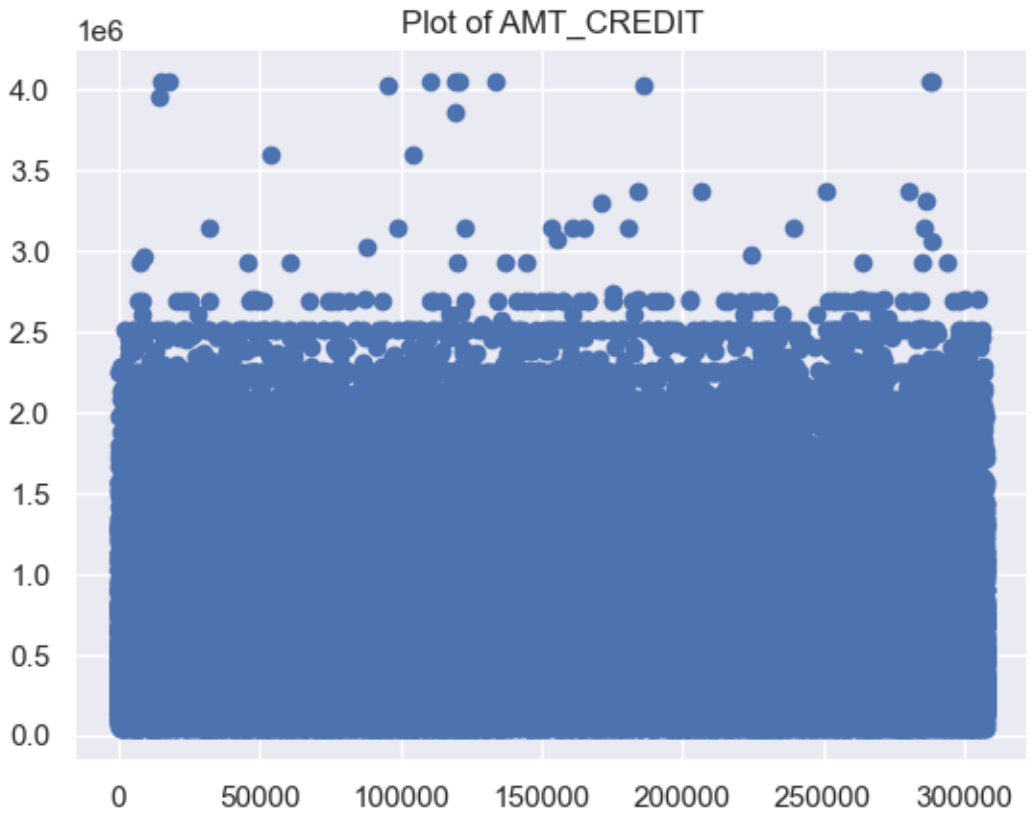
```
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'EXT_SOURCE_2',
       'EXT_SOURCE_3', 'YEARS_BEGINEXPLUATATION_AVG', 'FLOORSMAX_AVG',
       'YEARS_BEGINEXPLUATATION_MODE', 'FLOORSMAX_MODE',
       'YEARS_BEGINEXPLUATATION_MEDI', 'FLOORSMAX_MEDI', 'TOTALAREA_MODE',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
       'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
       'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9',
       'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12',
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15',
       'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', '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')
```

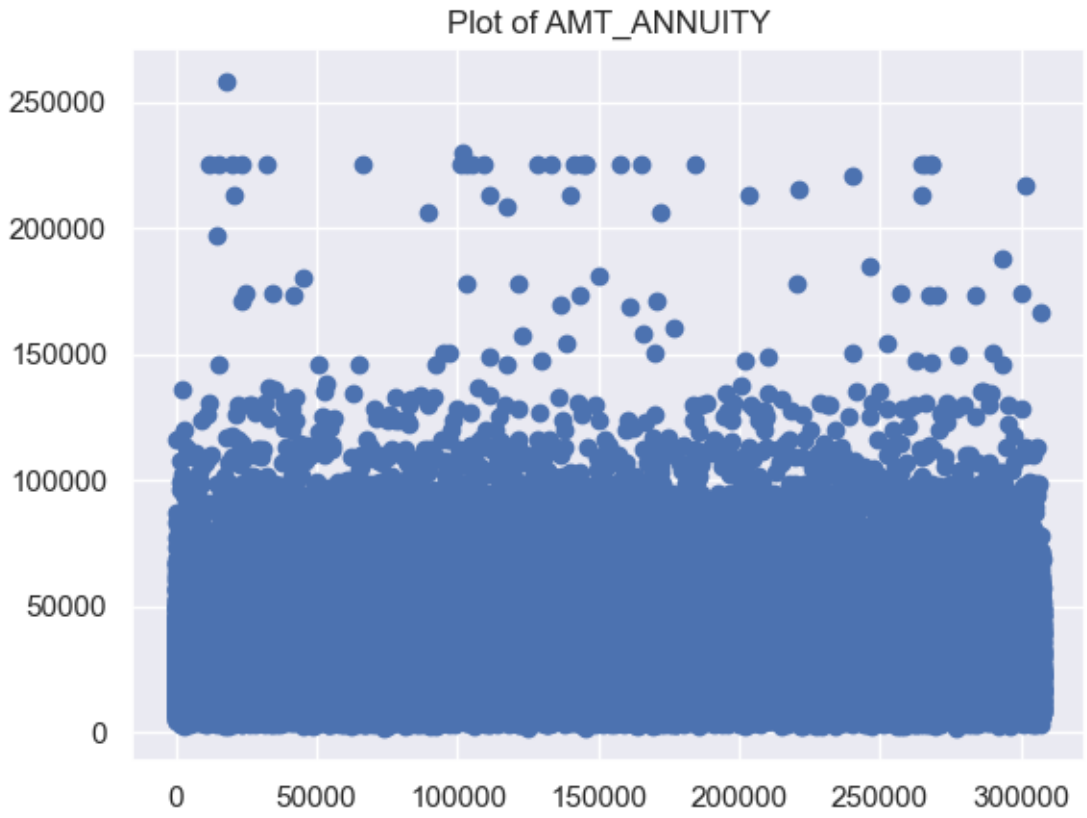
[136]:
```
# Plot each numerical column against the index of the DataFrame 'train'
for column in train_categorical:
    title = "Plot of " + column
    plt.scatter(train.index, train[column])
    plt.title(title)
    plt.show()
```

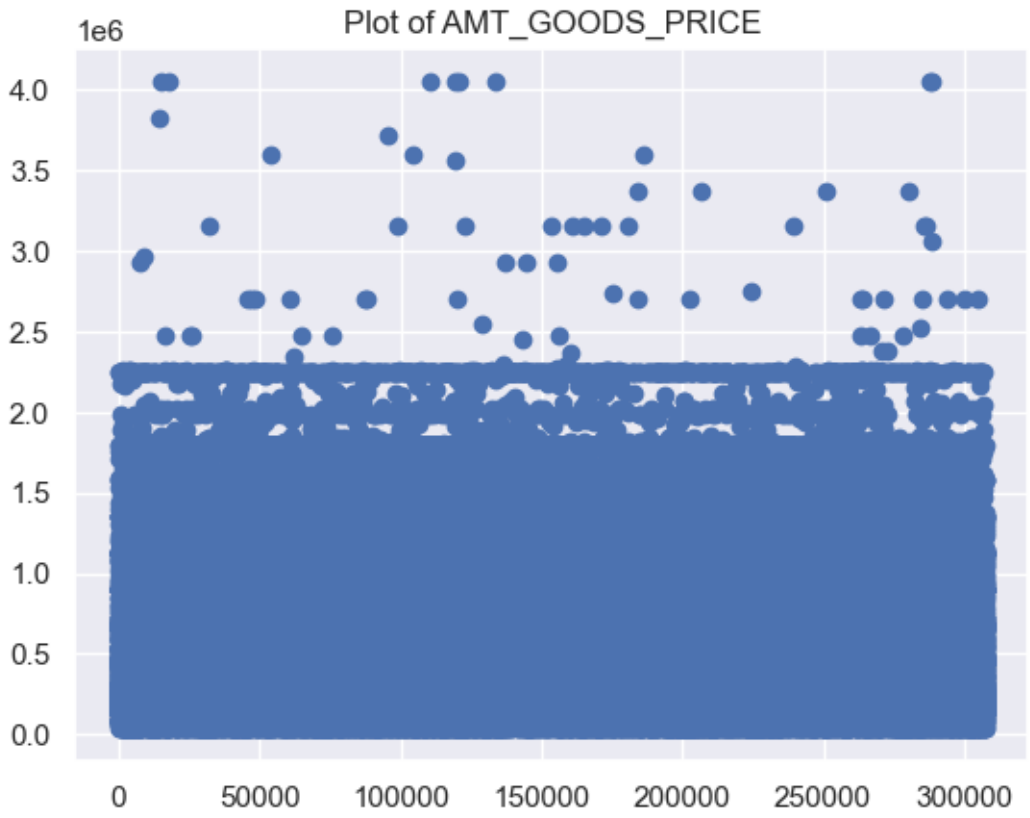

Plot of SK_ID_CURR
450000
400000
350000
300000
250000
200000
150000
100000
0
50000
100000
150000
200000
250000
300000

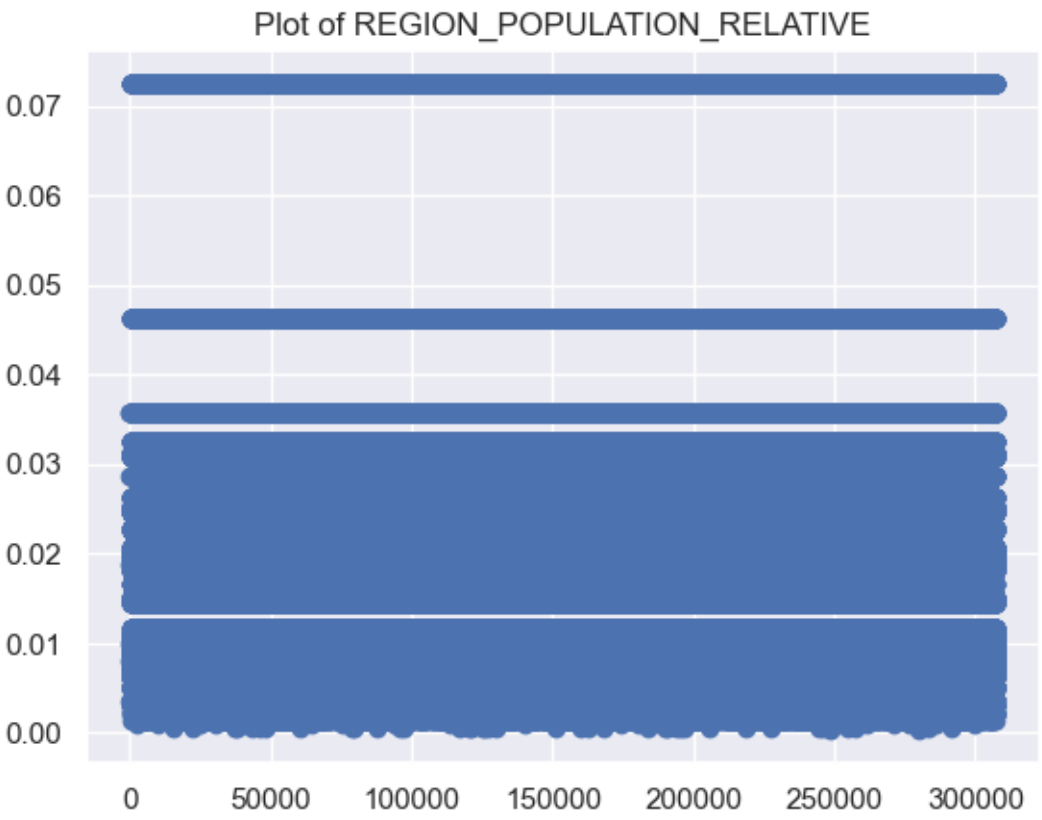

Plot of CNT_CHILDREN
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
50000
100000
150000
200000
250000
300000

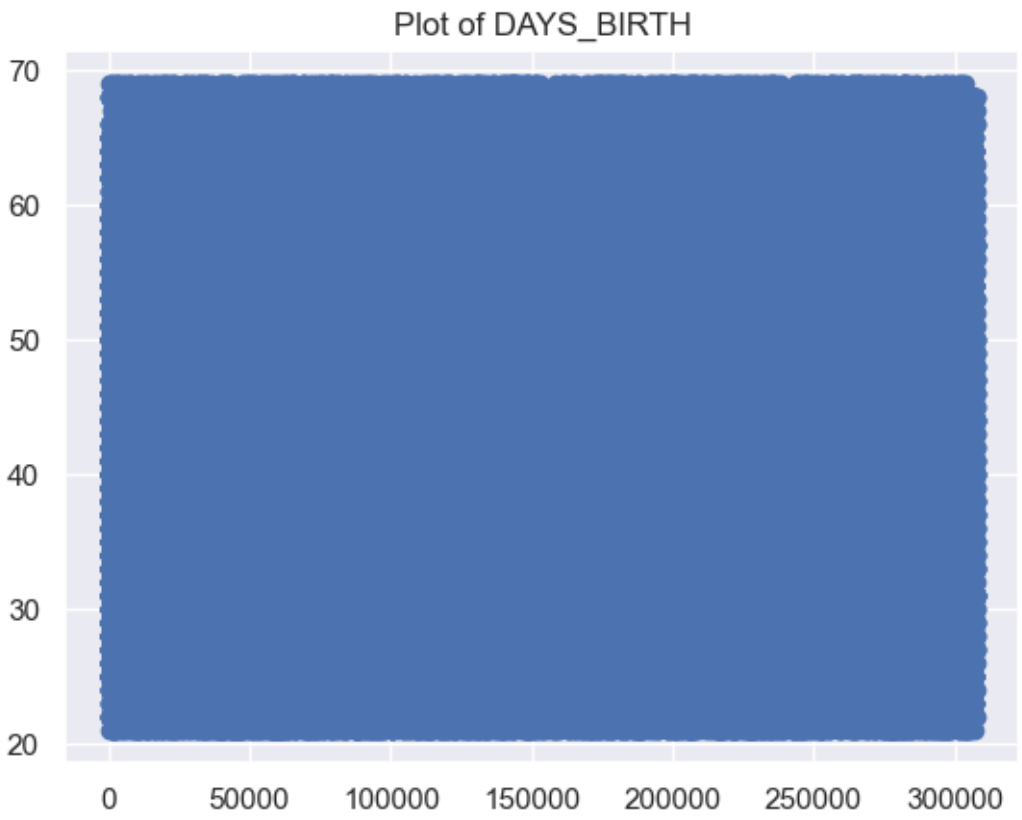

Plot of AMT_INCOME_TOTAL
1e8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000


Plot of AMT_CREDIT
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50000
100000
150000
200000
250000
300000

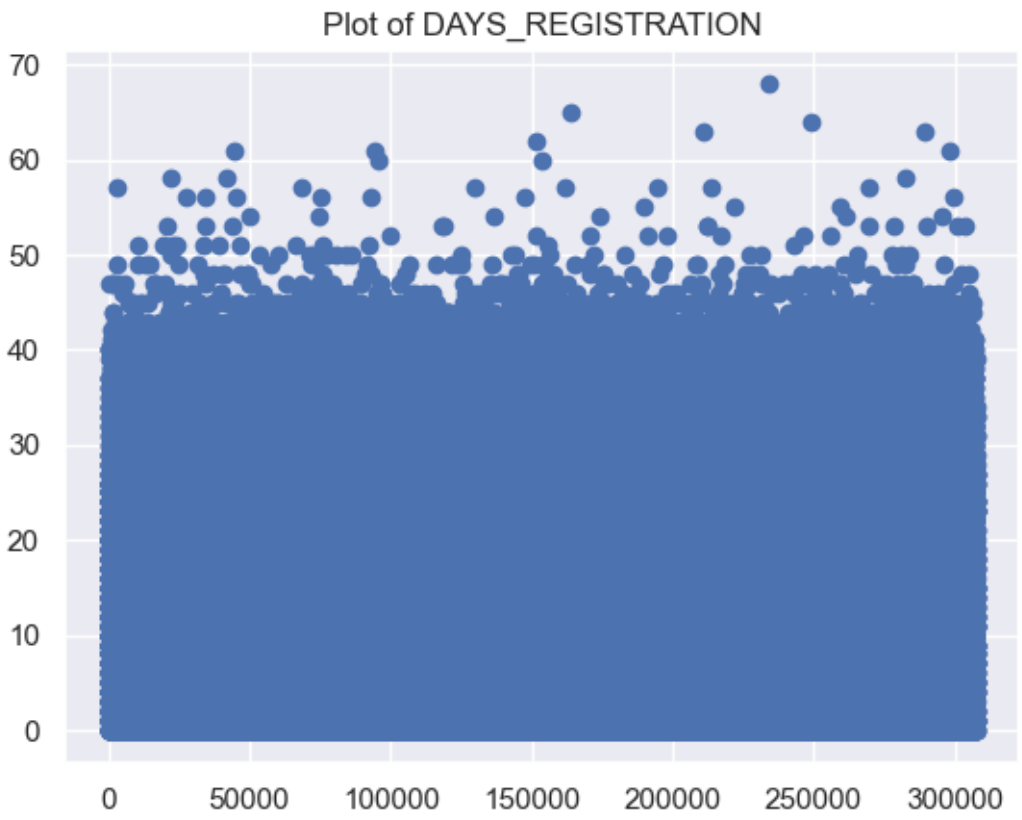

Plot of AMT_ANNUITY
250000
200000
150000
100000
50000
0
0
50000
100000
150000
200000
250000
300000

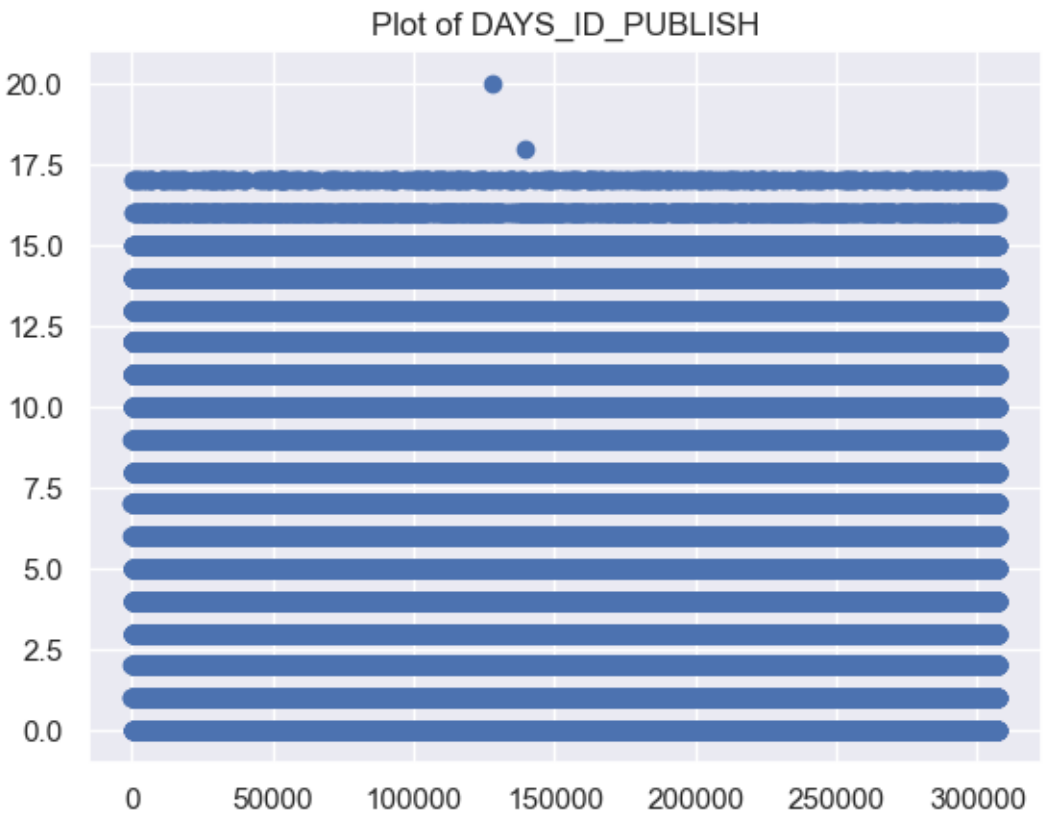

Plot of AMT_GOODS_PRICE
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50000
100000
150000
200000
250000
300000

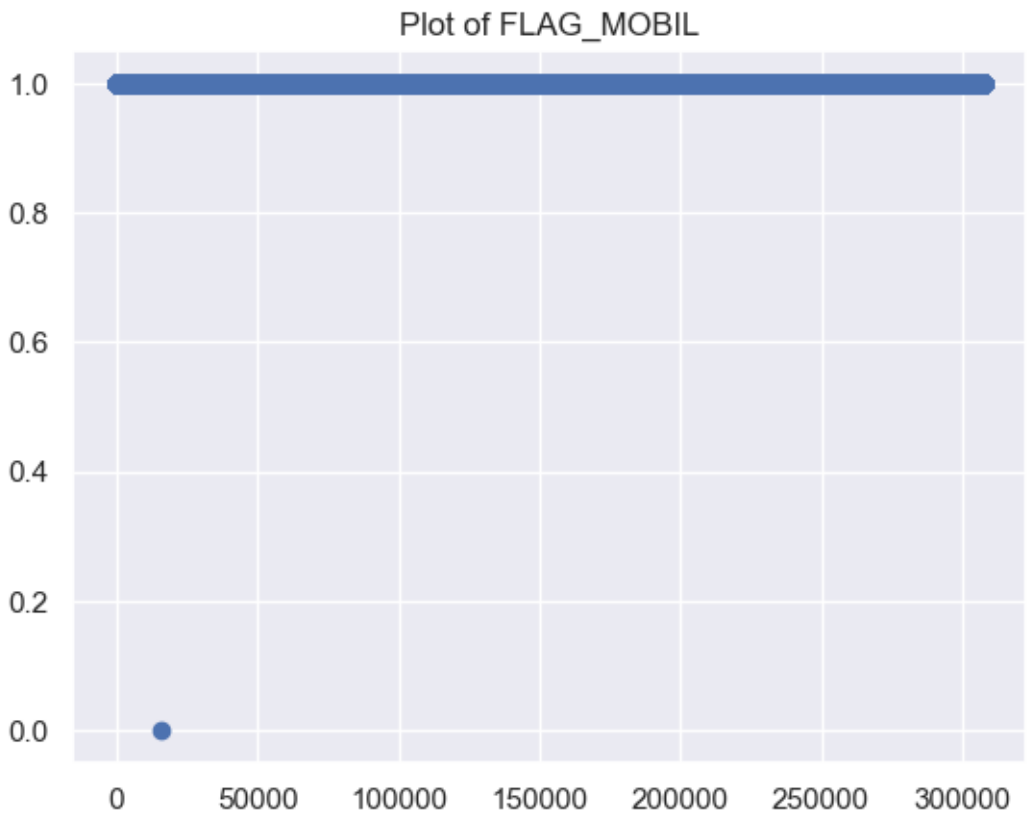

Plot of REGION_POPULATION_RELATIVE
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
50000
100000
150000
200000
250000
300000

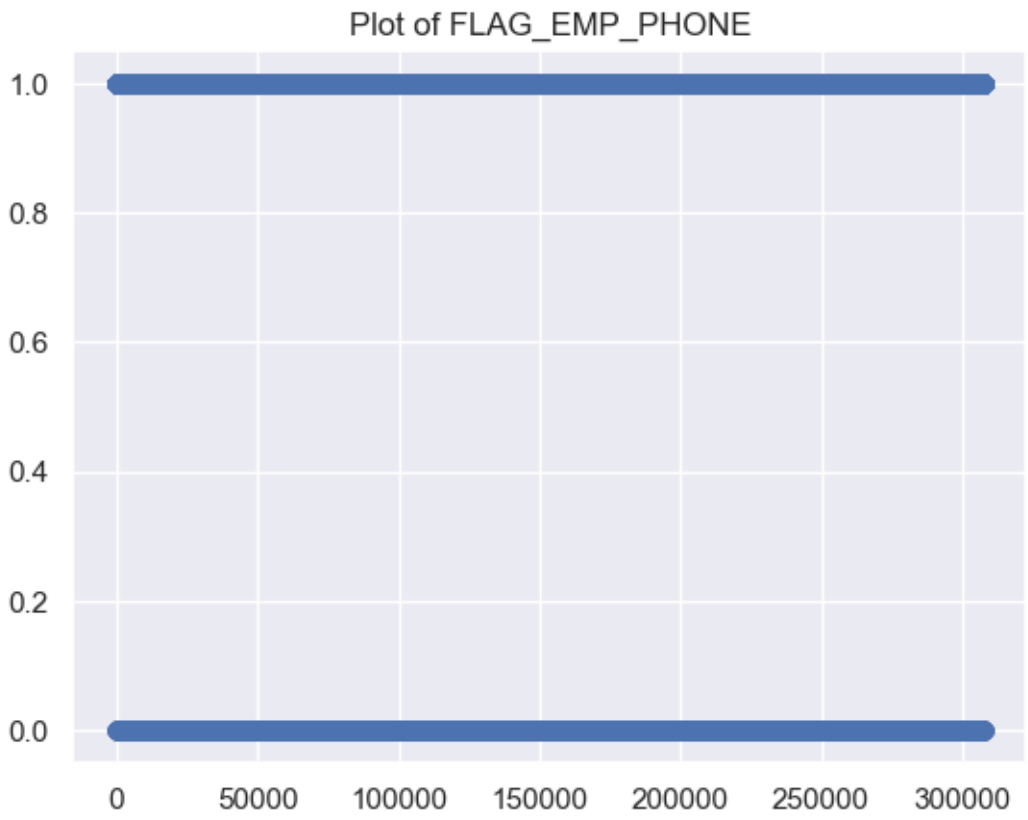

Plot of DAYS_BIRTH
70
60
50
40
30
20
0
50000
100000
150000
200000
250000
300000

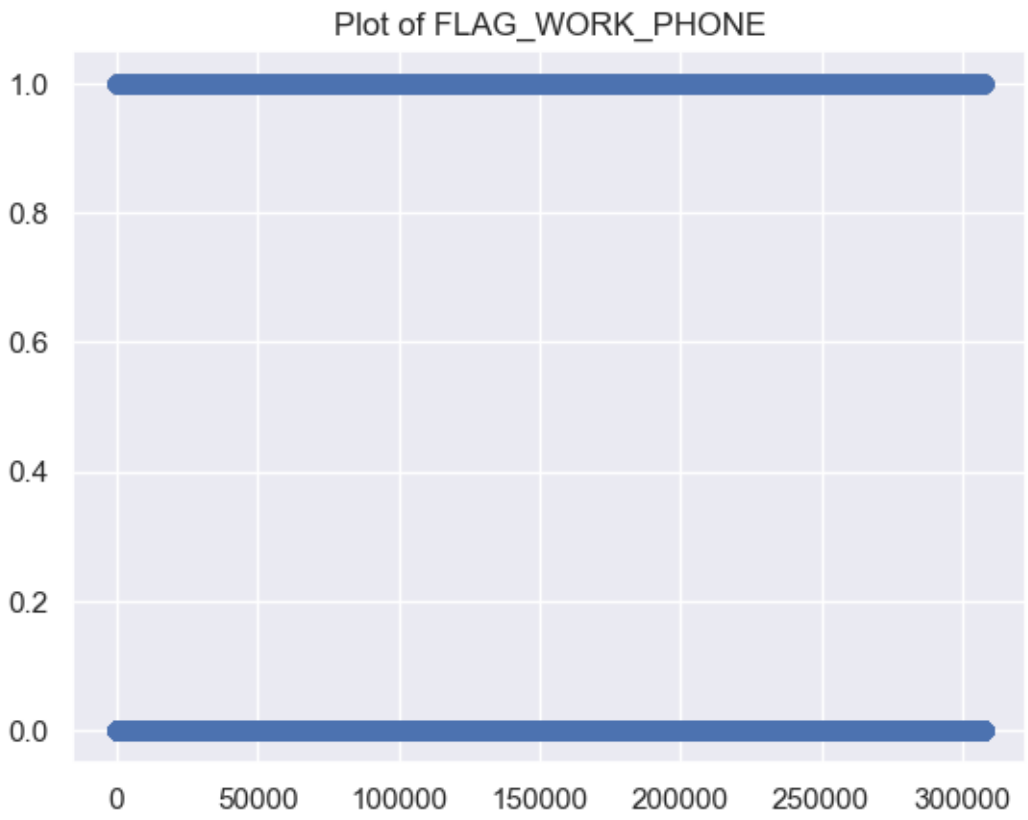

Plot of DAYS_EMPLOYED
350000
300000
250000
200000
150000
100000
50000
0
0
50000
100000
150000
200000
250000
300000

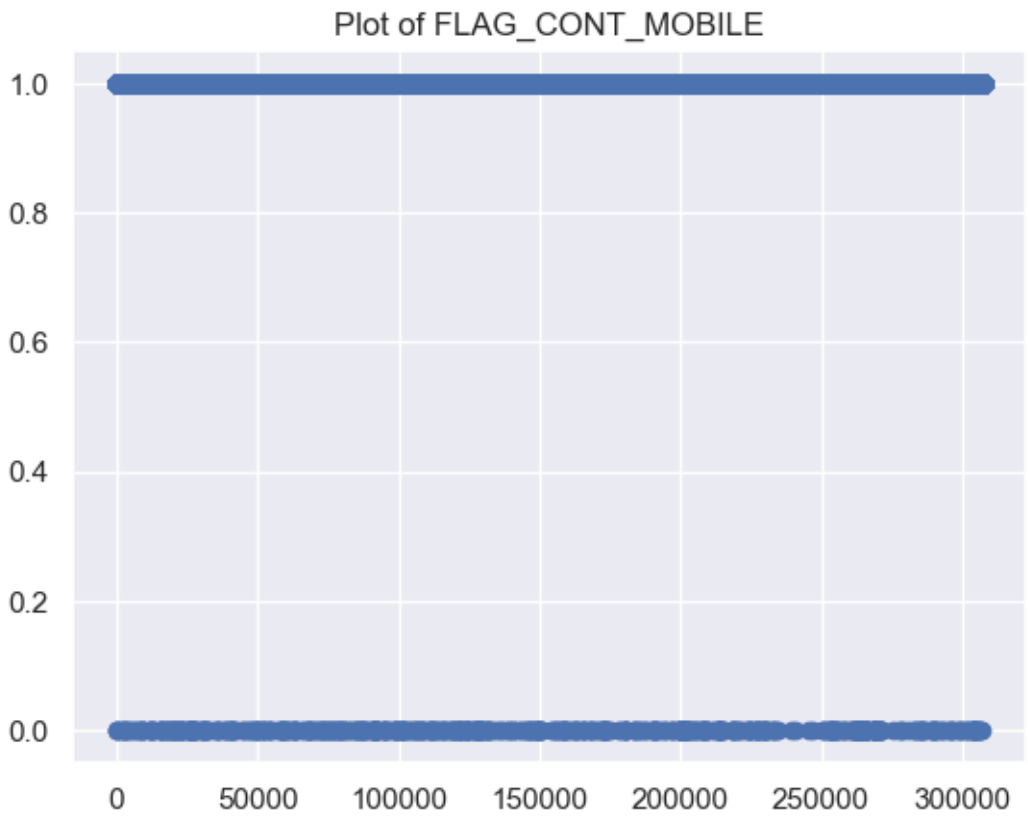

Plot of DAYS_REGISTRATION
70
60
50
40
30
20
10
0
0
50000
100000
150000
200000
250000
300000

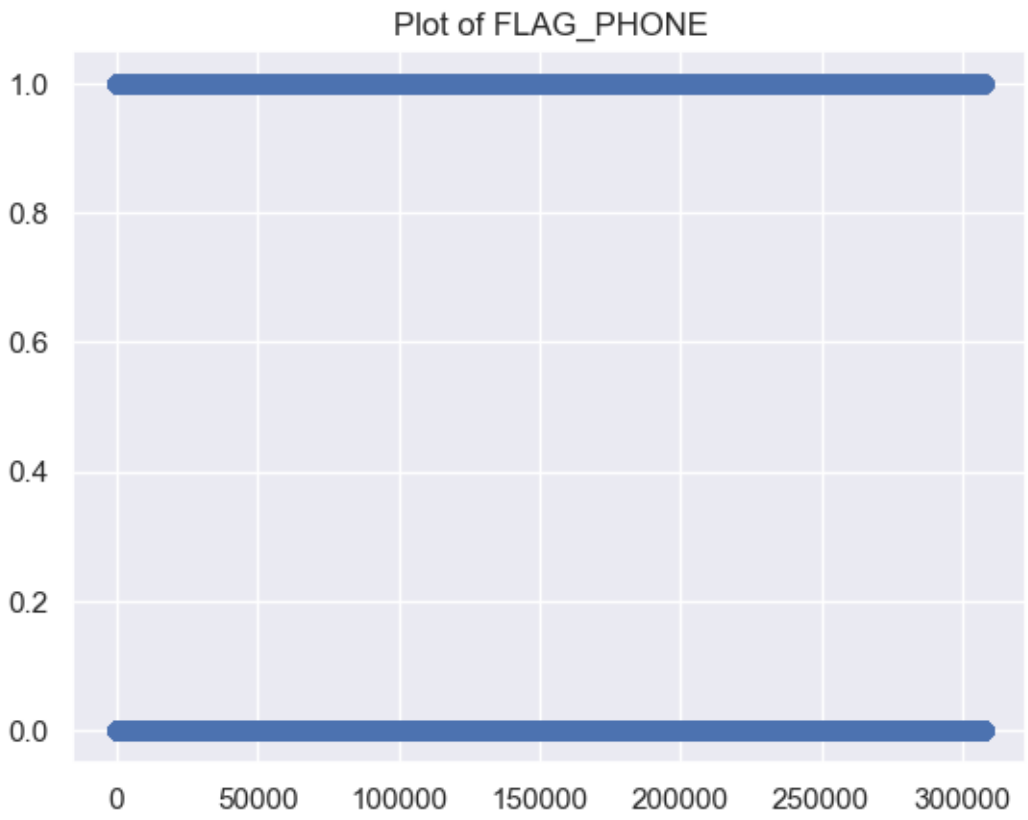

Plot of DAYS_ID_PUBLISH
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
50000
100000
150000
200000
250000
300000

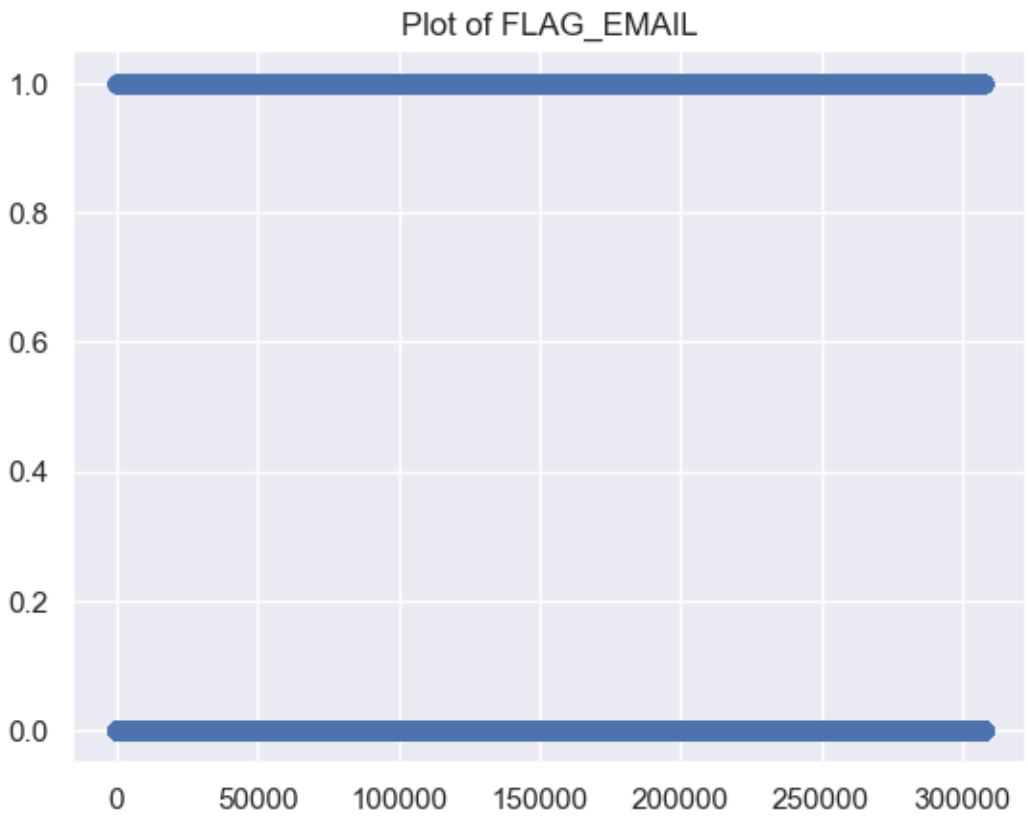

Plot of FLAG_MOBIL
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

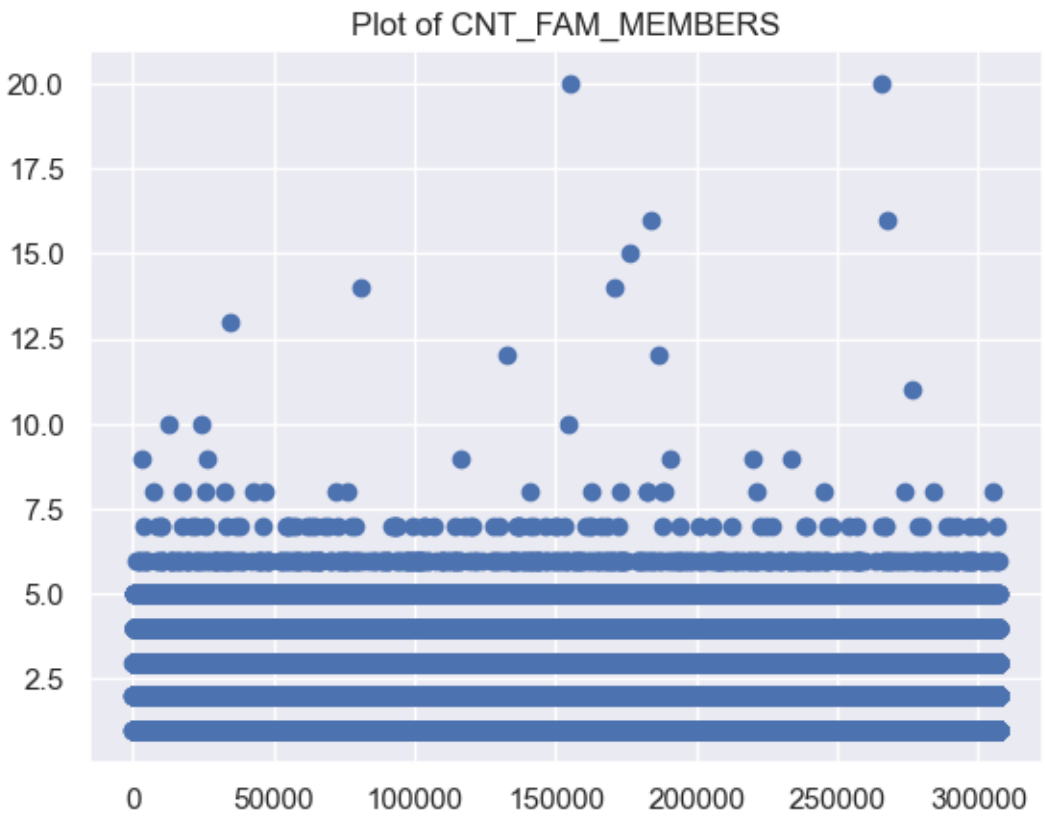

Plot of FLAG_EMP_PHONE
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000


Plot of FLAG_WORK_PHONE
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

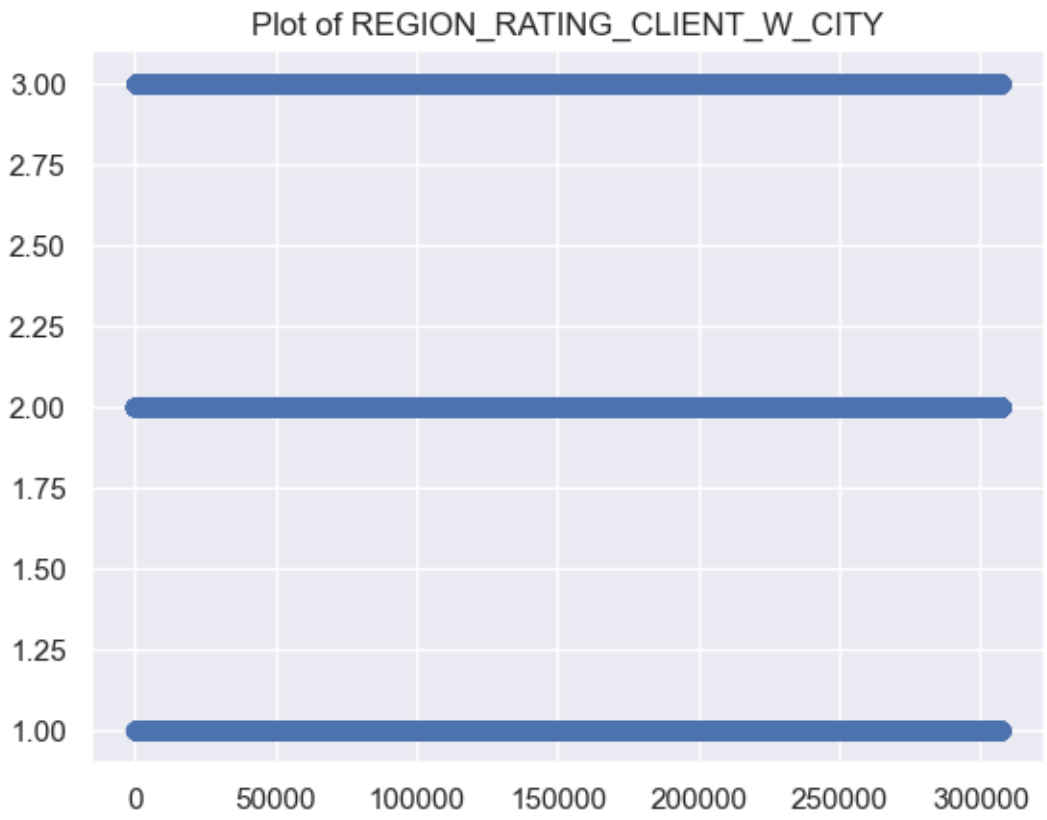

Plot of FLAG_CONT_MOBILE
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

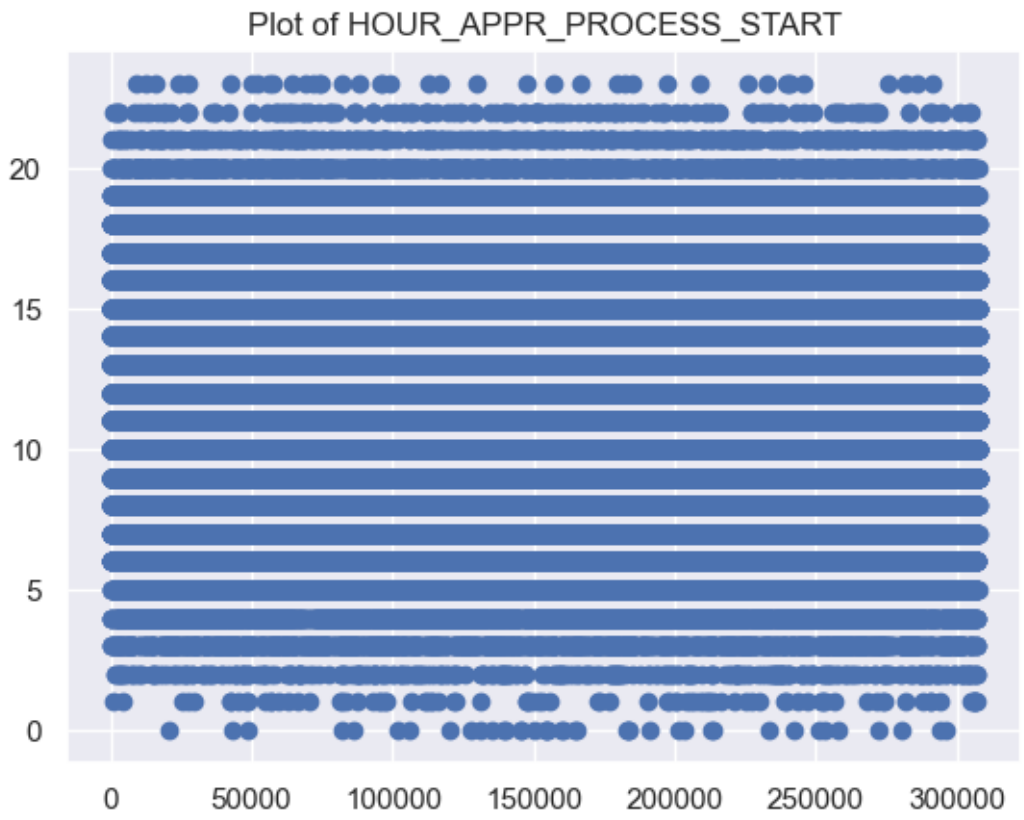

Plot of FLAG_PHONE
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

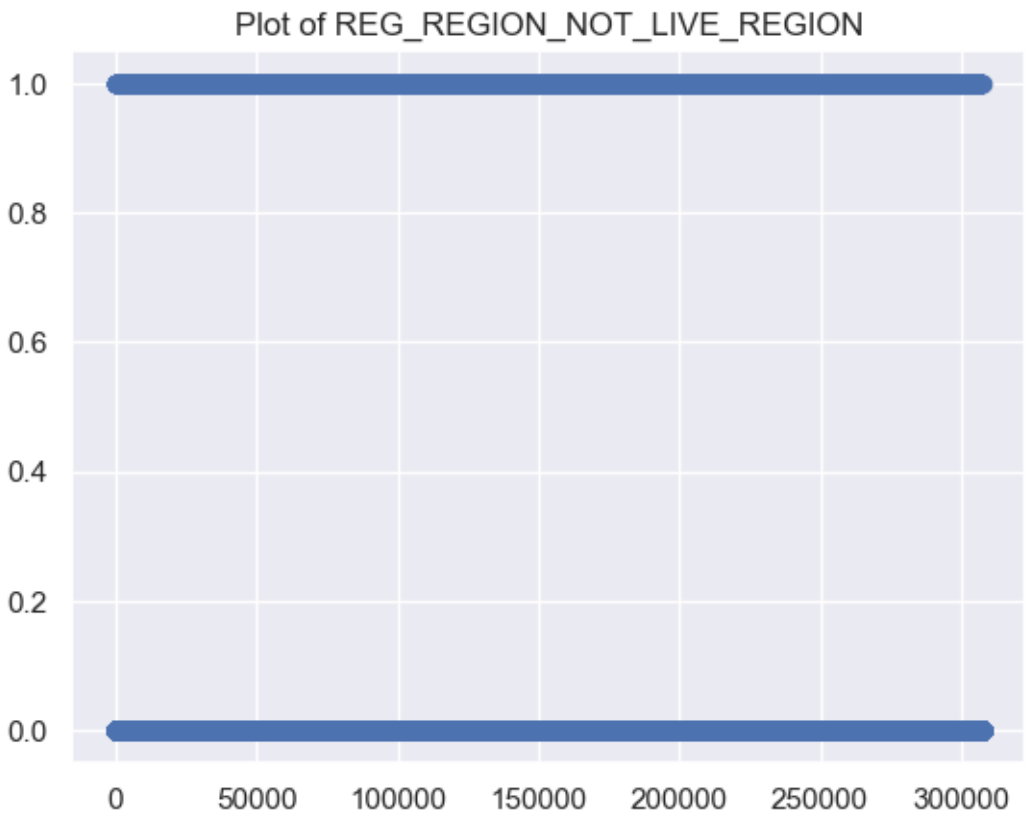

Plot of FLAG_EMAIL
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000


Plot of CNT_FAM_MEMBERS
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
50000
100000
150000
200000
250000
300000


Plot of REGION_RATING_CLIENT
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
50000
100000
150000
200000
250000
300000


Plot of REGION_RATING_CLIENT_W_CITY
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
50000
100000
150000
200000
250000
300000

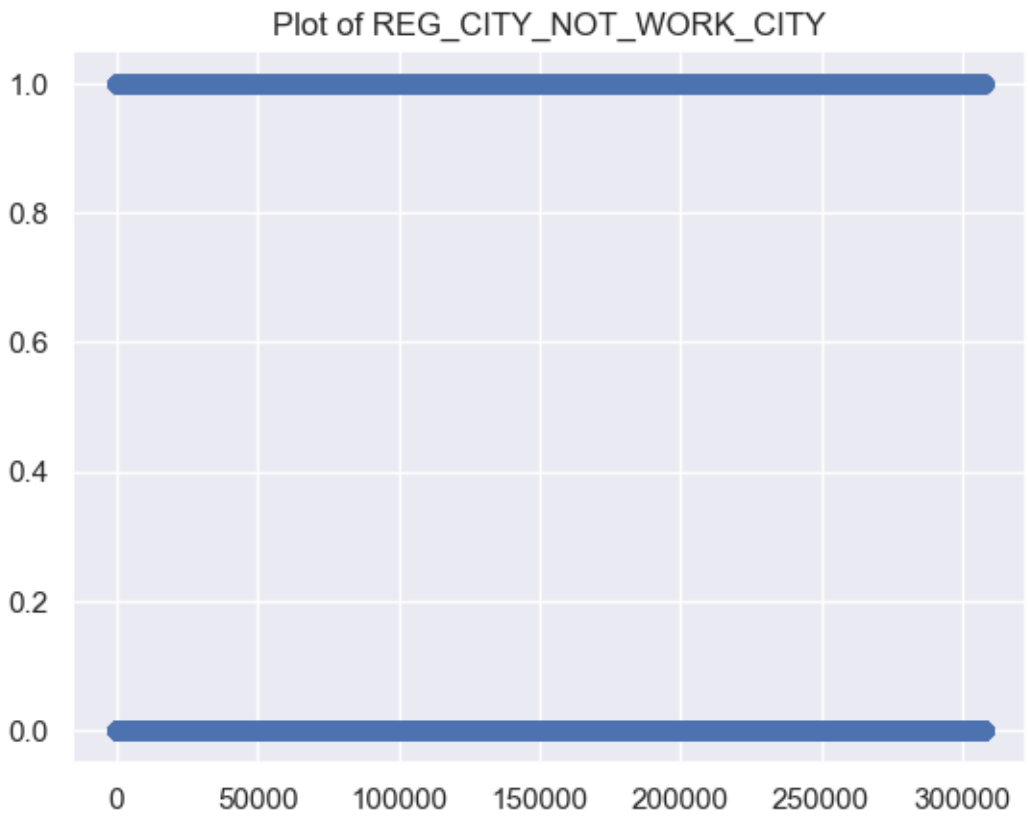

Plot of HOUR_APPR_PROCESS_START
20
15
10
5
0
0
50000
100000
150000
200000
250000
300000

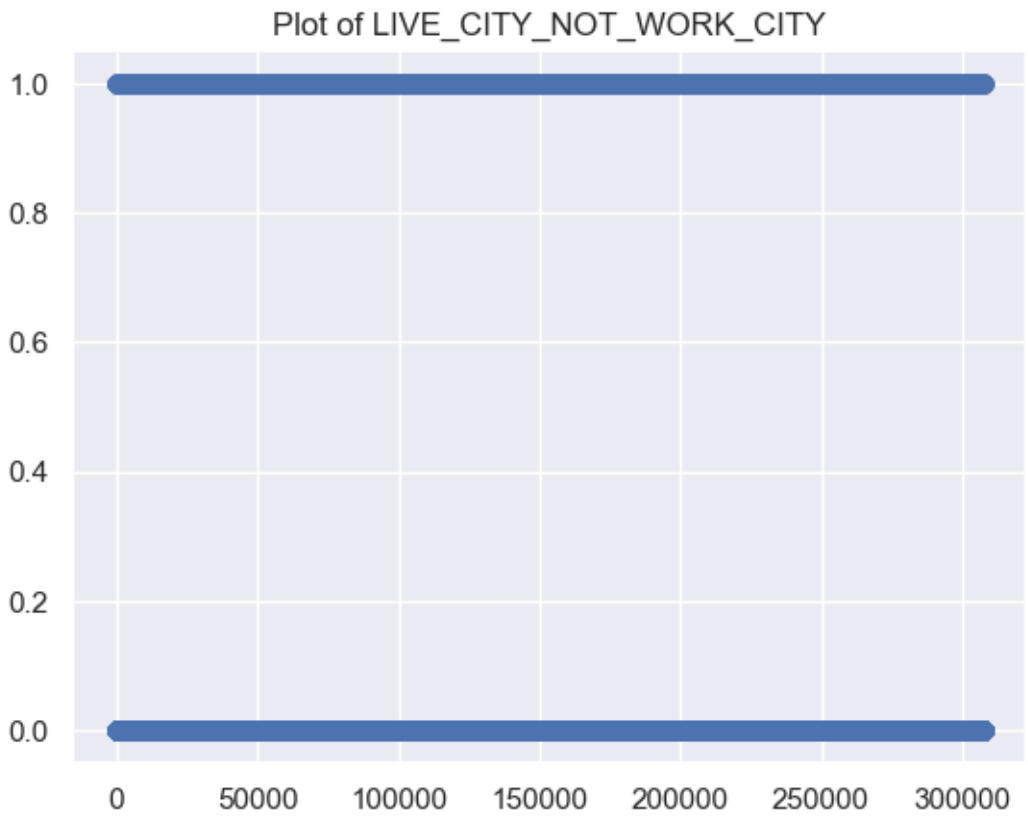

Plot of REG_REGION_NOT_LIVE_REGION
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

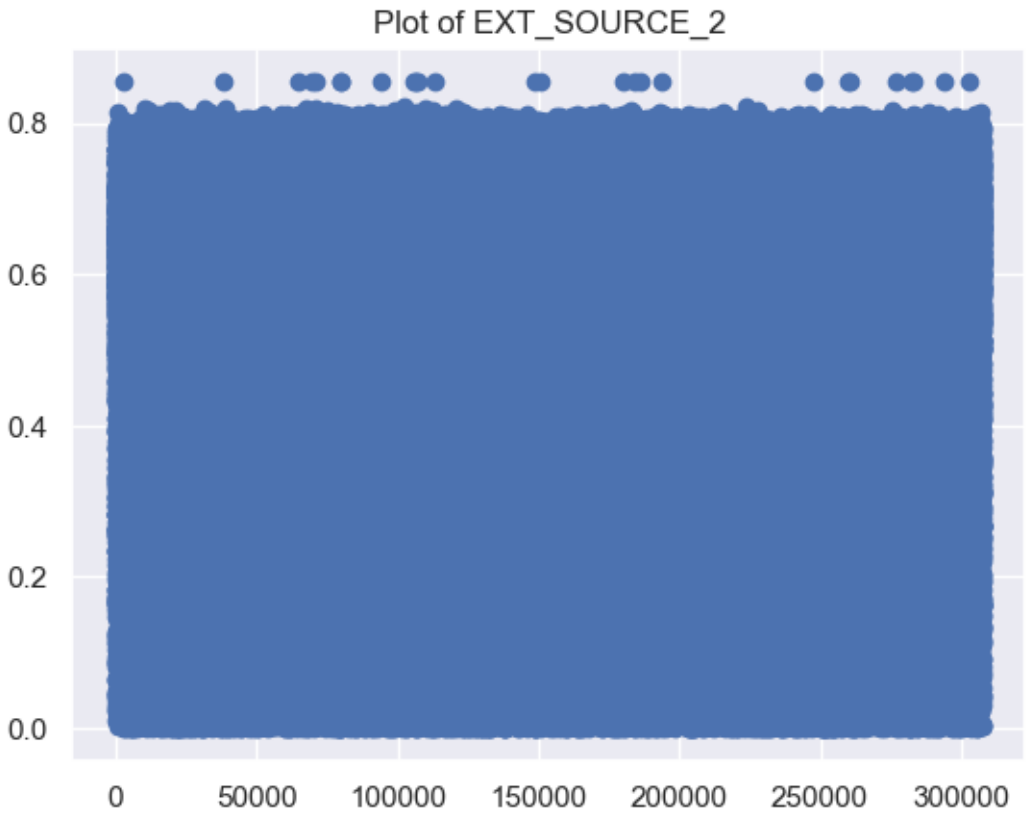

Plot of REG_REGION_NOT_WORK_REGION
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

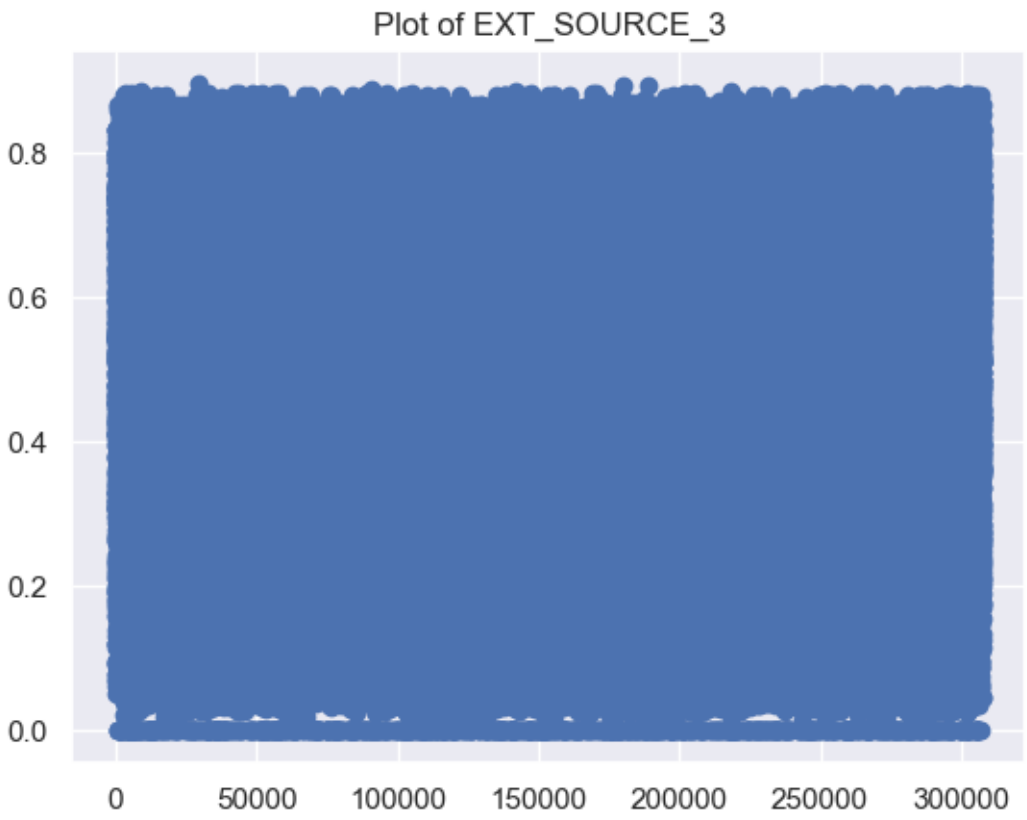

Plot of LIVE_REGION_NOT_WORK_REGION
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000


Plot of REG_CITY_NOT_LIVE_CITY
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

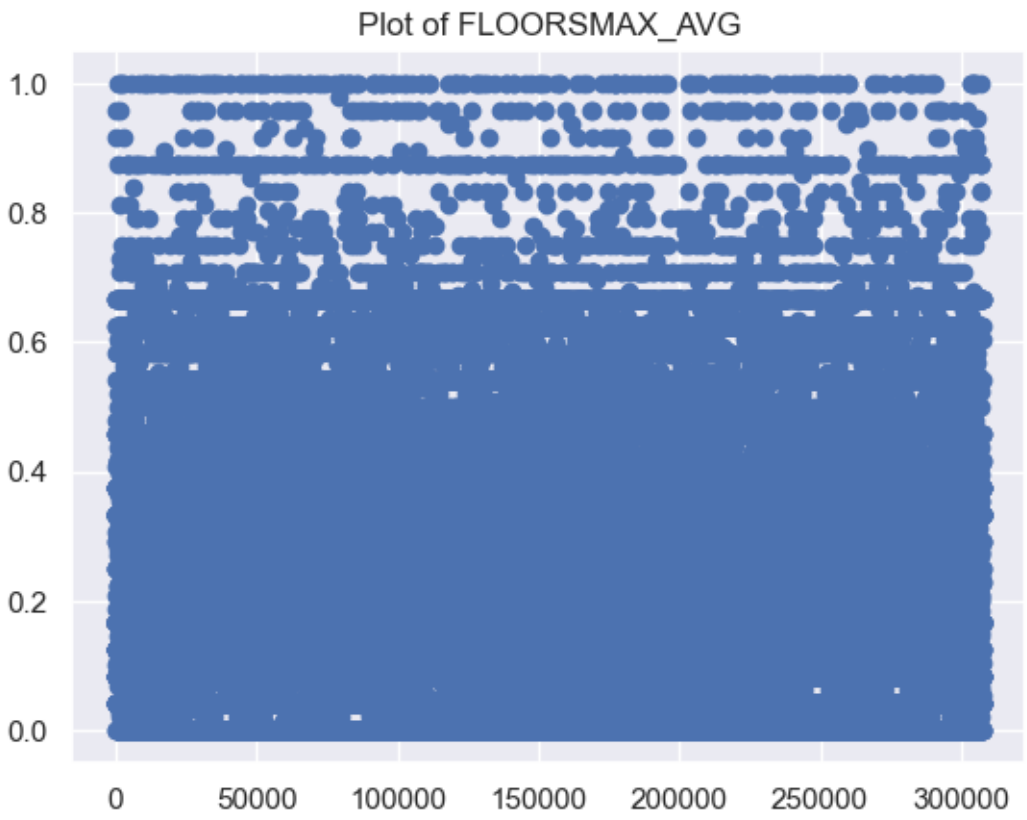

Plot of REG_CITY_NOT_WORK_CITY
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000


Plot of LIVE_CITY_NOT_WORK_CITY
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

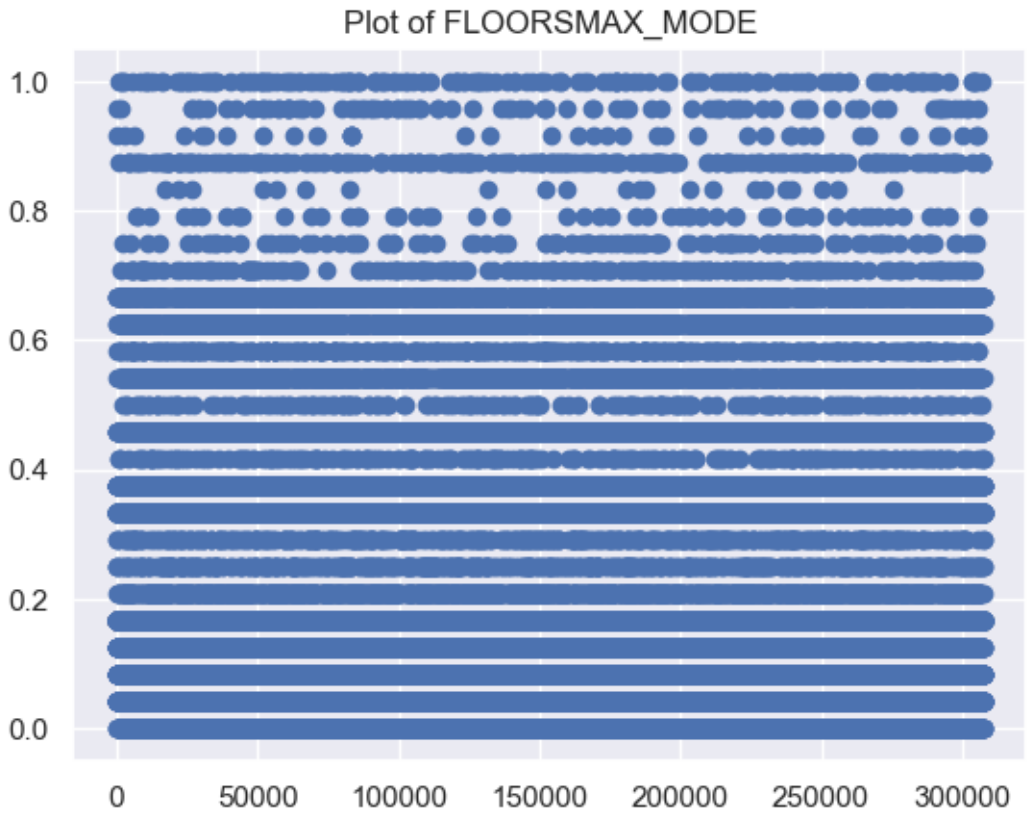

Plot of EXT_SOURCE_2
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

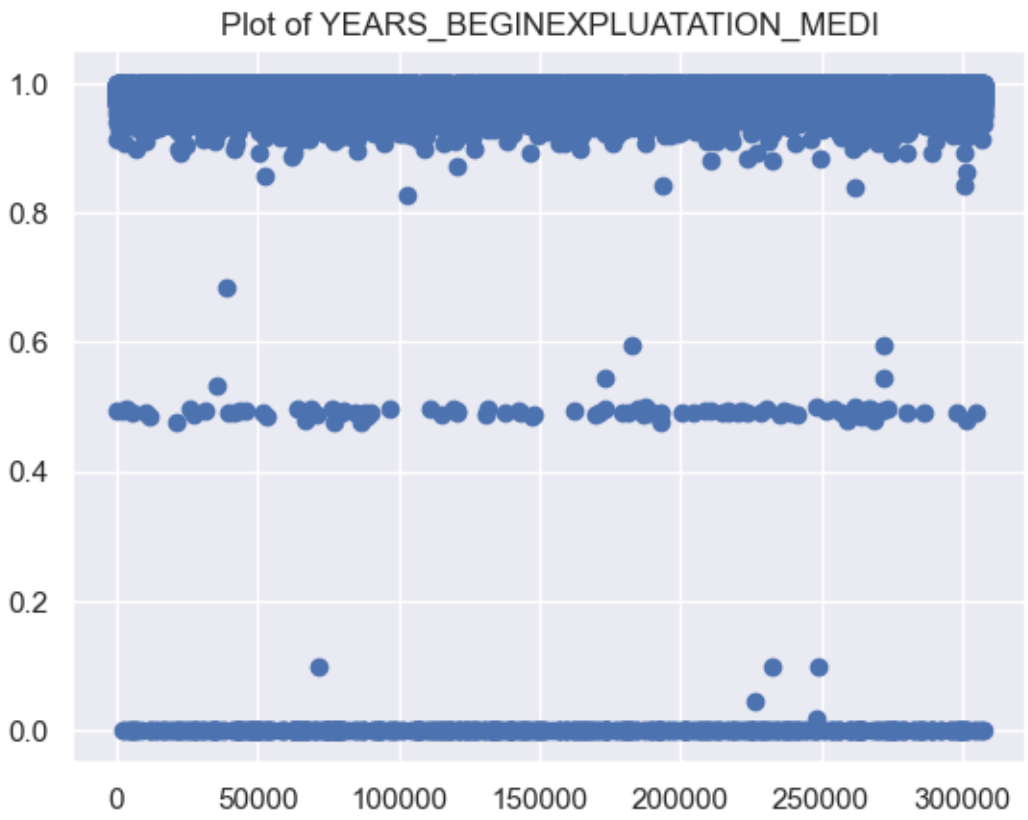

Plot of EXT_SOURCE_3
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

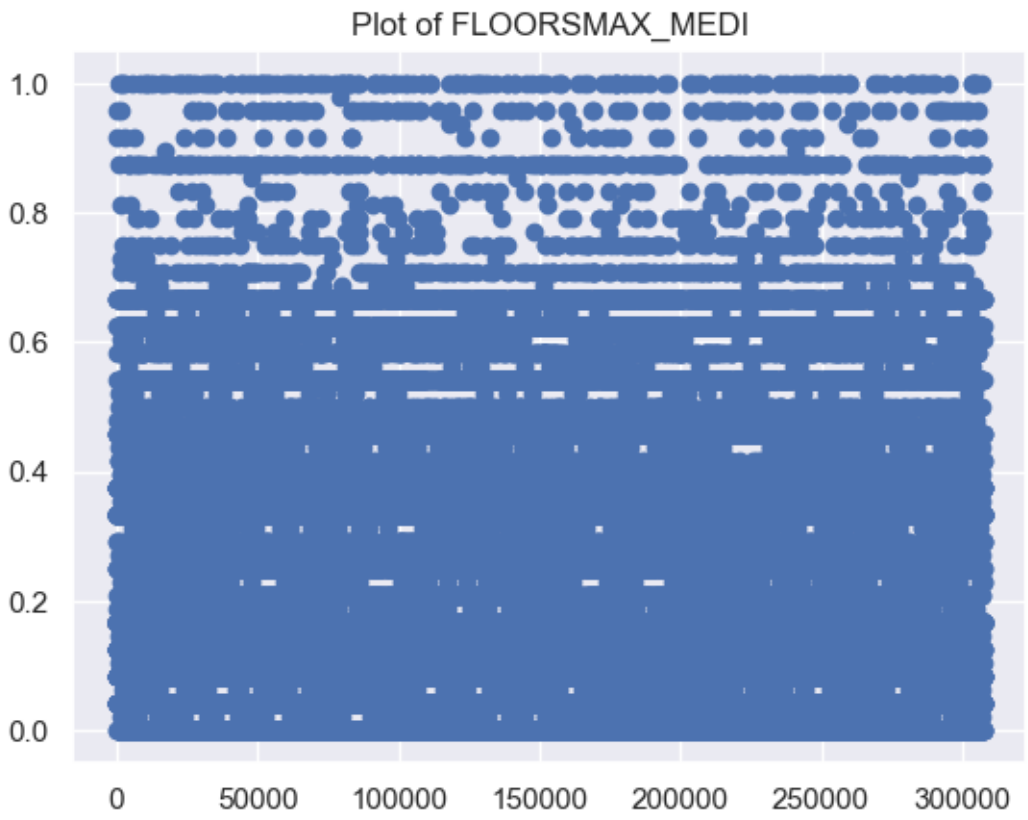

Plot of YEARS_BEGINEXPLUATATION_AVG
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

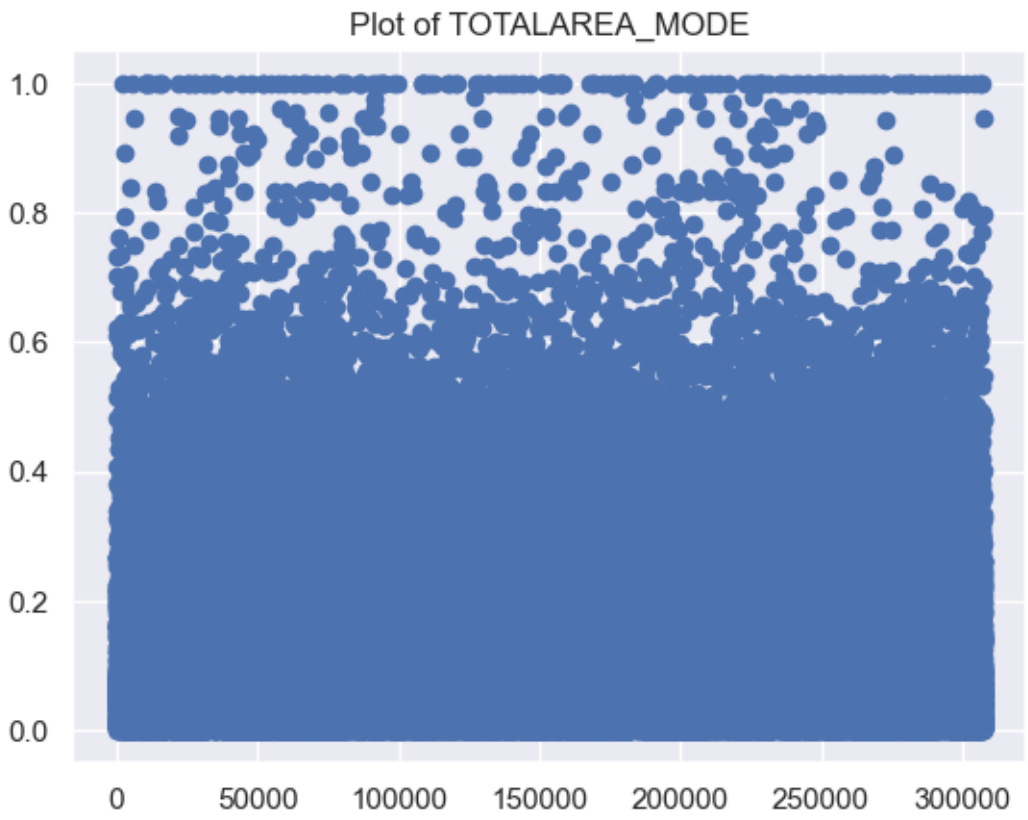

Plot of FLOORSMAX_AVG
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

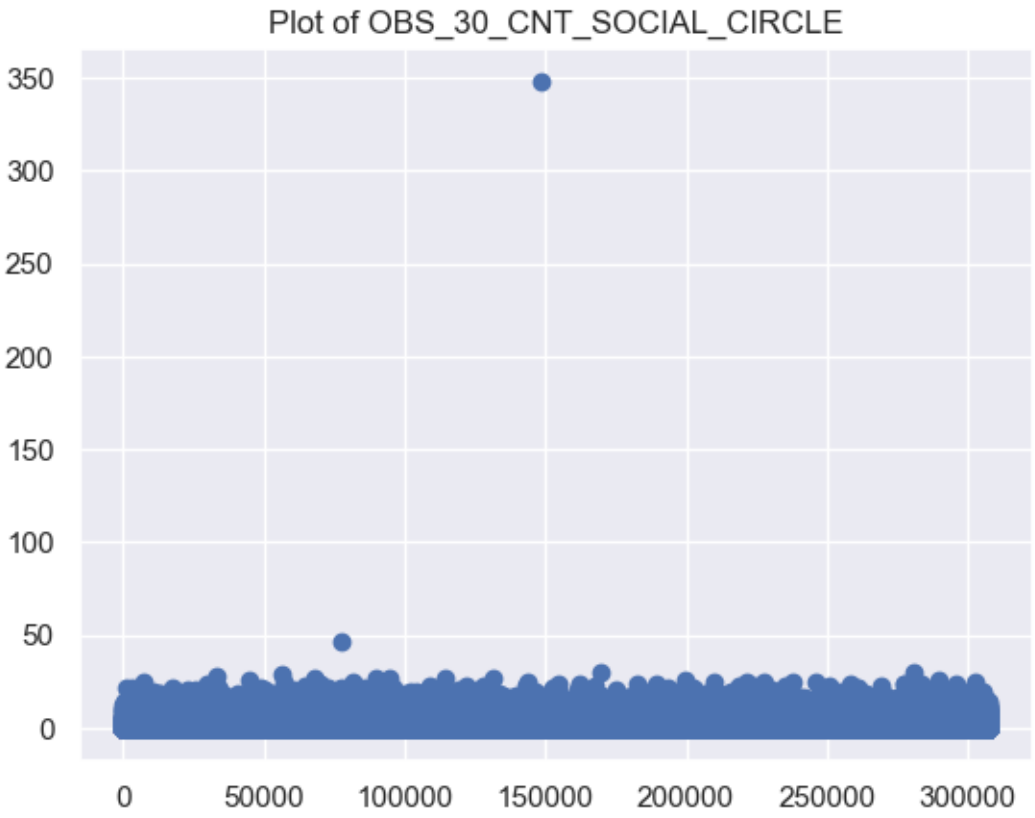

Plot of YEARS_BEGINEXPLUATATION_MODE
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

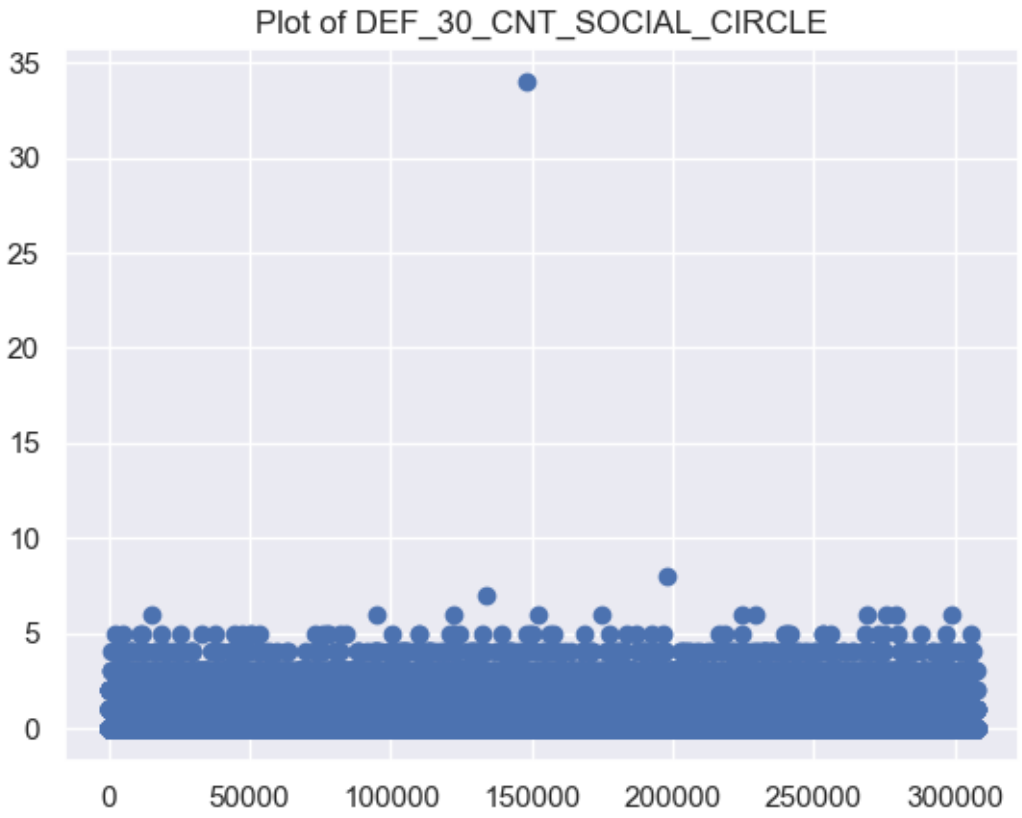

Plot of FLOORSMAX_MODE
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

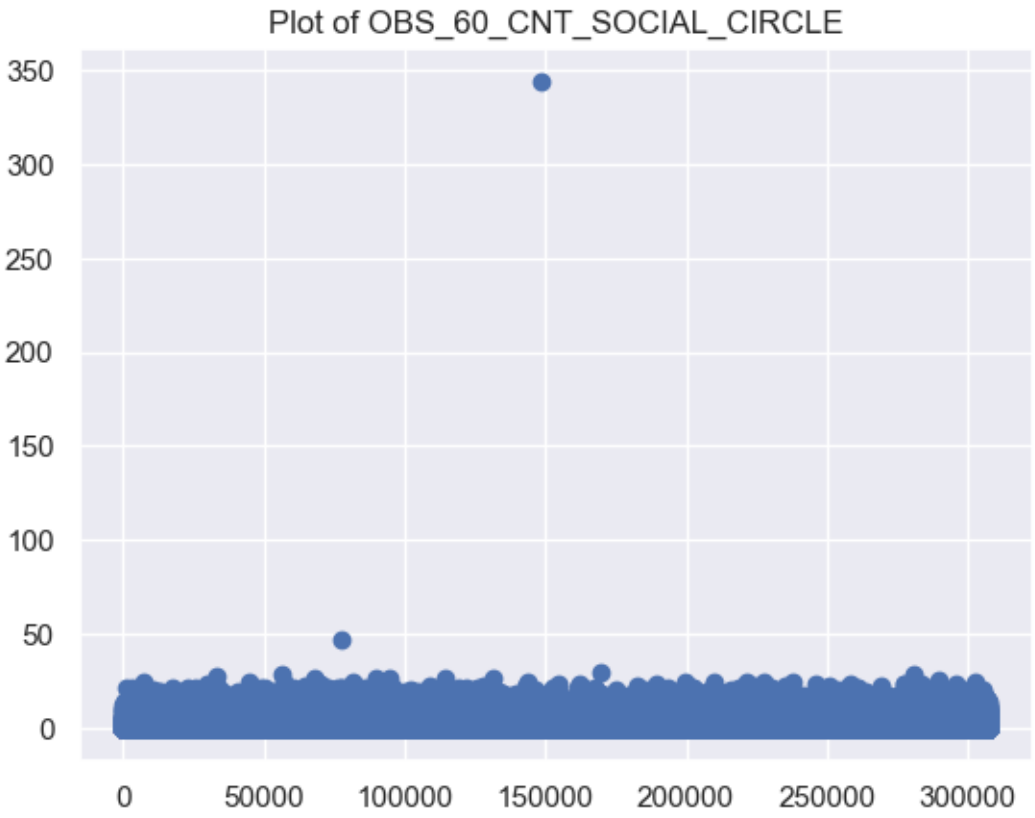

Plot of YEARS_BEGINEXPLUATATION_MEDI
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

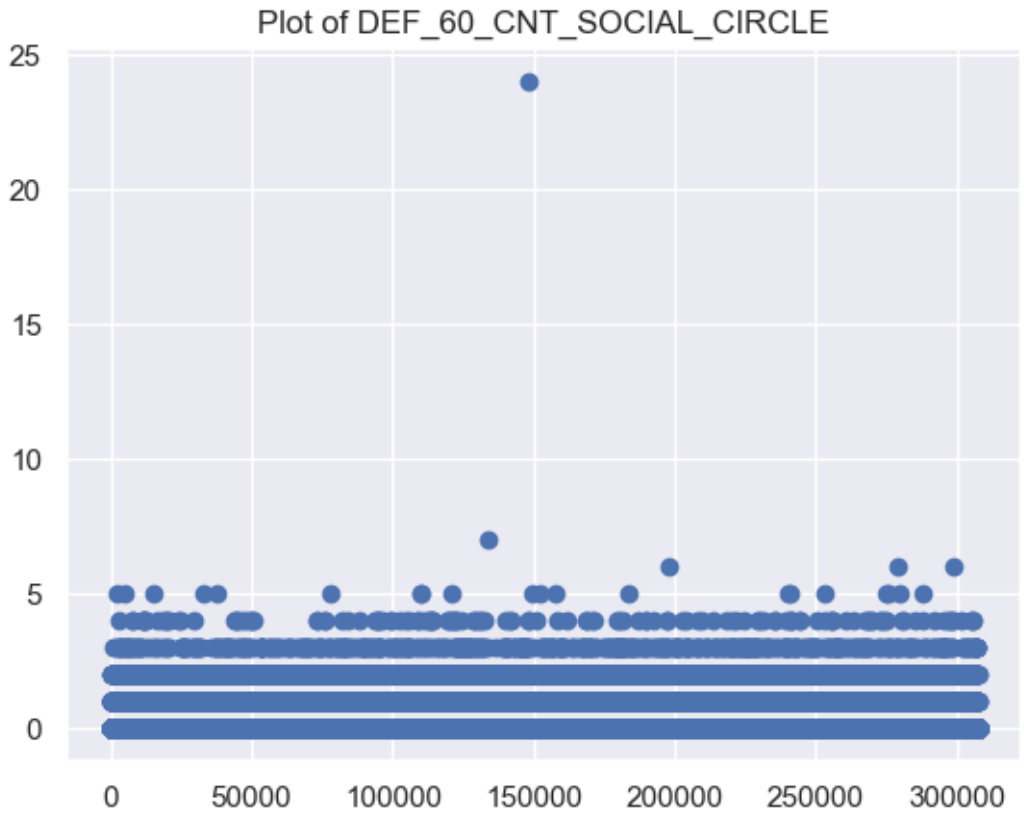

Plot of FLOORSMAX_MEDI
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

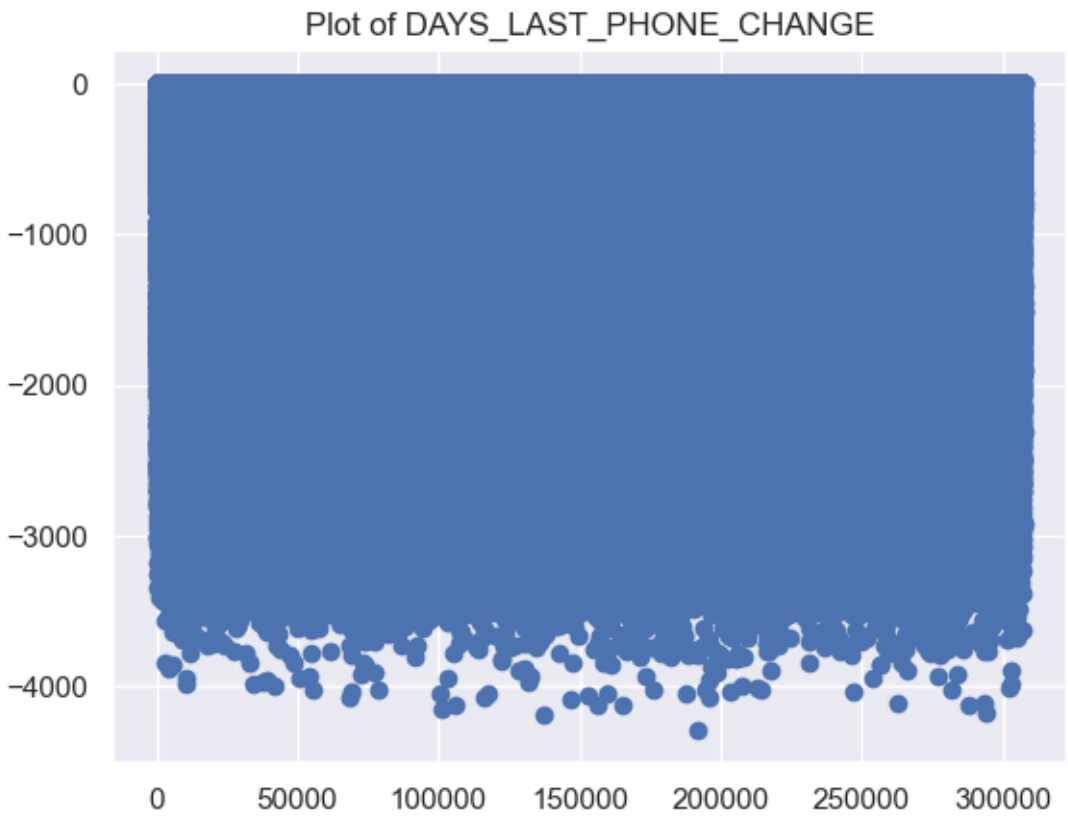

Plot of TOTALAREA_MODE
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

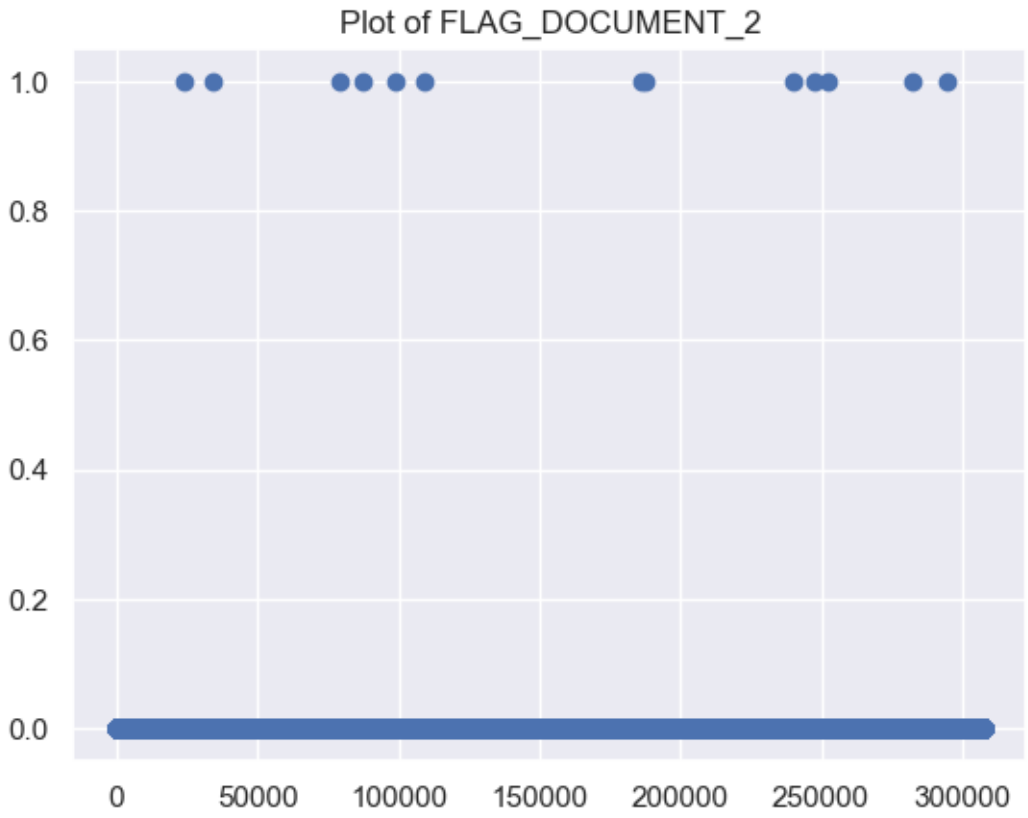

Plot of OBS_30_CNT_SOCIAL_CIRCLE
350
300
250
200
150
100
50
0
0
50000
100000
150000
200000
250000
300000

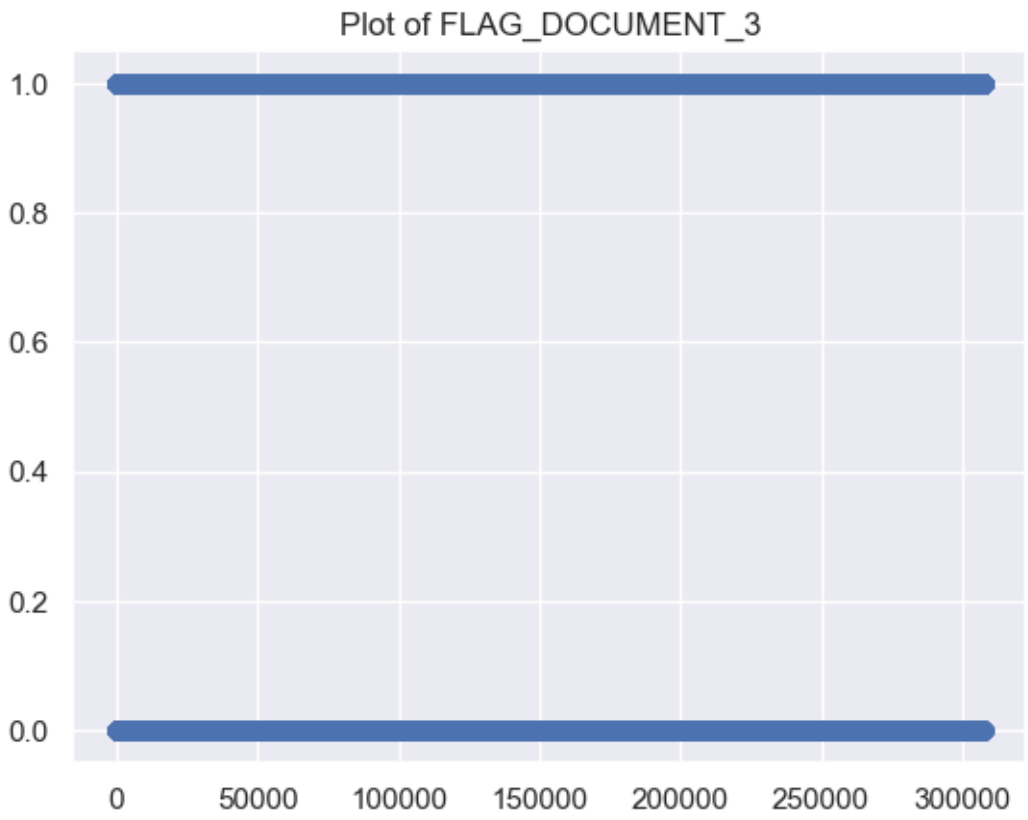

Plot of DEF_30_CNT_SOCIAL_CIRCLE
35
30
25
20
15
10
5
0
0
50000
100000
150000
200000
250000
300000


Plot of OBS_60_CNT_SOCIAL_CIRCLE
350
300
250
200
150
100
50
0
0
50000
100000
150000
200000
250000
300000

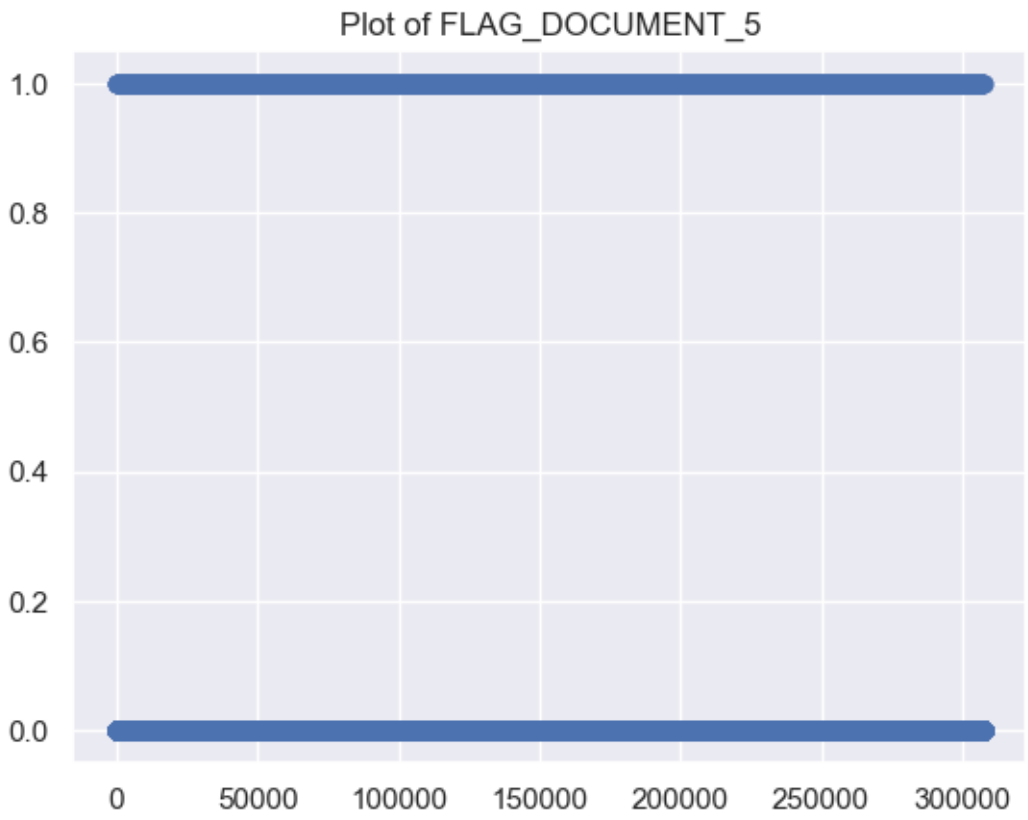

Plot of DEF_60_CNT_SOCIAL_CIRCLE
25
20
15
10
5
0
0
50000
100000
150000
200000
250000
300000

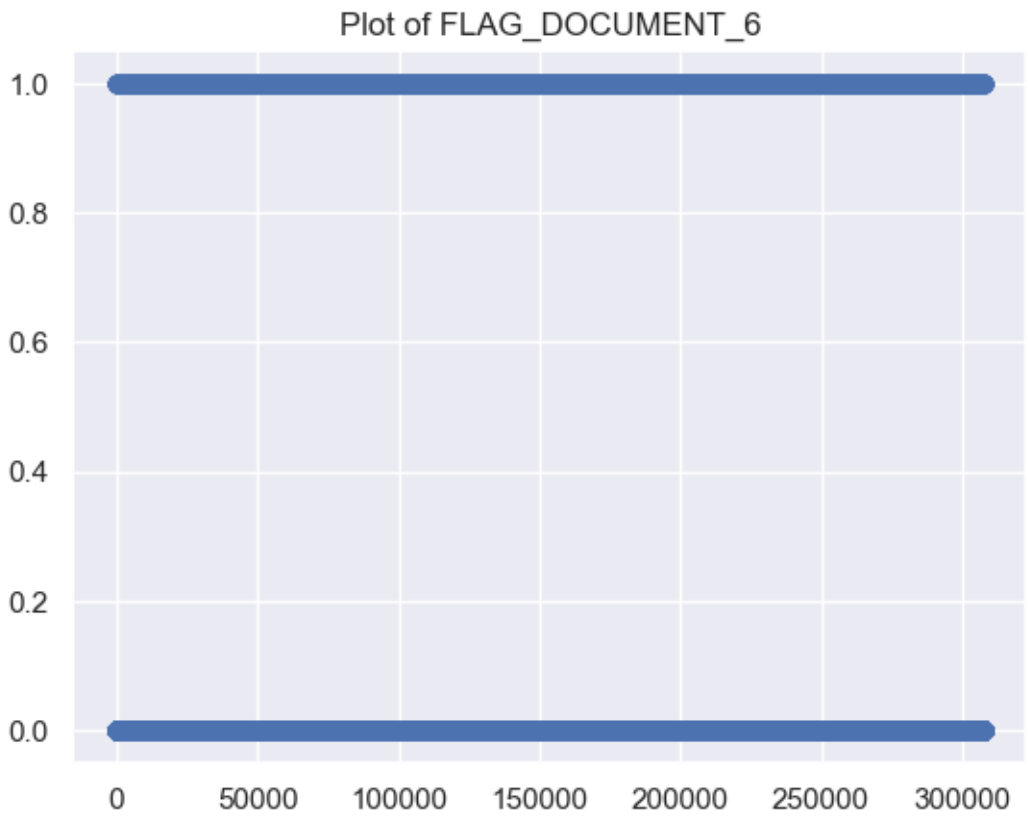

Plot of DAYS_LAST_PHONE_CHANGE
0
−1000
−2000
−3000
−4000
0
50000
100000
150000
200000
250000
300000

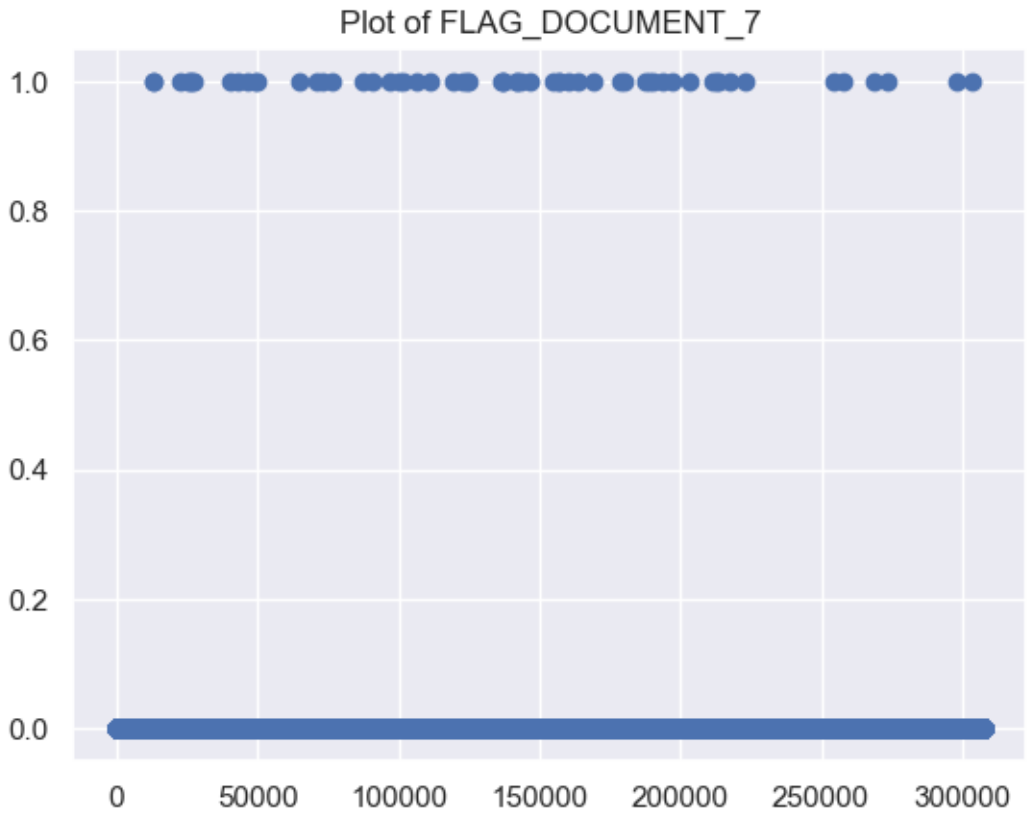

Plot of FLAG_DOCUMENT_2
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

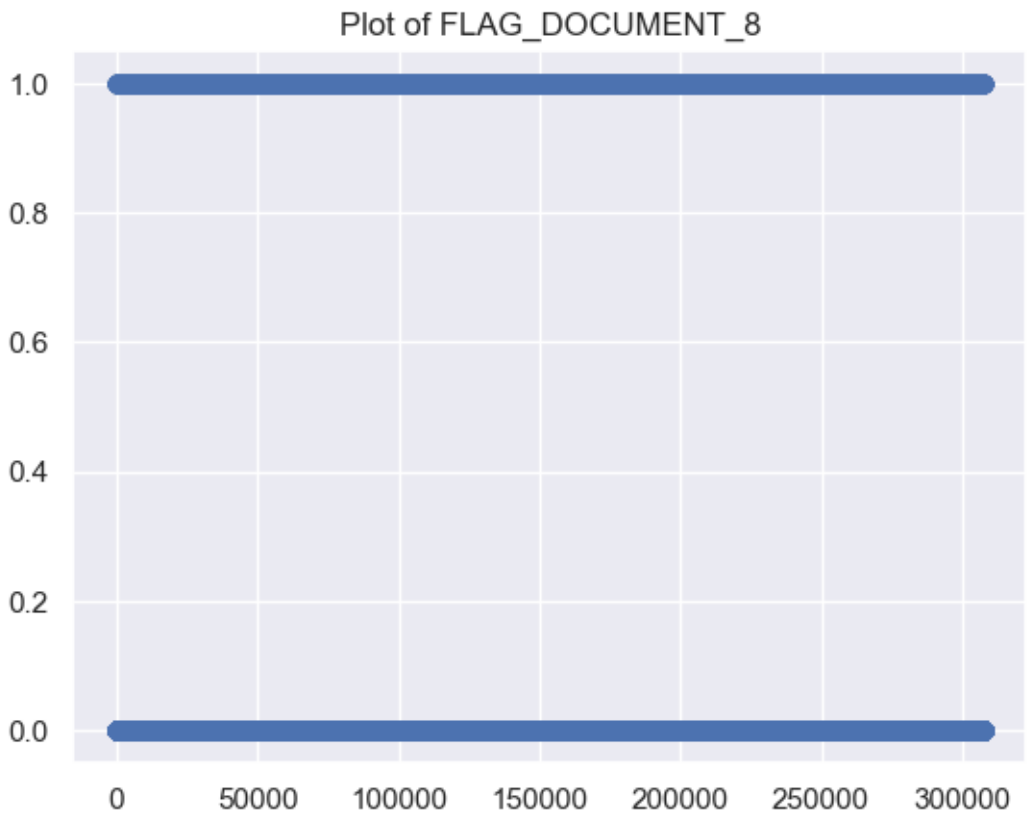

Plot of FLAG_DOCUMENT_3
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

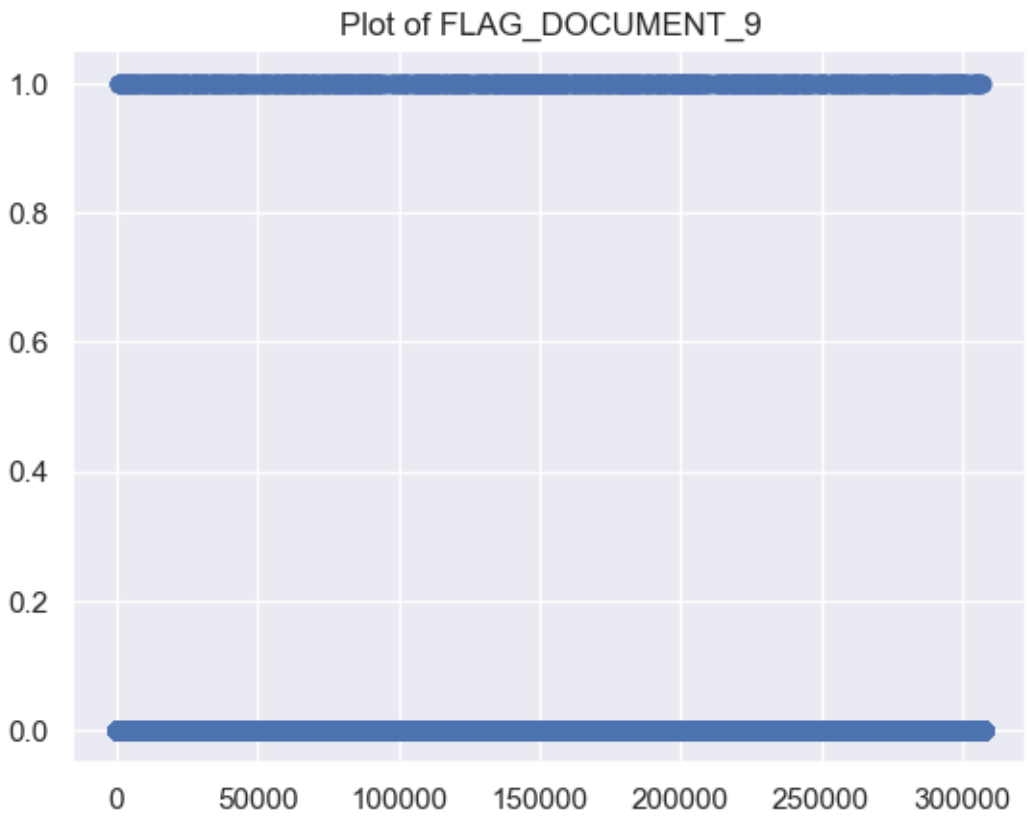

Plot of FLAG_DOCUMENT_4
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

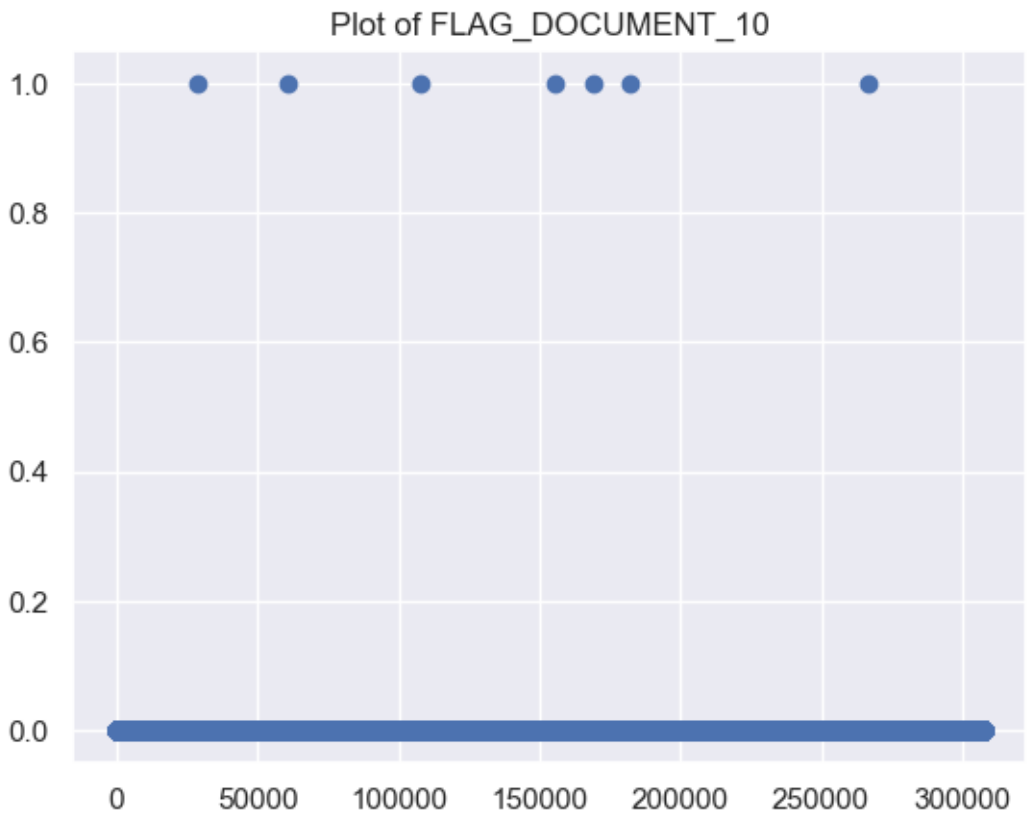

Plot of FLAG_DOCUMENT_5
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

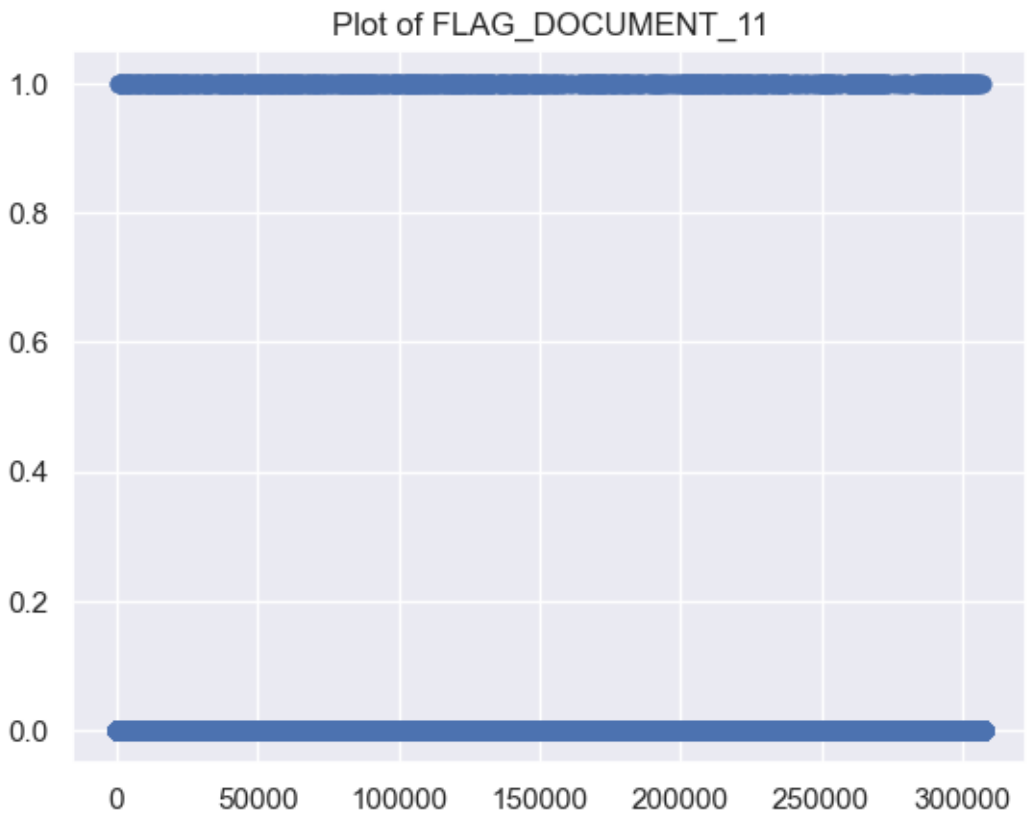

Plot of FLAG_DOCUMENT_6
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000


Plot of FLAG_DOCUMENT_7
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

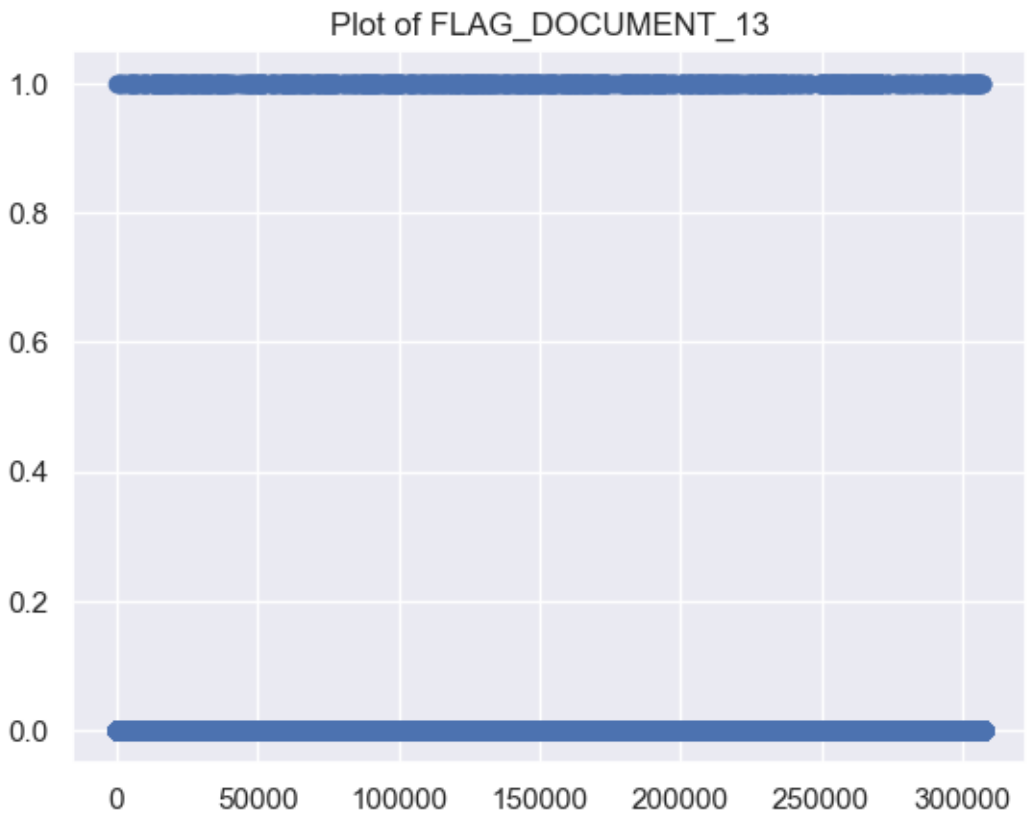

Plot of FLAG_DOCUMENT_8
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

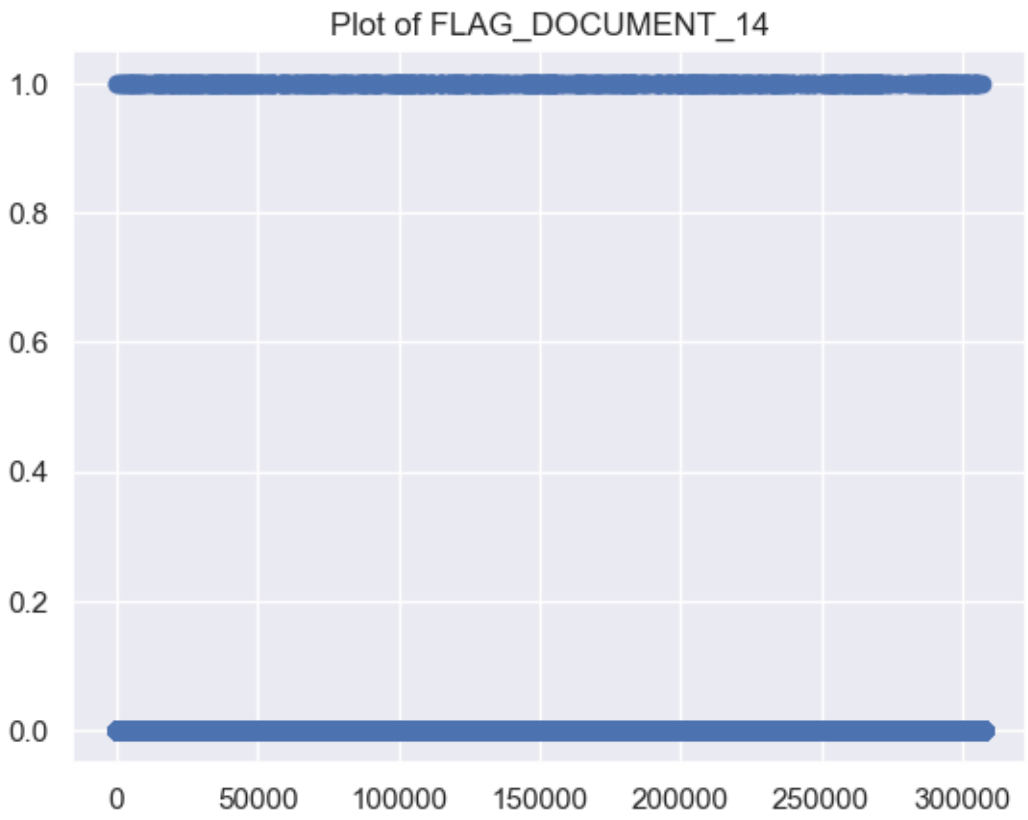

Plot of FLAG_DOCUMENT_9
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

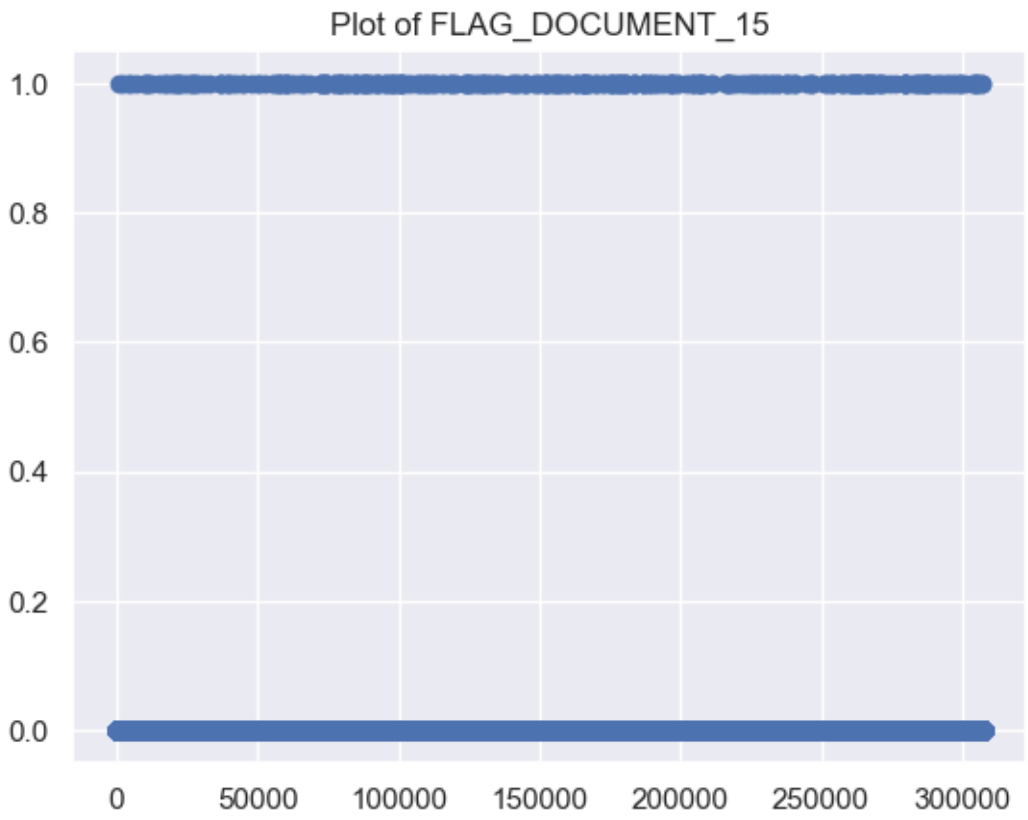

Plot of FLAG_DOCUMENT_10
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

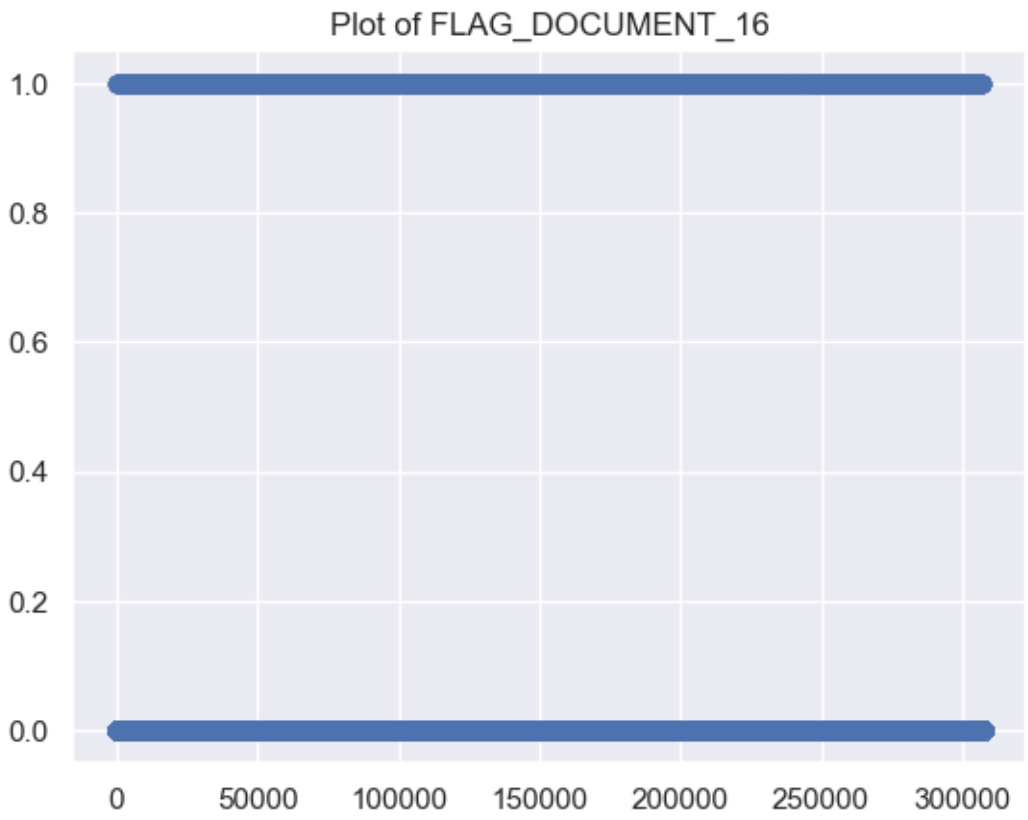

Plot of FLAG_DOCUMENT_11
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

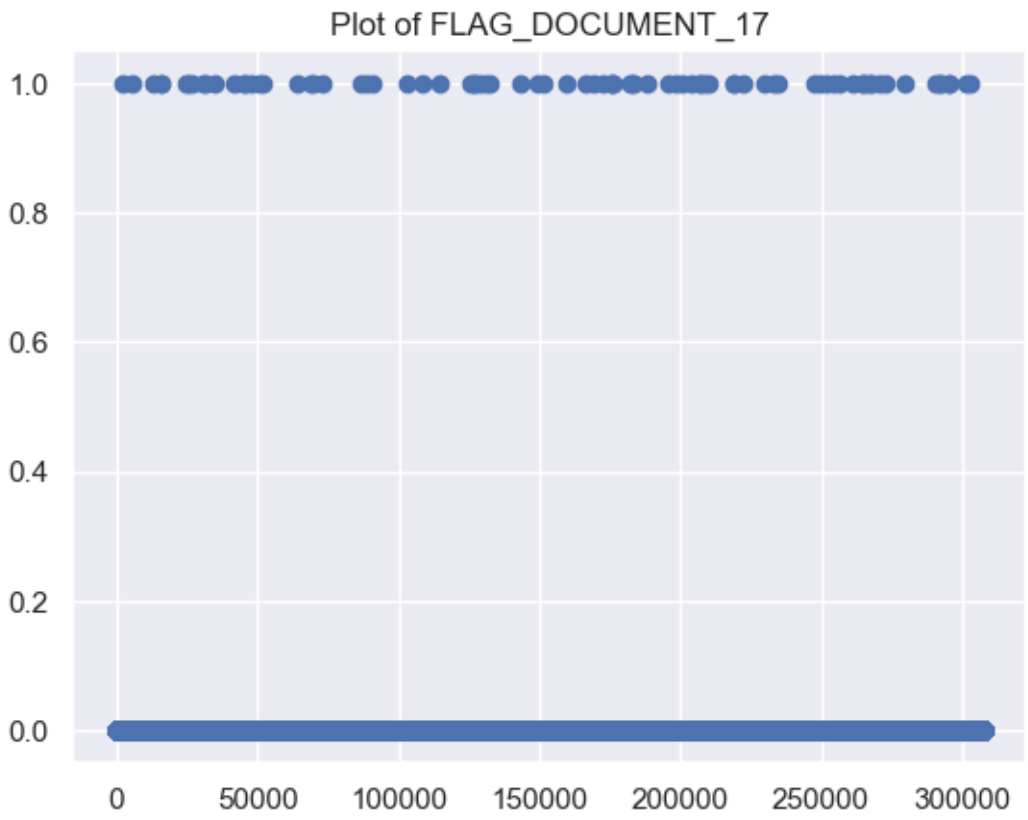

Plot of FLAG_DOCUMENT_12
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

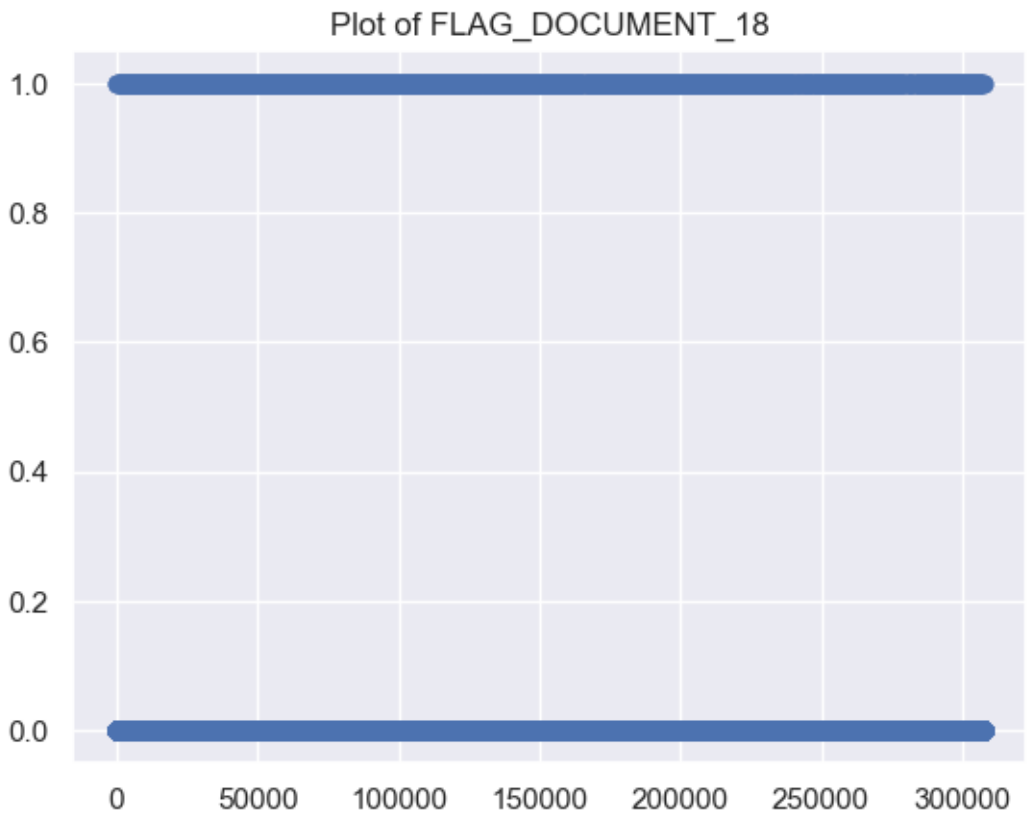

Plot of FLAG_DOCUMENT_13
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

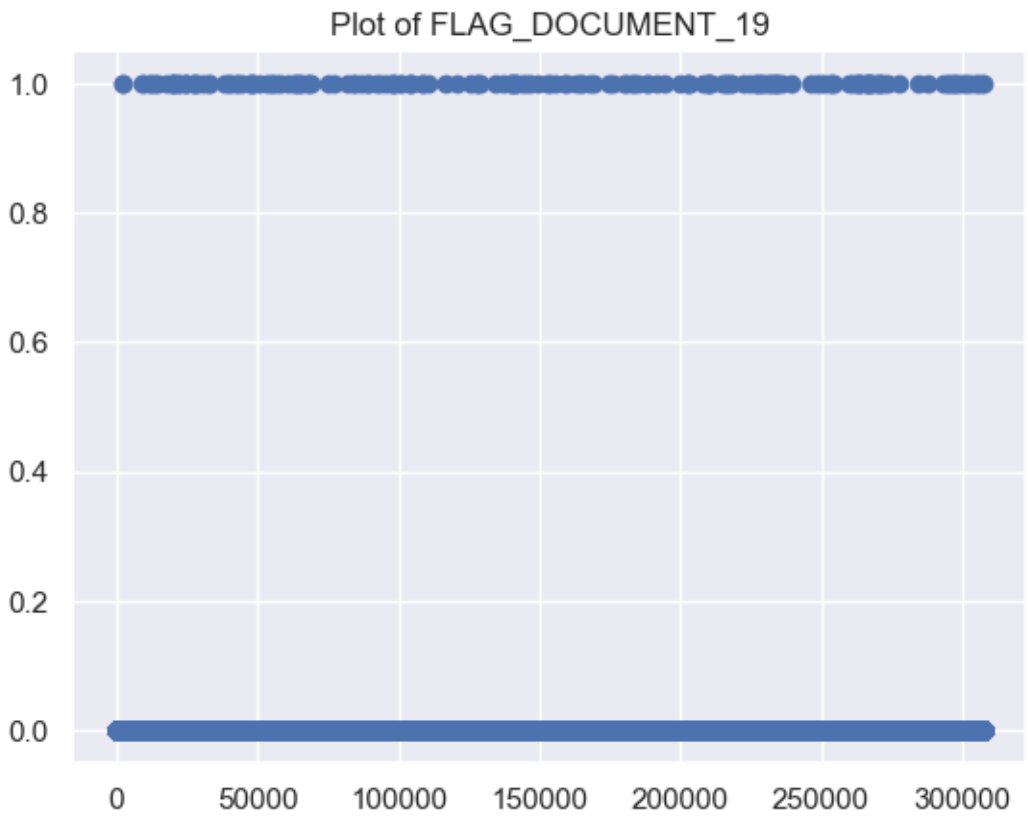

Plot of FLAG_DOCUMENT_14
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

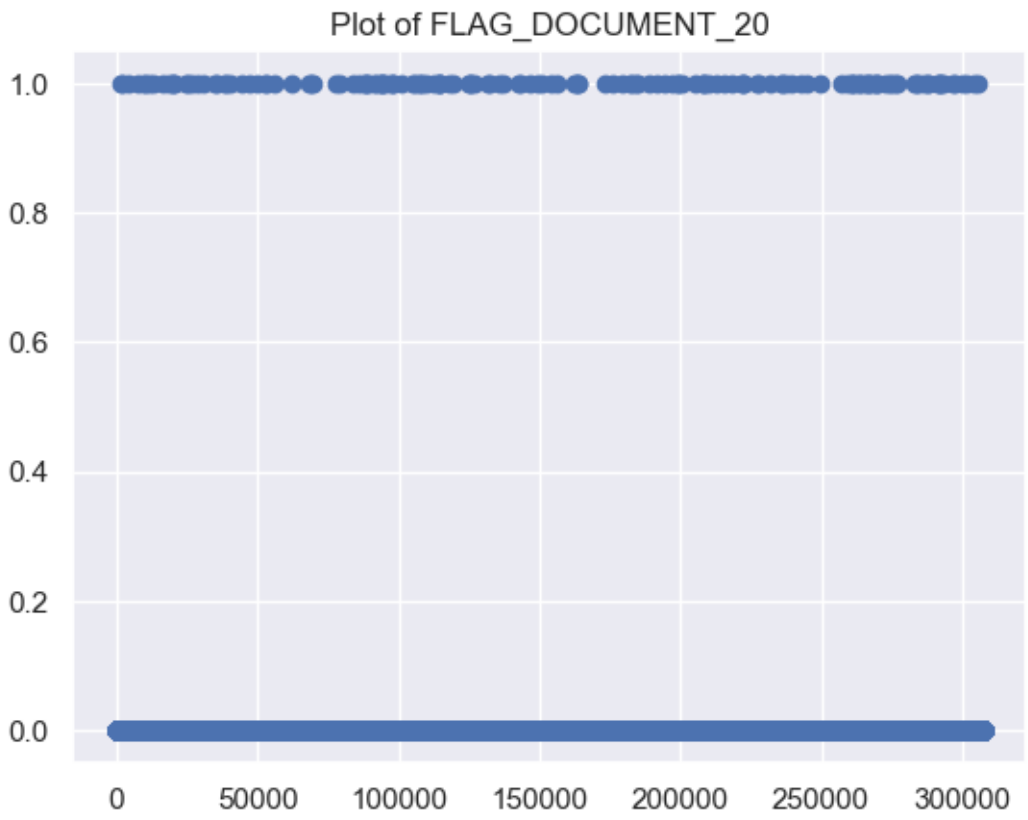

Plot of FLAG_DOCUMENT_15
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

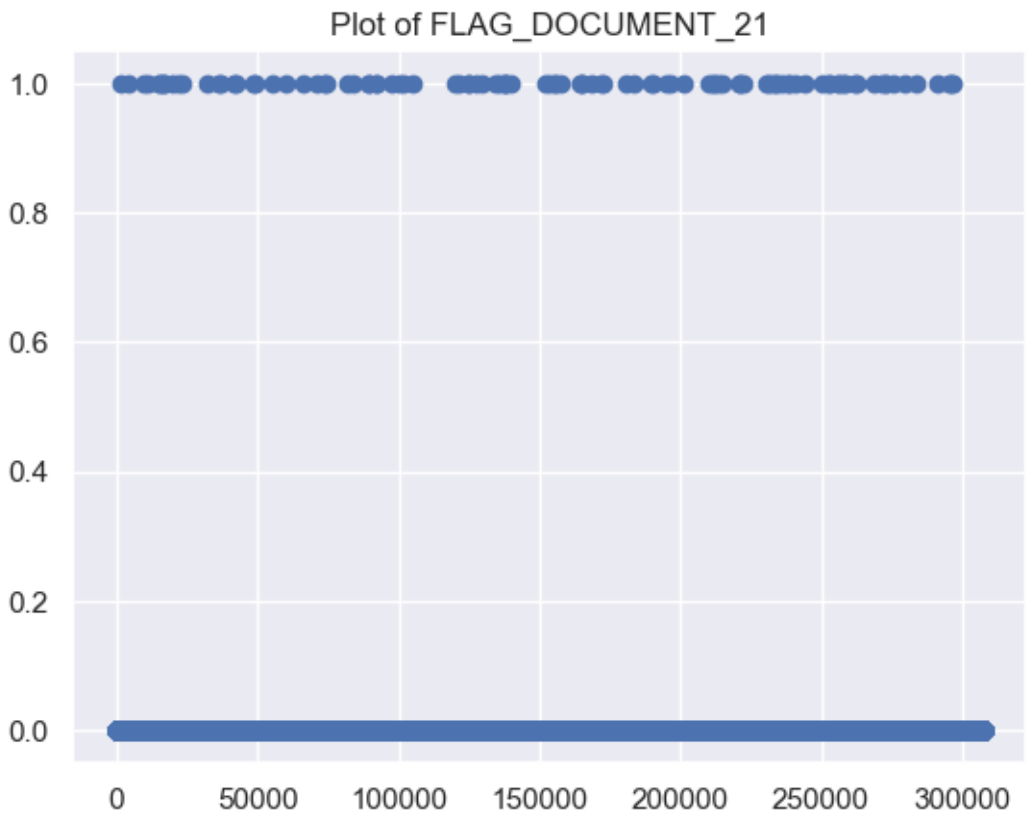

Plot of FLAG_DOCUMENT_16
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

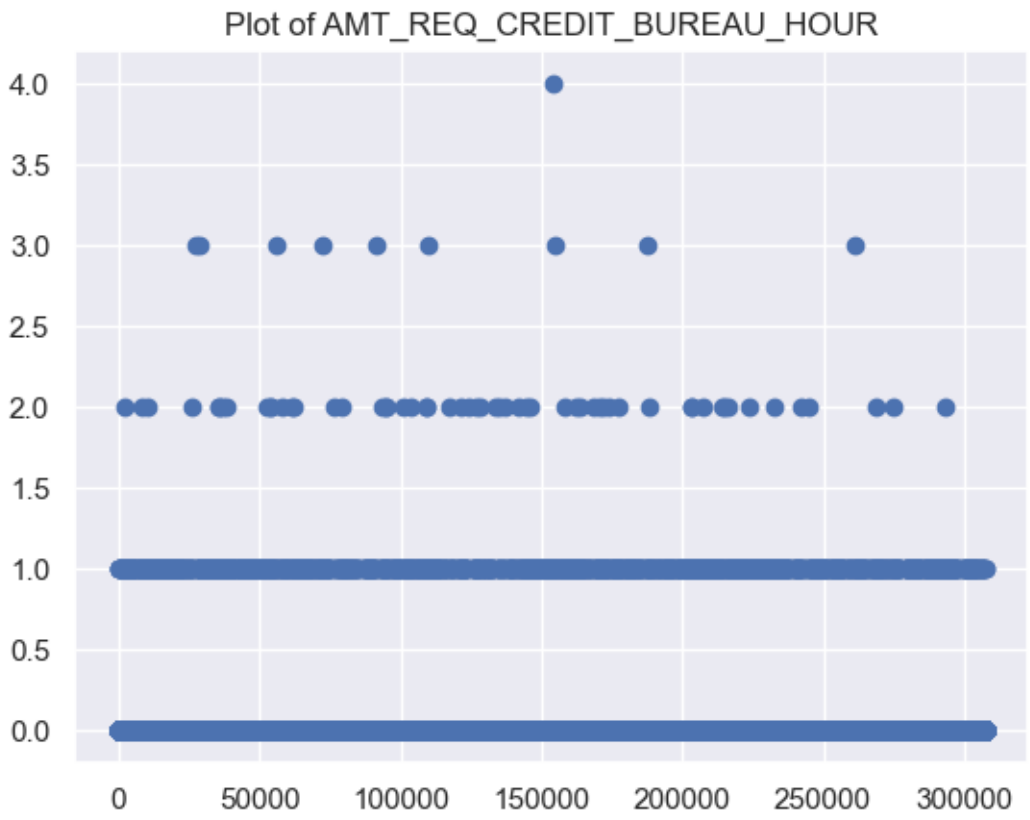

Plot of FLAG_DOCUMENT_17
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

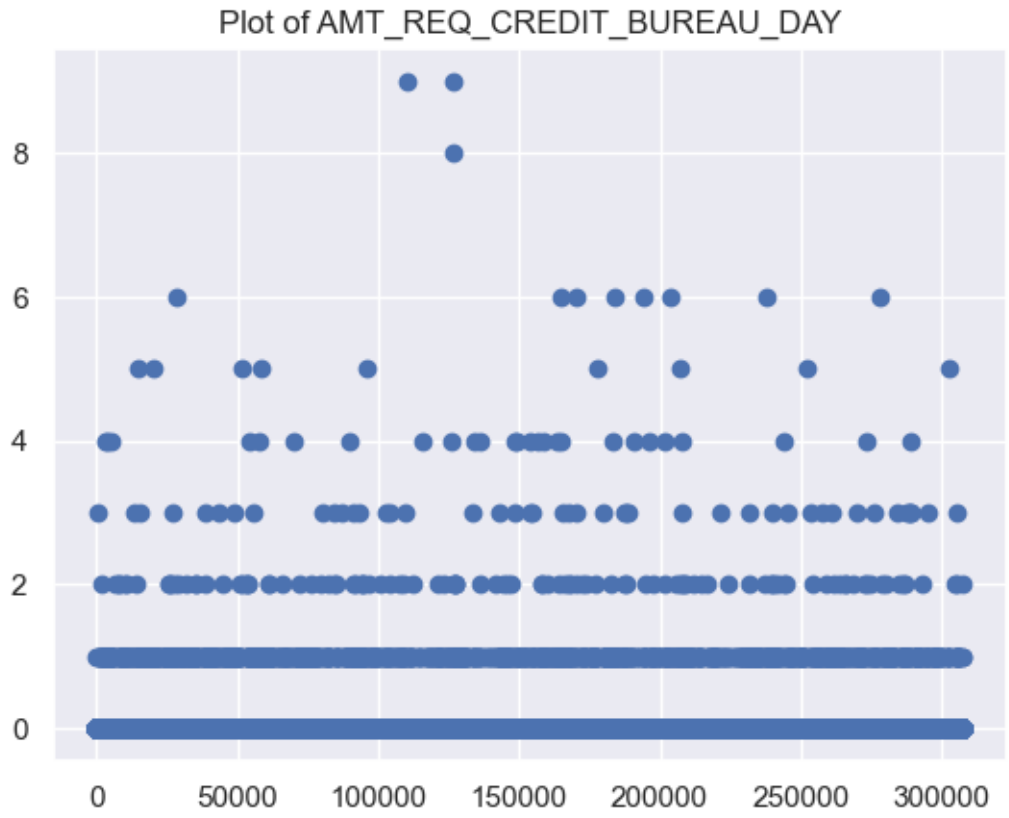

Plot of FLAG_DOCUMENT_18
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

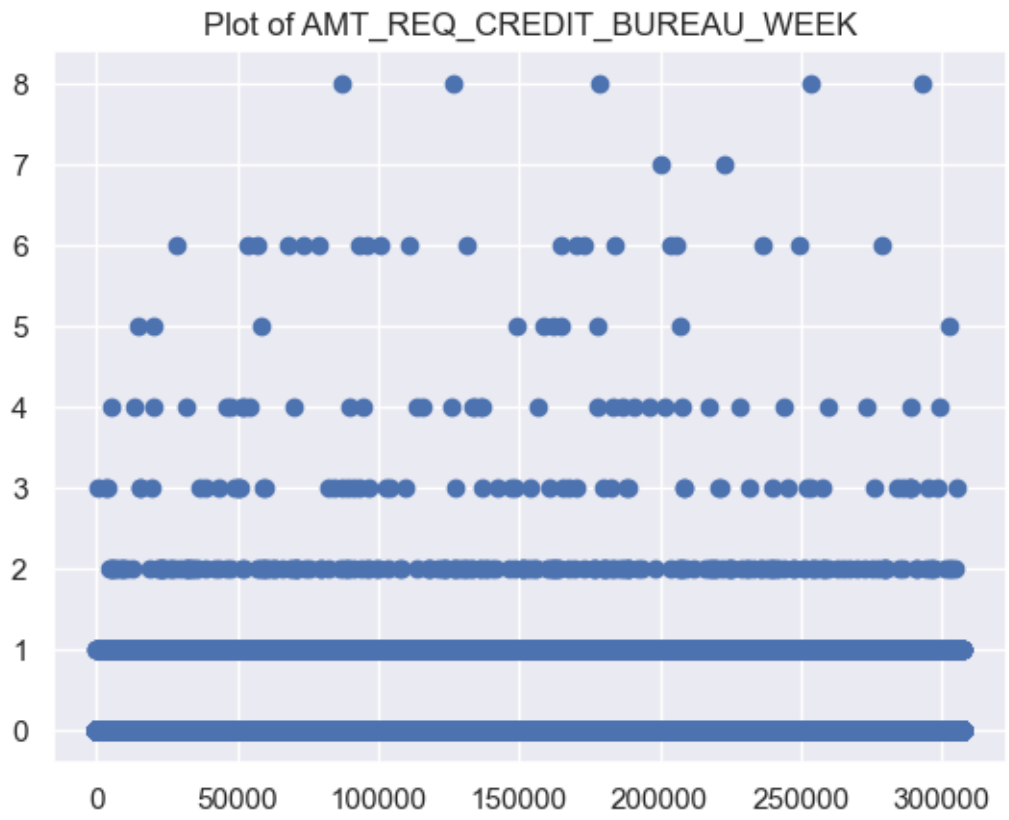

Plot of FLAG_DOCUMENT_19
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

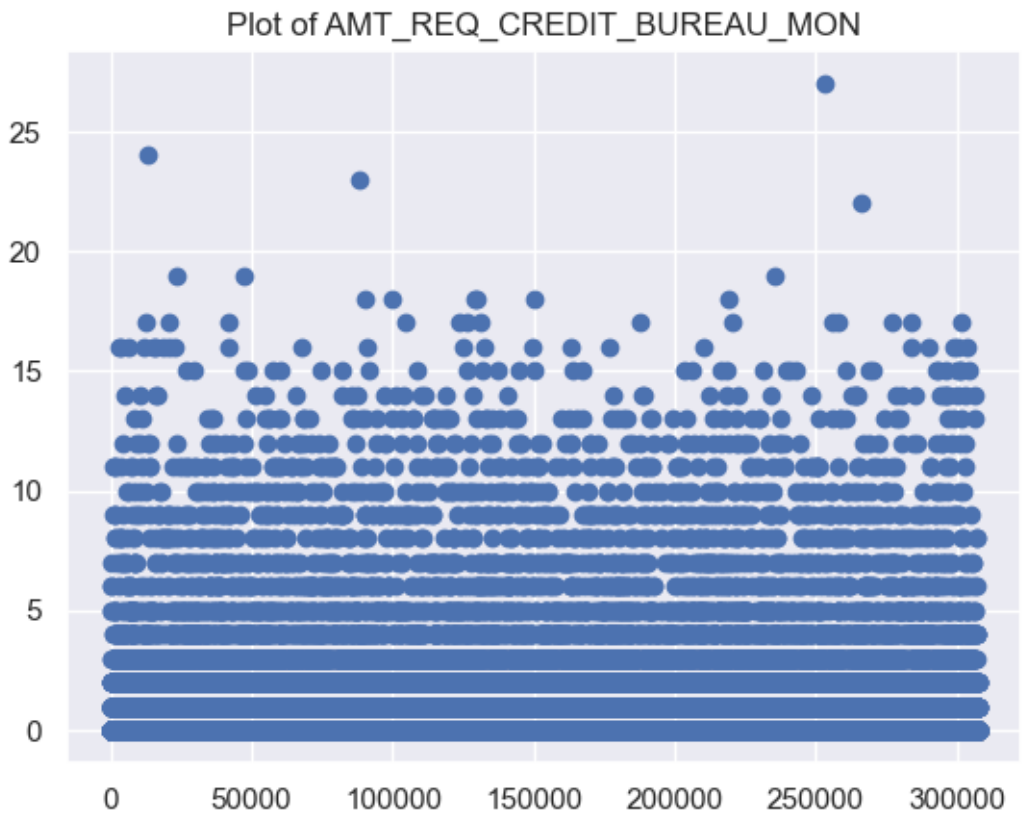

Plot of FLAG_DOCUMENT_20
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

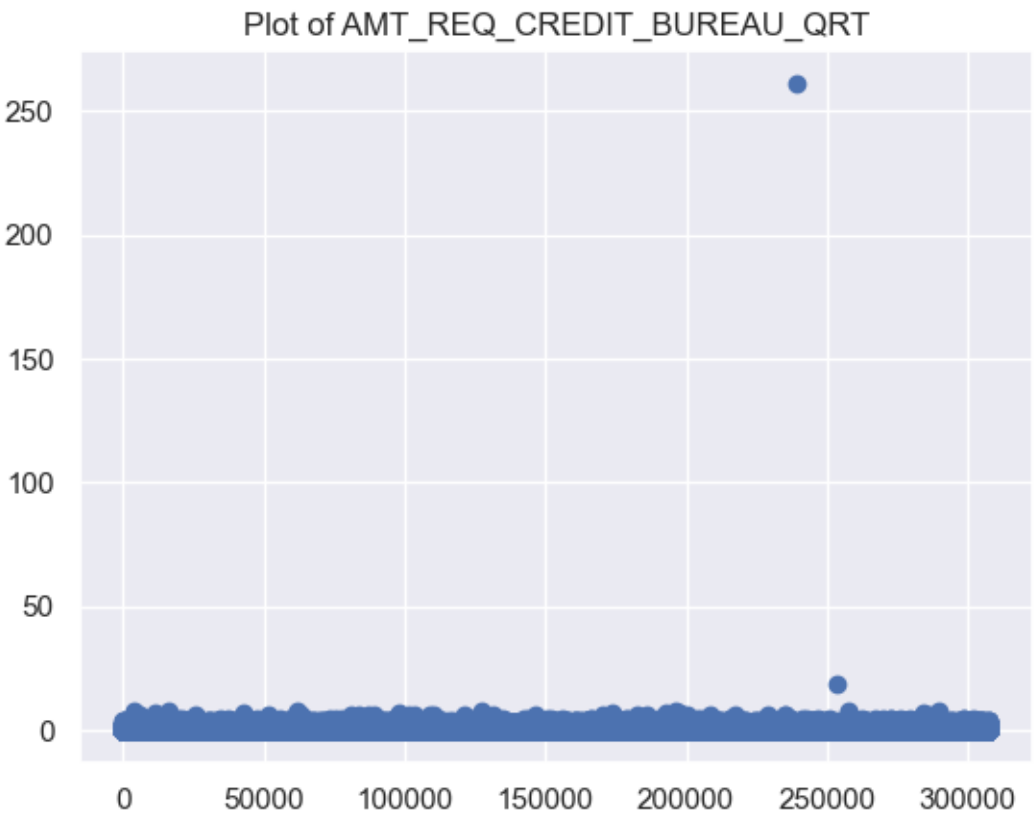

Plot of FLAG_DOCUMENT_21
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

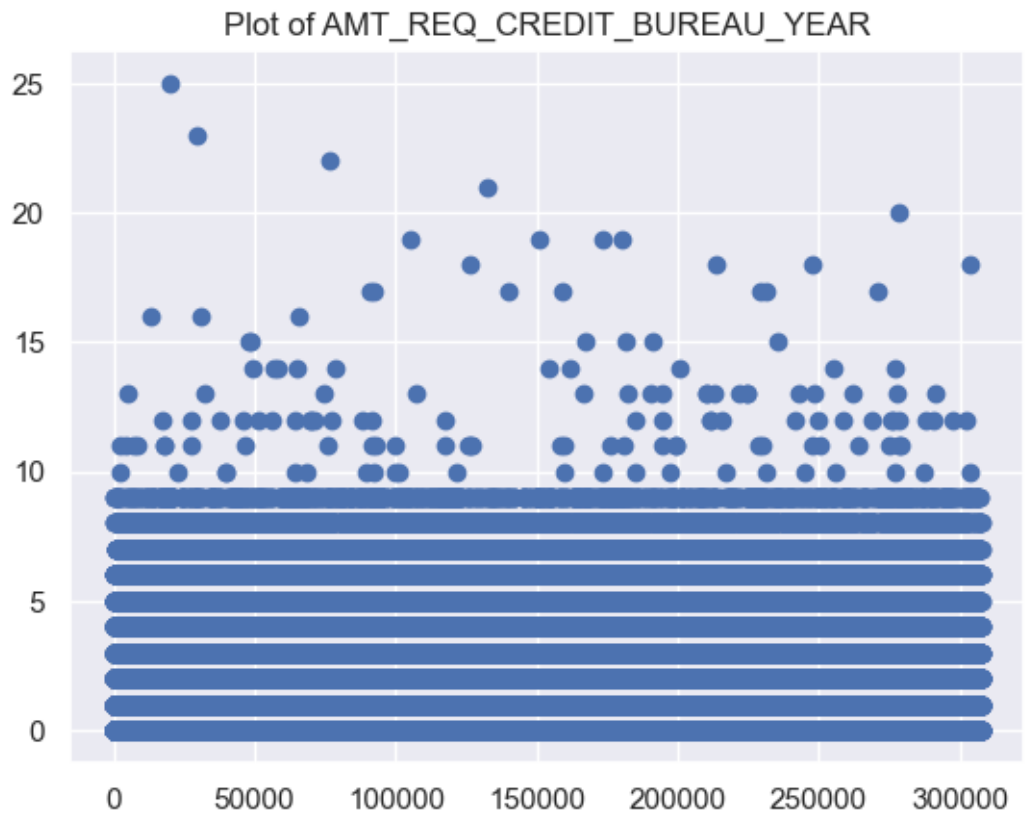

Plot of AMT_REQ_CREDIT_BUREAU_HOUR
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50000
100000
150000
200000
250000
300000


Plot of AMT_REQ_CREDIT_BUREAU_DAY
8
6
4
2
0
0
50000
100000
150000
200000
250000
300000


Plot of AMT_REQ_CREDIT_BUREAU_WEEK
8
7
6
5
4
3
2
1
0
0
50000
100000
150000
200000
250000
300000

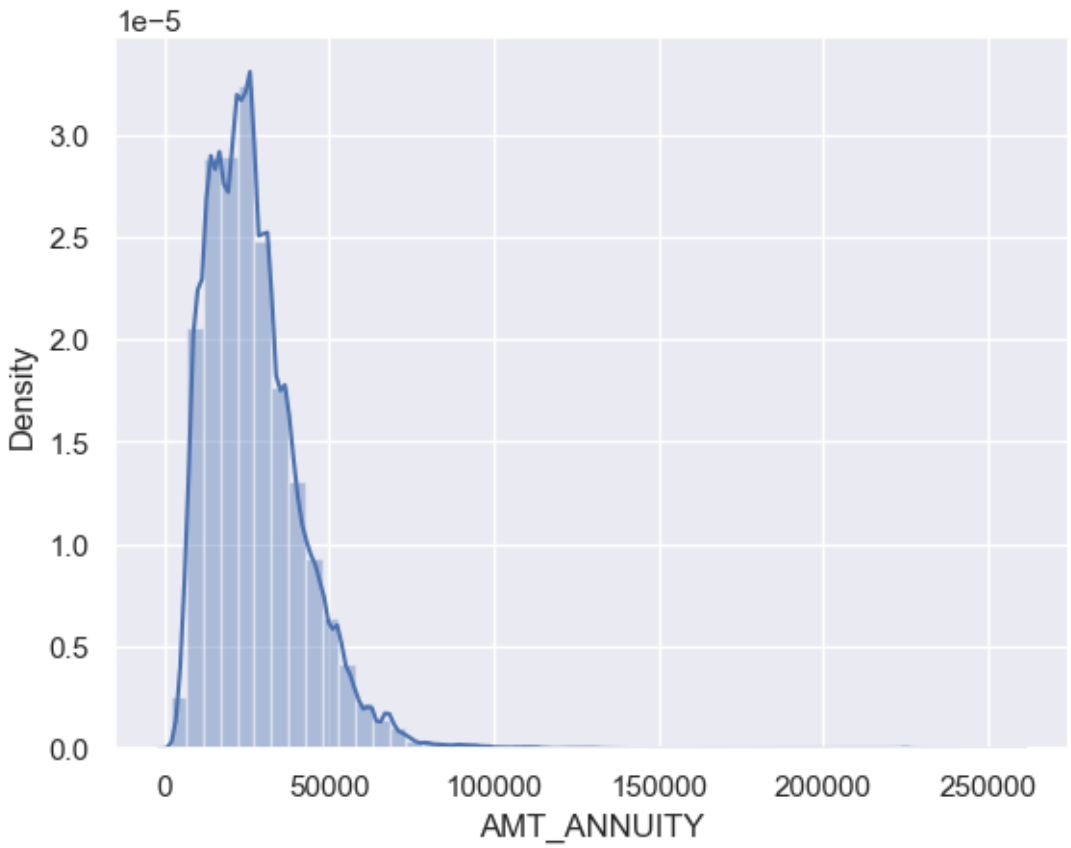

Plot of AMT_REQ_CREDIT_BUREAU_MON
25
20
15
10
5
0
0
50000
100000
150000
200000
250000
300000


Plot of AMT_REQ_CREDIT_BUREAU_QRT
250
200
150
100
50
0
0
50000
100000
150000
200000
250000
300000



[137]:
```
# Converting a numerial data to categorical for analysis

# Plot a distribution plot for the 'AMT_ANNUITY' column after removing any␣
 ↪missing values
sns.distplot(train['AMT_ANNUITY'].dropna())
```

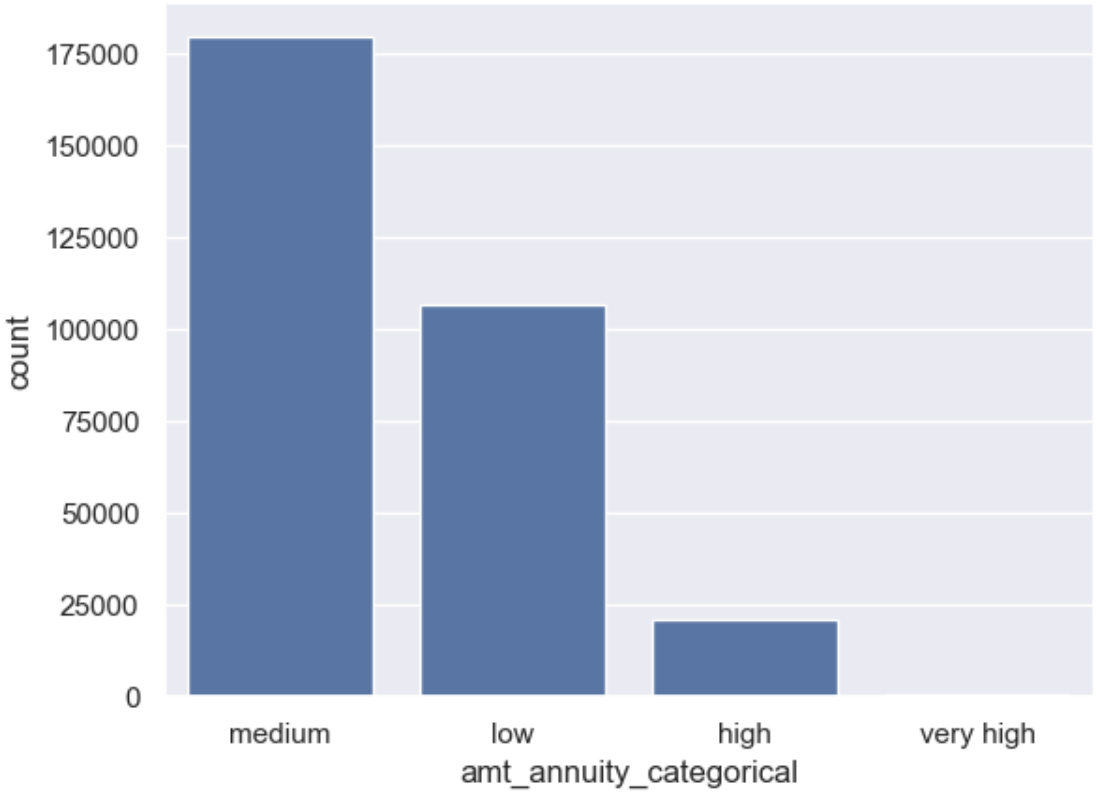
[137]: <Axes: xlabel='AMT_ANNUITY', ylabel='Density'>



[138]:
```
def amt_annuity(x):
    if x <= 20000:
        return 'low'
    elif x > 20000 and x <= 50000:
        return 'medium'
    elif x > 50000 and x <= 100000:
        return 'high'
    else:
        return 'very high'

# Apply the 'amt_annuity' function to create a new categorical column␣
 ↪'amt_annuity_categorical'
train['amt_annuity_categorical'] = train['AMT_ANNUITY'].apply(lambda x:␣
 ↪amt_annuity(x))
```

[139]:
```
# Plot a count plot of the 'amt_annuity_categorical' column
sns.countplot(x='amt_annuity_categorical', data=train)
```

[139]:
```
<Axes: xlabel='amt_annuity_categorical', ylabel='count'>
```

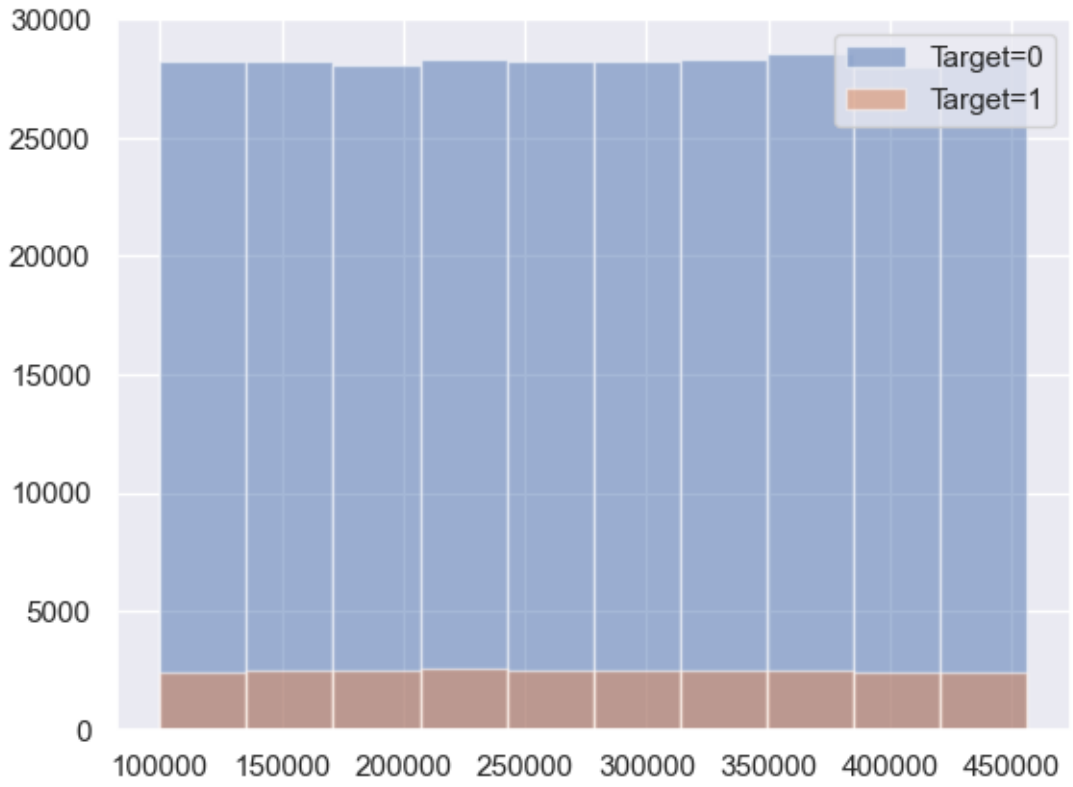


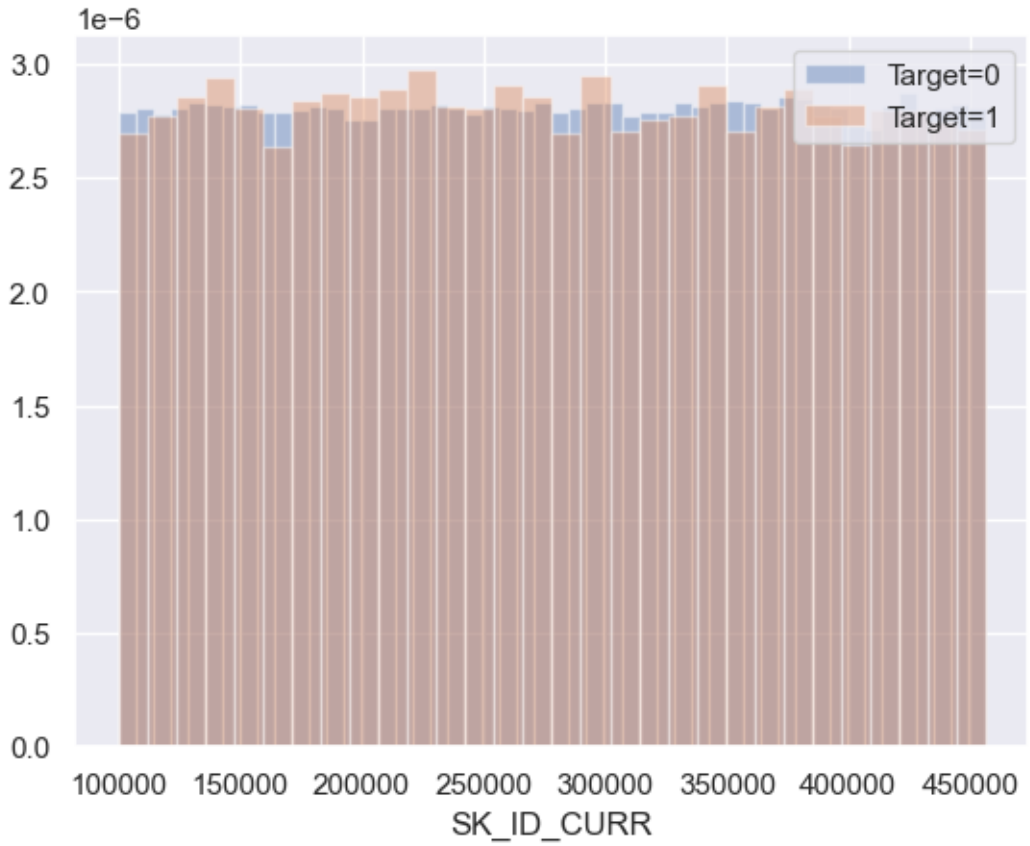
[140]:
```
# Task-4.2 - Univariate Analysis for Numerical data
# For univariate analysis of the numerical columns, we will plot the histogram␣
 ↪and the distribution plot.
```

[141]:
```
# Iterate over each categorical column to generate plots
for column in train_categorical:
    # Construct and print the title for current plot
    title = "Plot of " + column
    print(title)

    # Plot histograms for the distribution of the variable for both TARGET␣
 ↪categories
    plt.hist(train_0[column], alpha=0.5, label='Target=0')  # Histogram for␣
 ↪category '0'
    plt.hist(train_1[column], alpha=0.5, label='Target=1')  # Histogram for␣
 ↪category '1'
    plt.legend()  # Add a legend to distinguish between categories
    plt.show()  # Display the histogram

    # Plot distribution plots for the non-null values in both TARGET categories
```

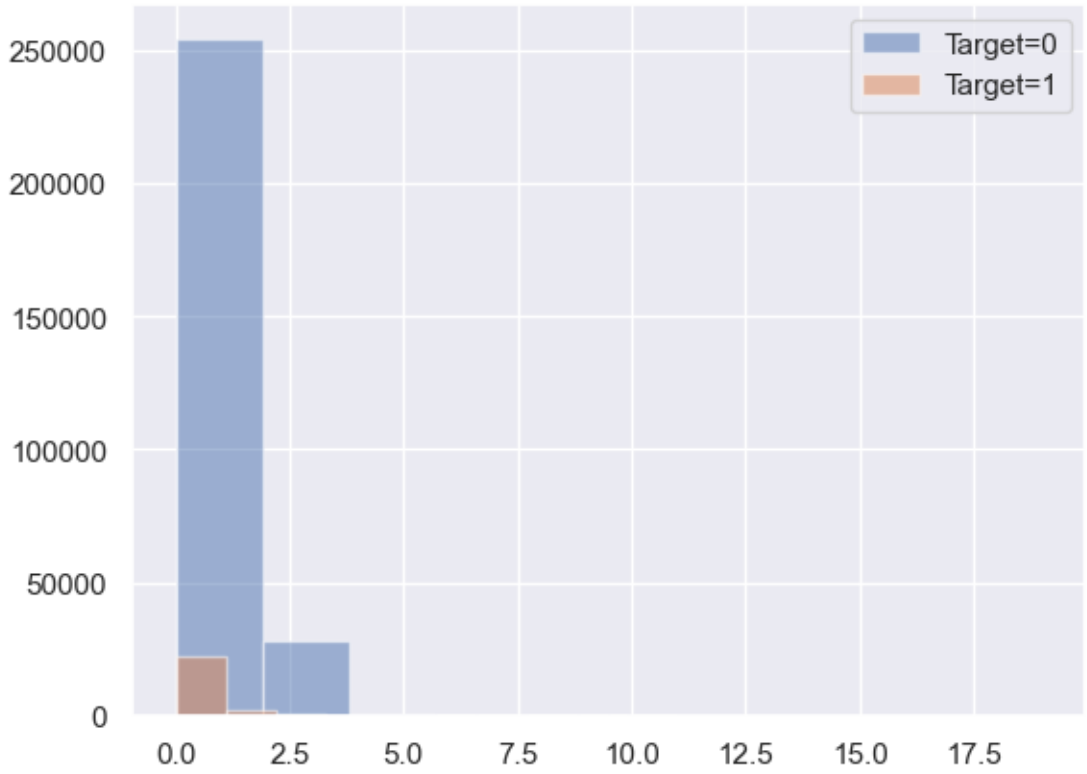
```
    sns.distplot(train_0[column].dropna(), label='Target=0', kde=False,␣
↪norm_hist=True)  # Distribution plot for '0'
    sns.distplot(train_1[column].dropna(), label='Target=1', kde=False,␣
↪norm_hist=True)  # Distribution plot for '1'
    plt.legend()  # Add a legend to distinguish between categories
    plt.show()  # Display the distribution plot

    # Placeholder for a box plot function that might be defined elsewhere
    # box_plot(train_0, train_1, column)

    # Print a separator for readability between plots
  ␣
↪print("------------------------------------------------------------------------")
```

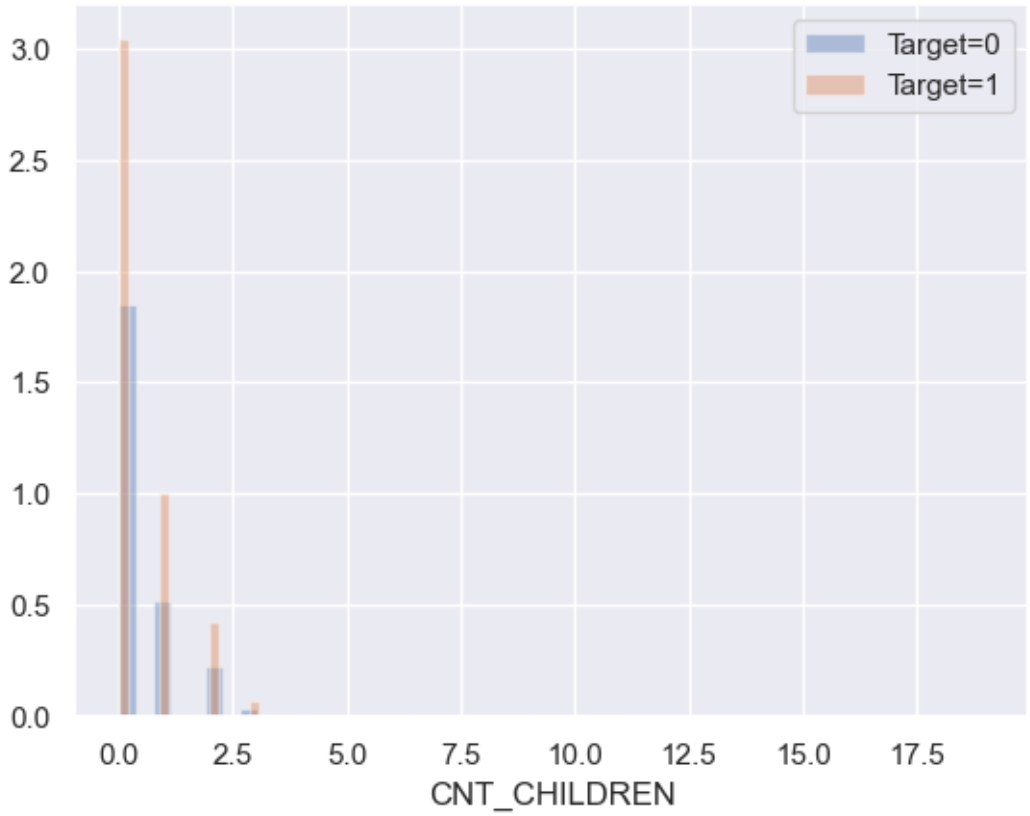
```
Plot of SK_ID_CURR
```



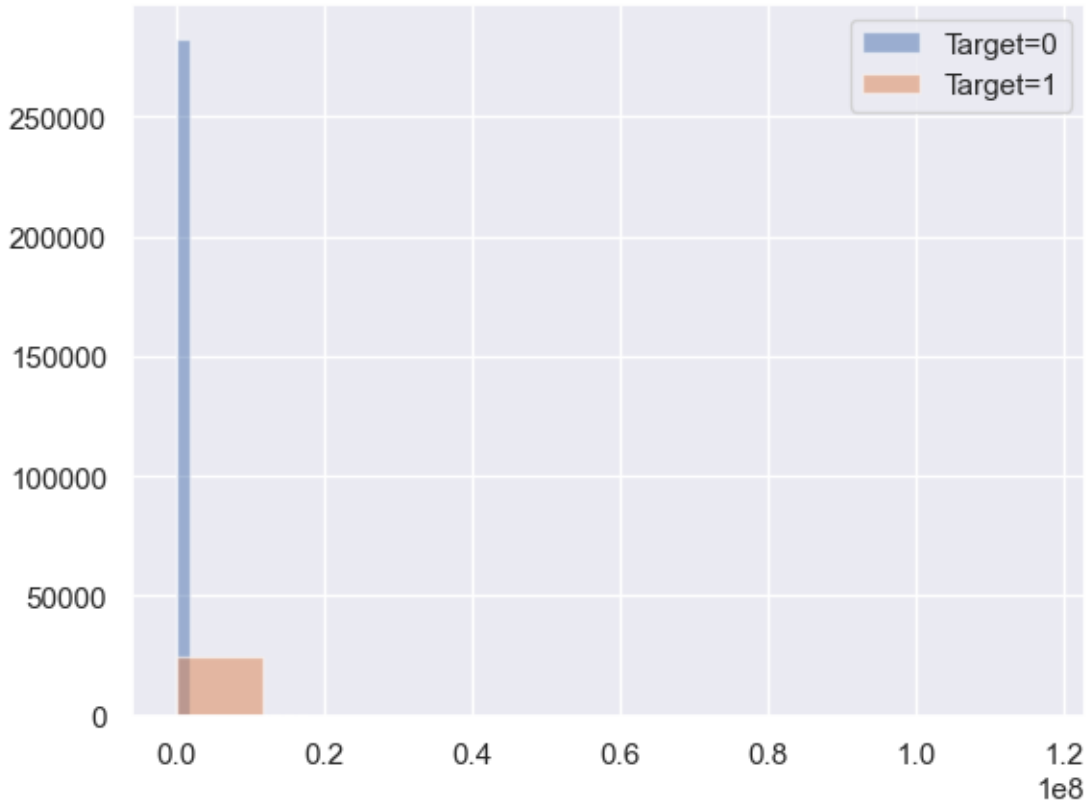


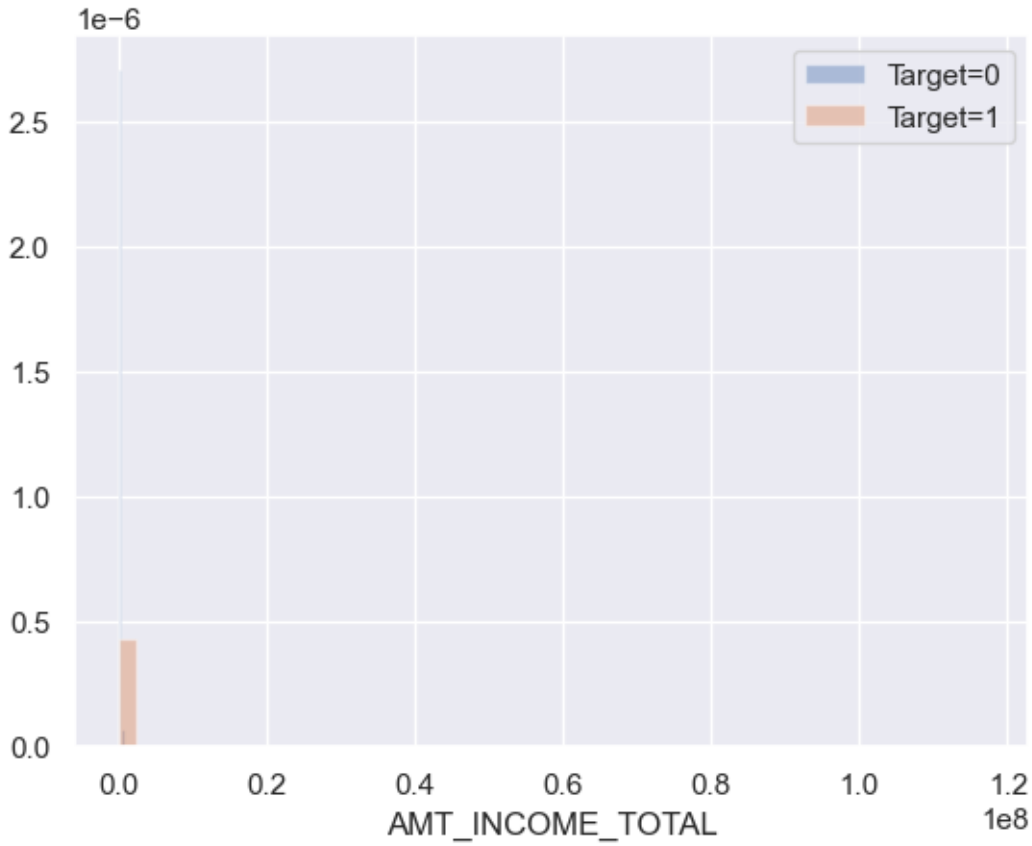
Plot of CNT_CHILDREN


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

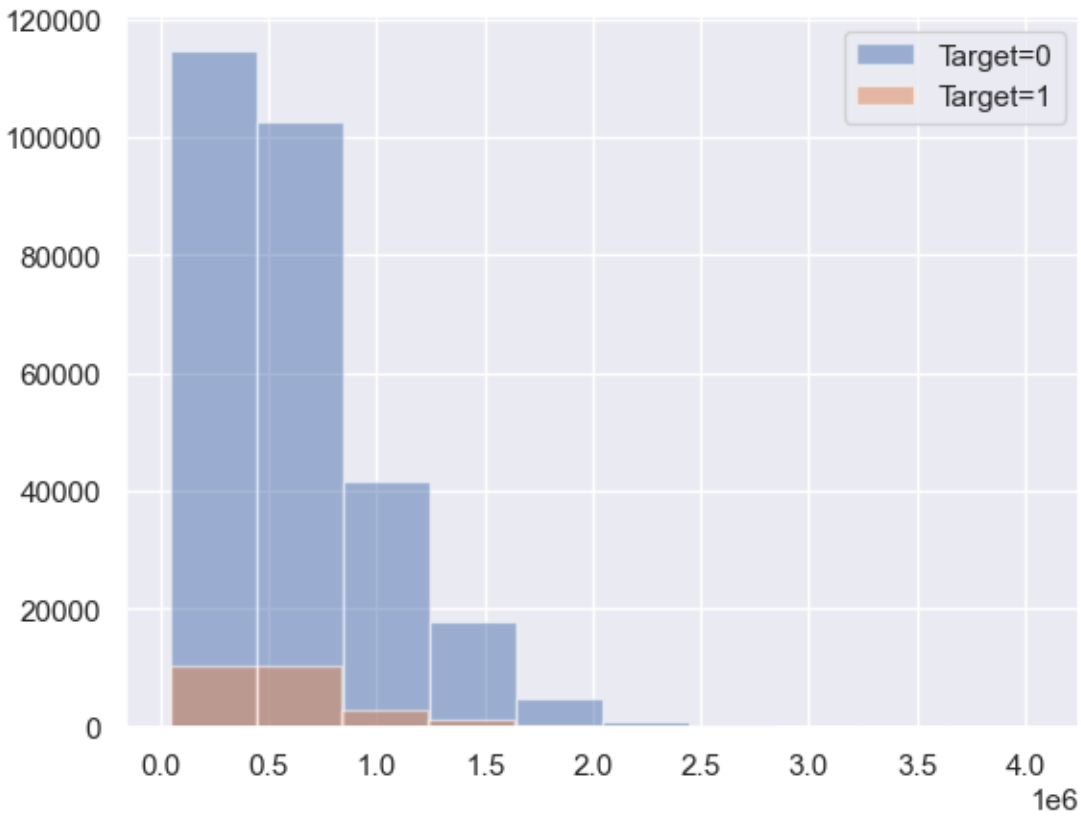


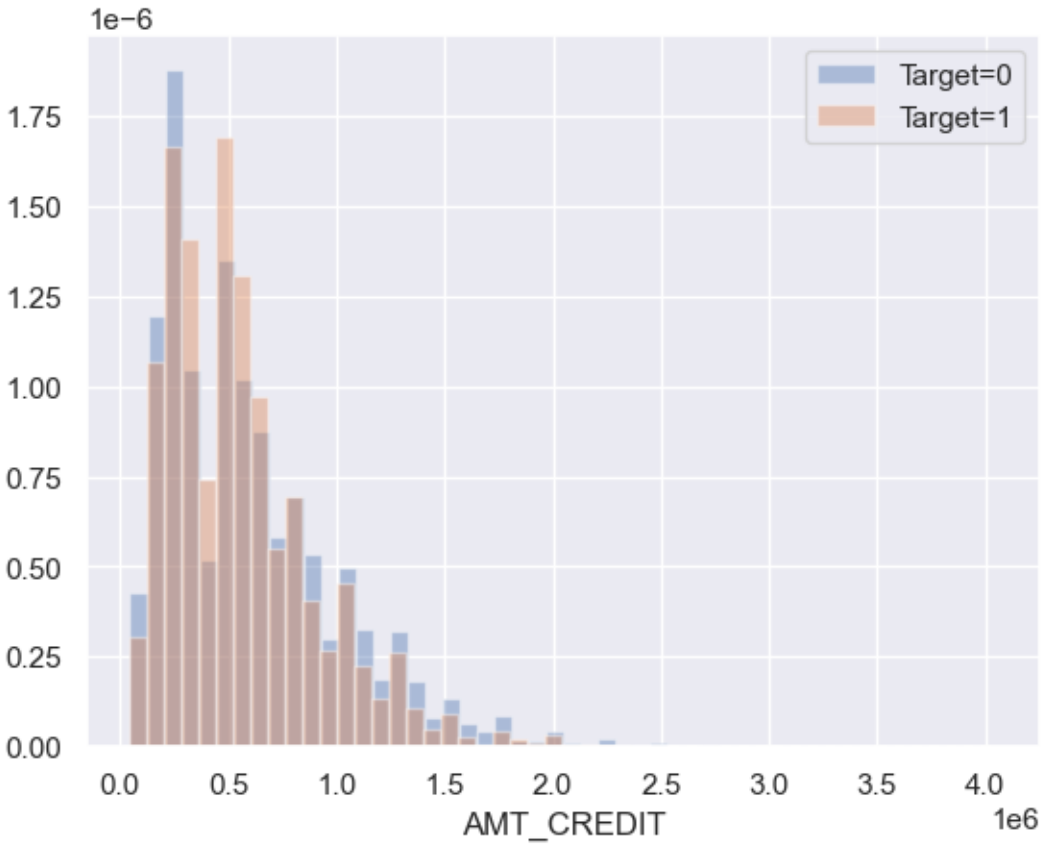
Plot of AMT_INCOME_TOTAL


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e8

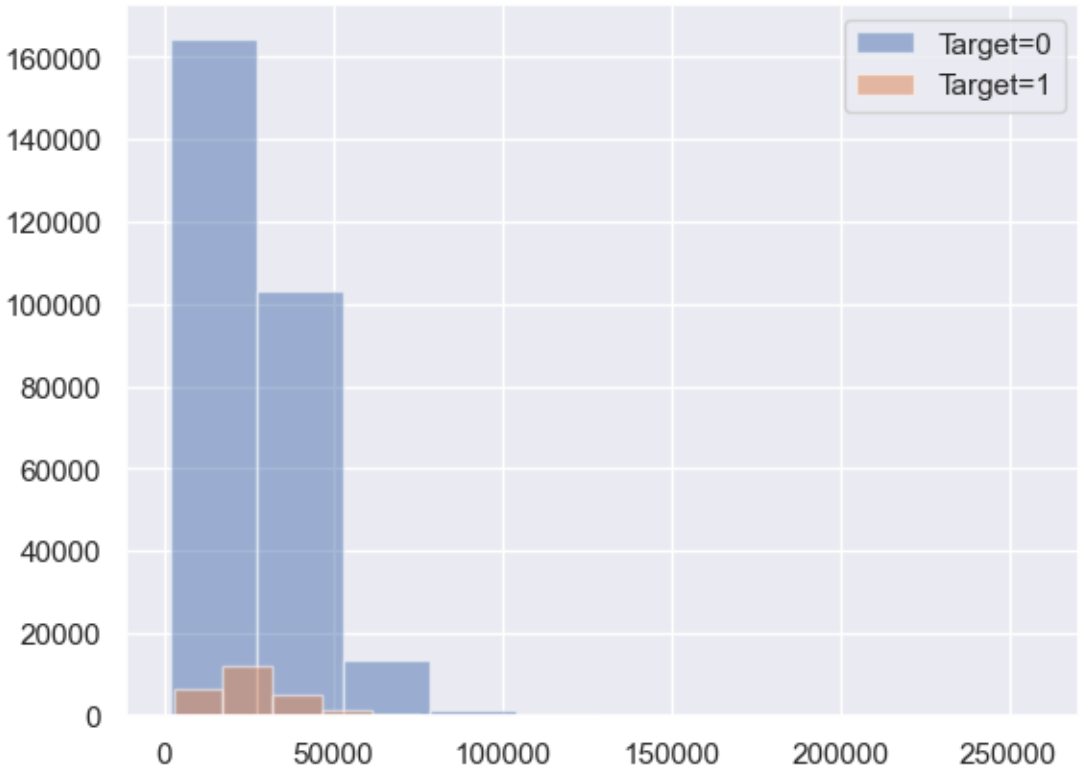


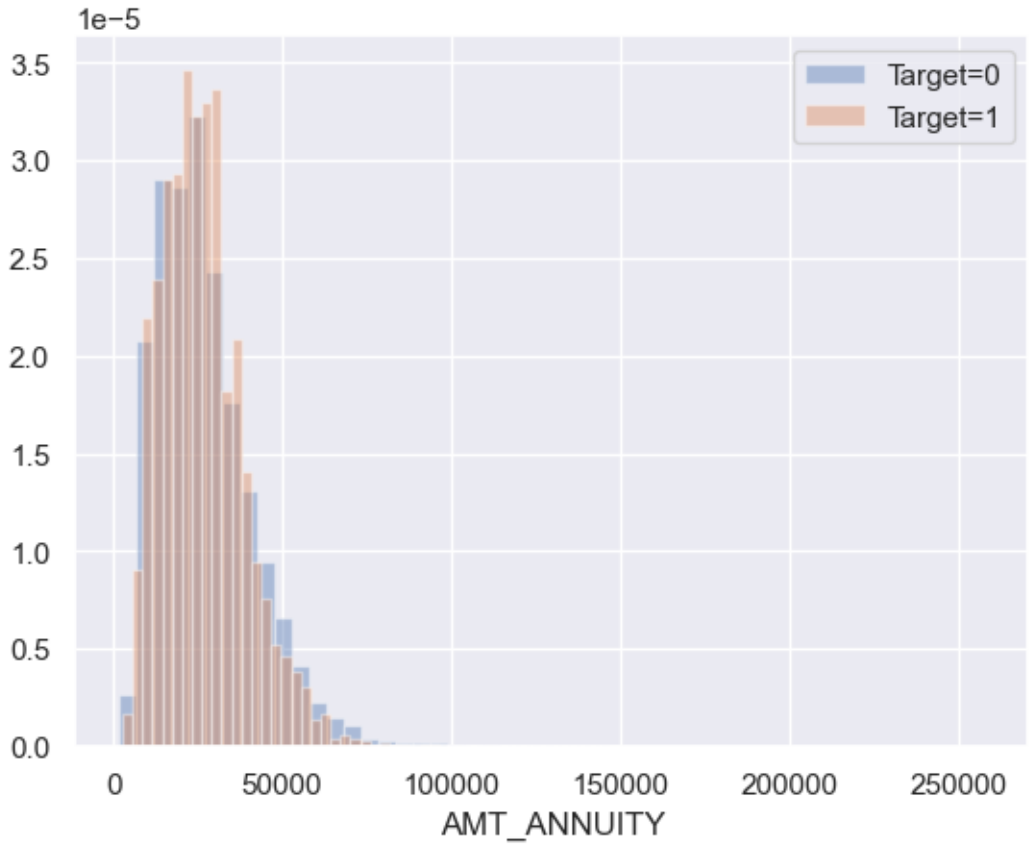
Plot of AMT_CREDIT


120000
100000
80000
60000
40000
20000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e6
Target=0
Target=1

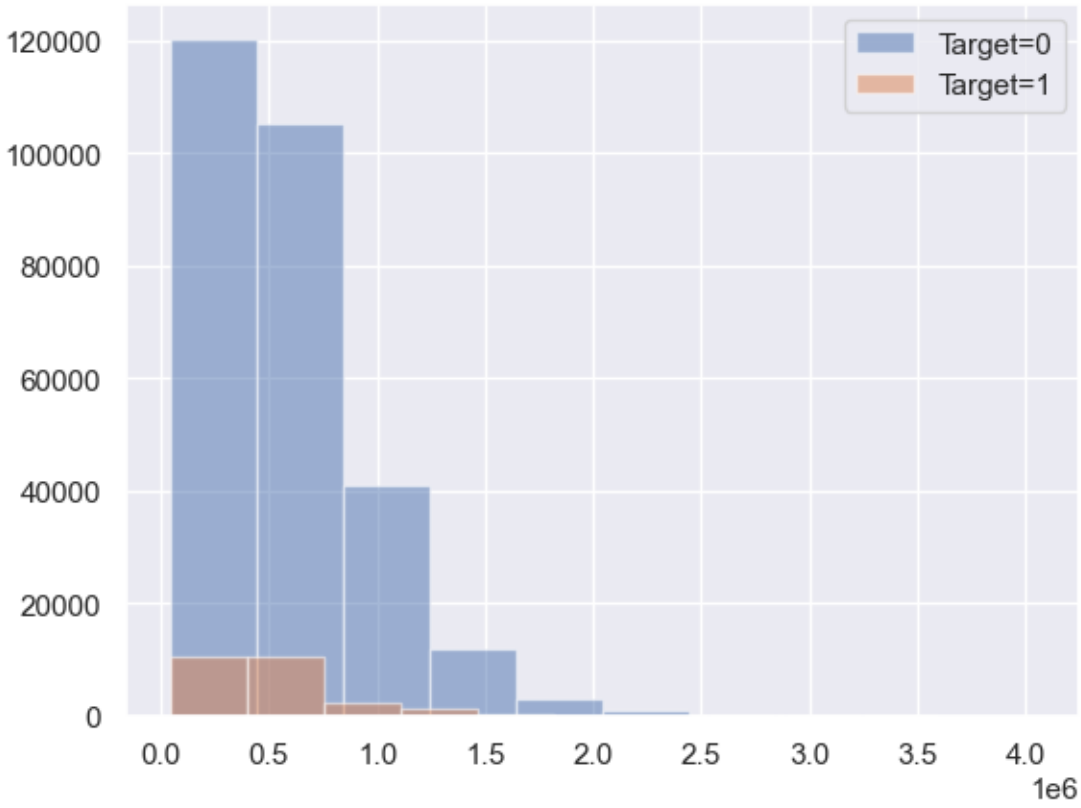


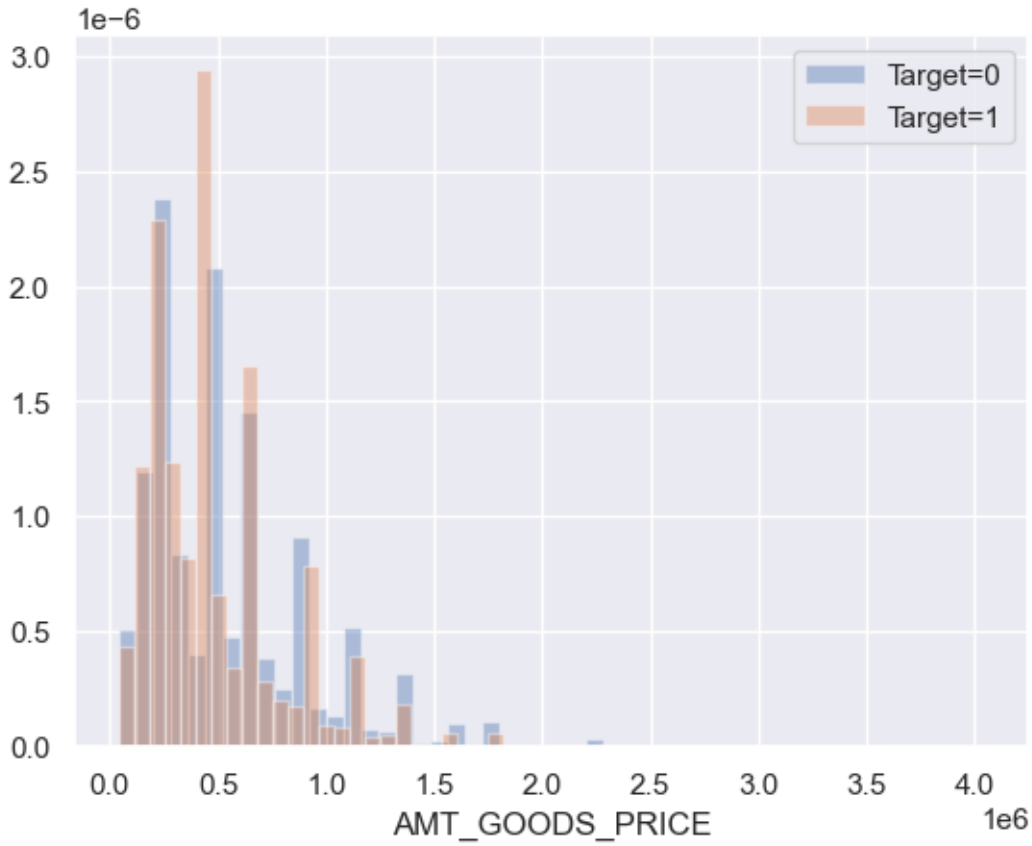
Plot of AMT_ANNUITY


Target=0
Target=1
160000
140000
120000
100000
80000
60000
40000
20000
0
0
50000
100000
150000
200000
250000

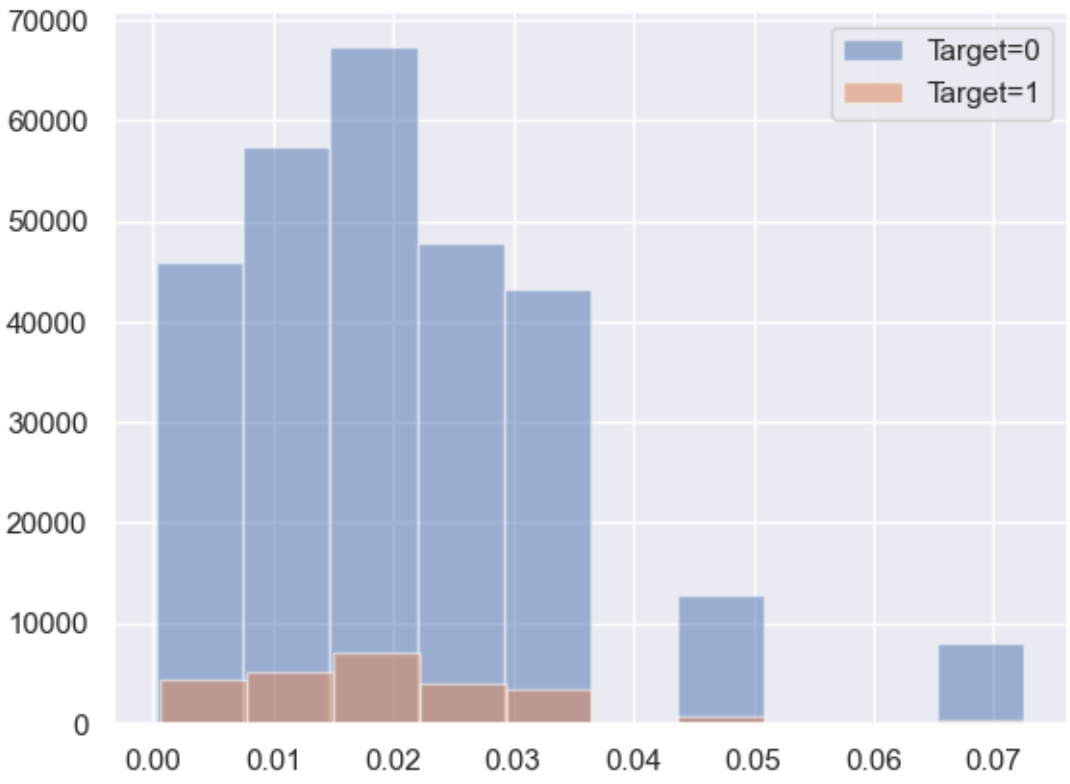


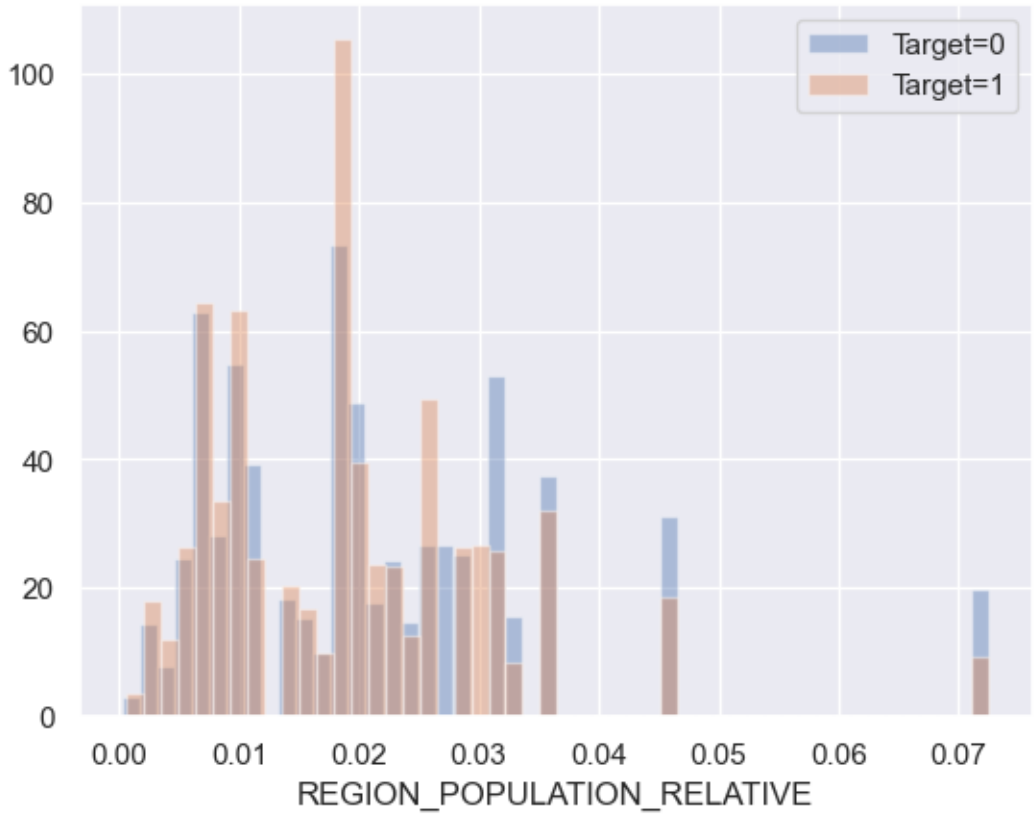
Plot of AMT_GOODS_PRICE


Target=0
Target=1
120000
100000
80000
60000
40000
20000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e6

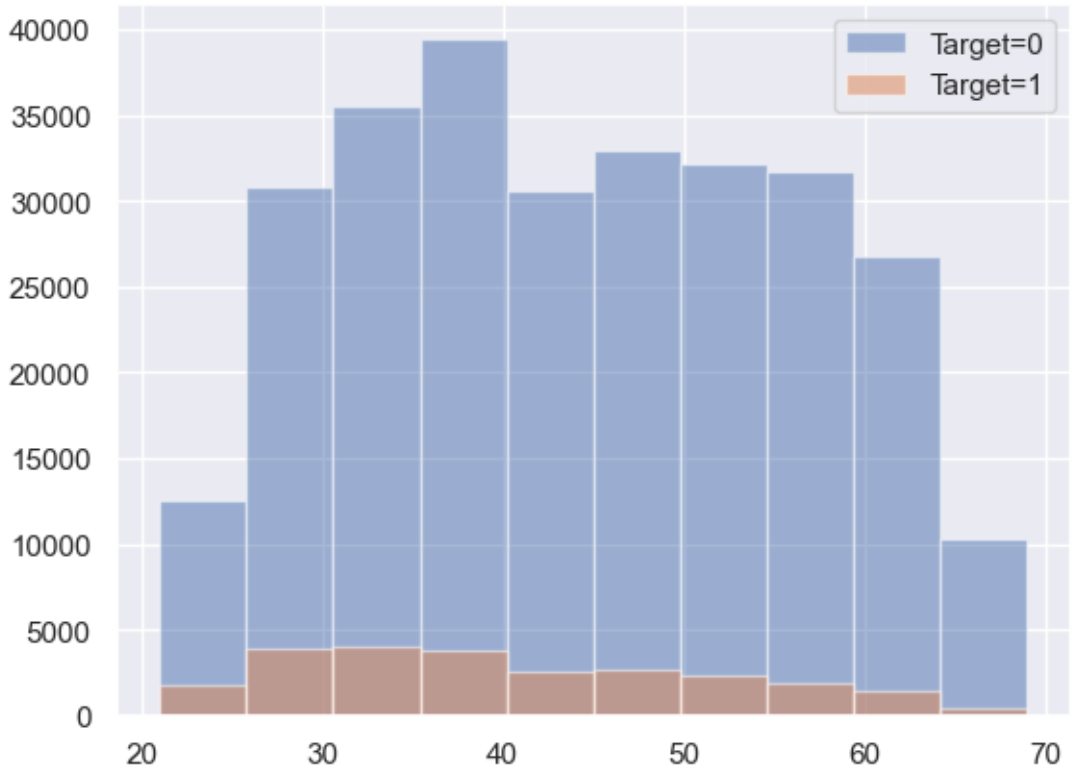


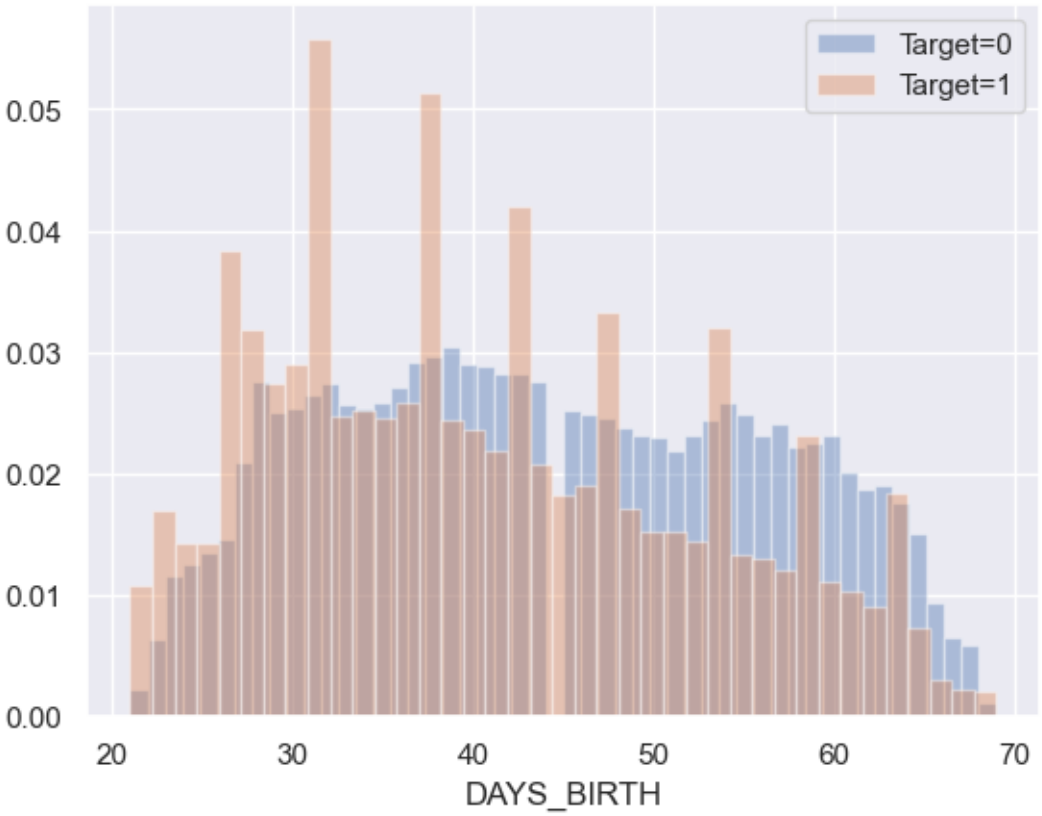
Plot of REGION_POPULATION_RELATIVE


70000
60000
50000
40000
30000
20000
10000
0
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Target=0
Target=1

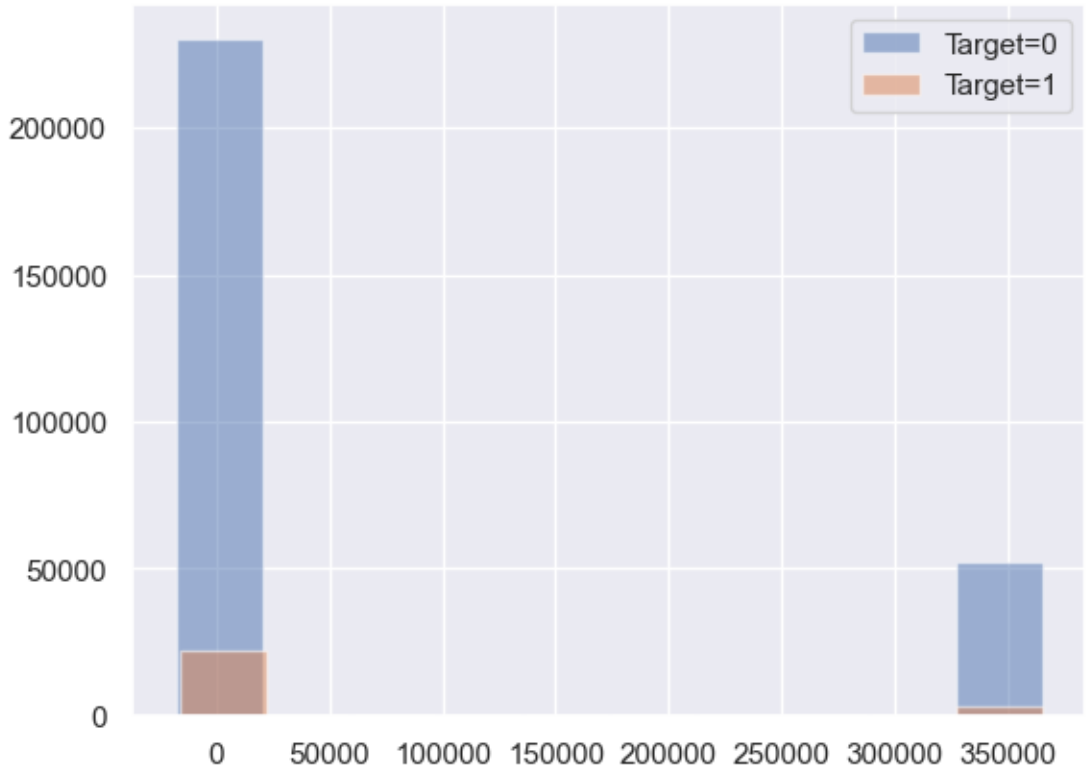


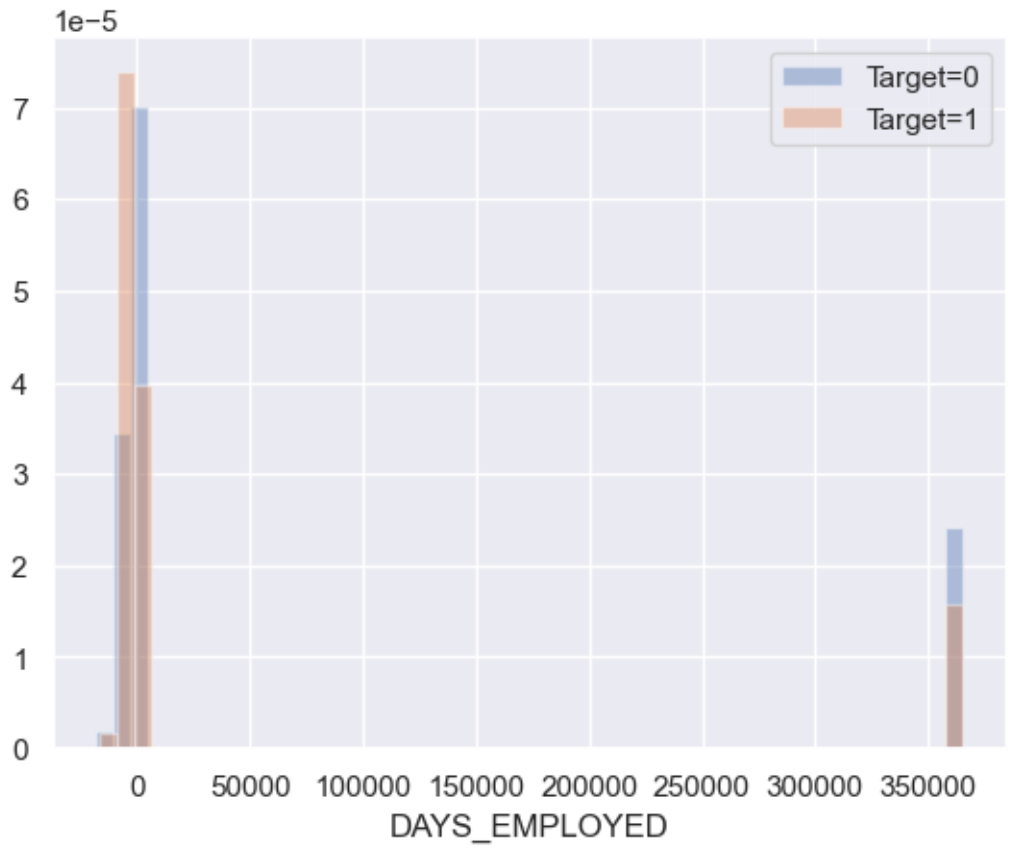
Plot of DAYS_BIRTH


Target=0
Target=1
40000
35000
30000
25000
20000
15000
10000
5000
0
20
30
40
50
60
70

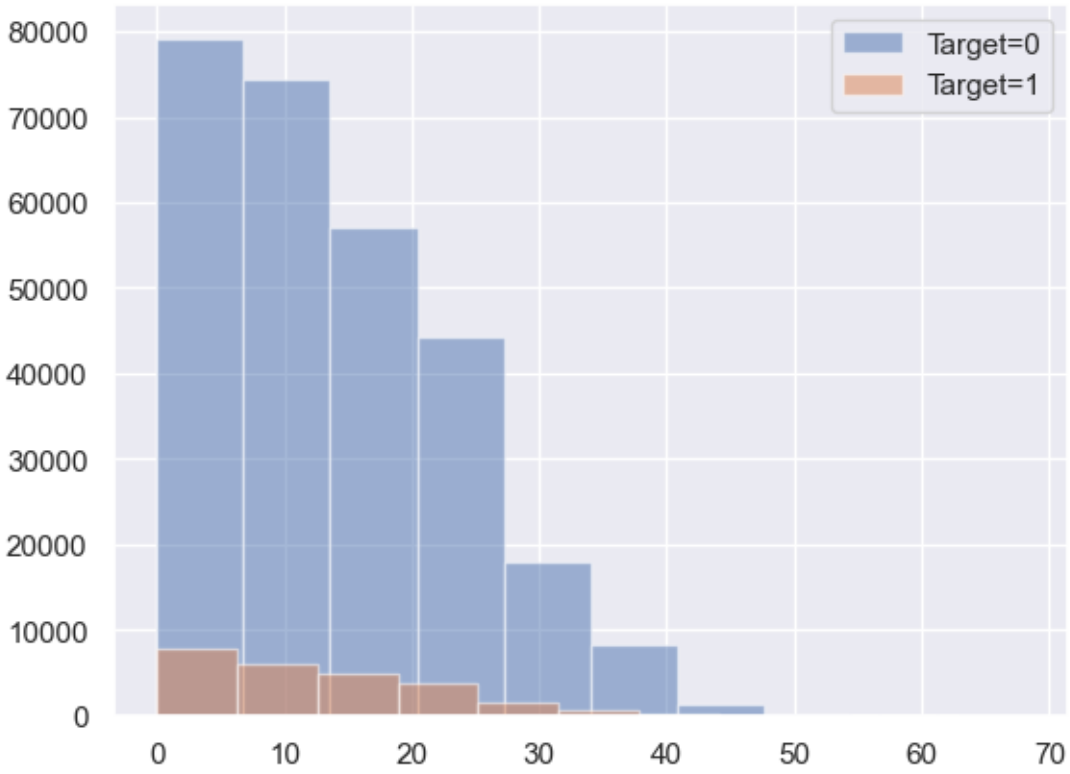


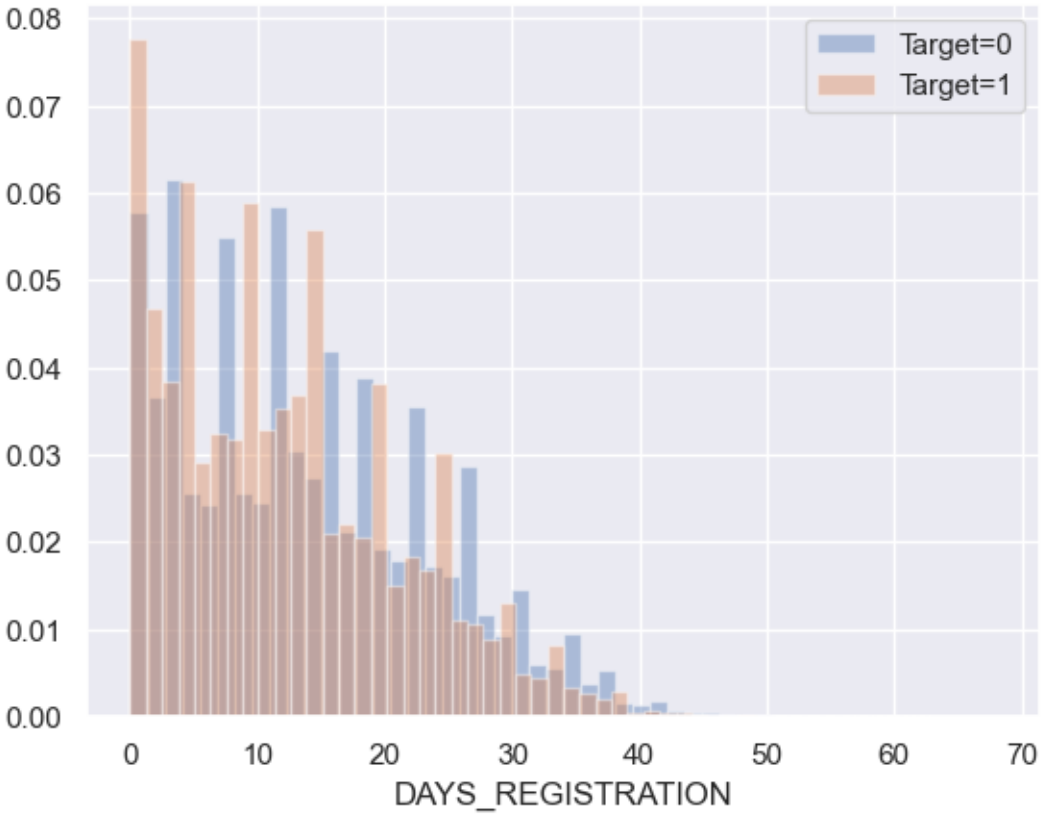
Plot of DAYS_EMPLOYED


Target=0
Target=1
200000
150000
100000
50000
0
0
50000
100000
150000
200000
250000
300000
350000

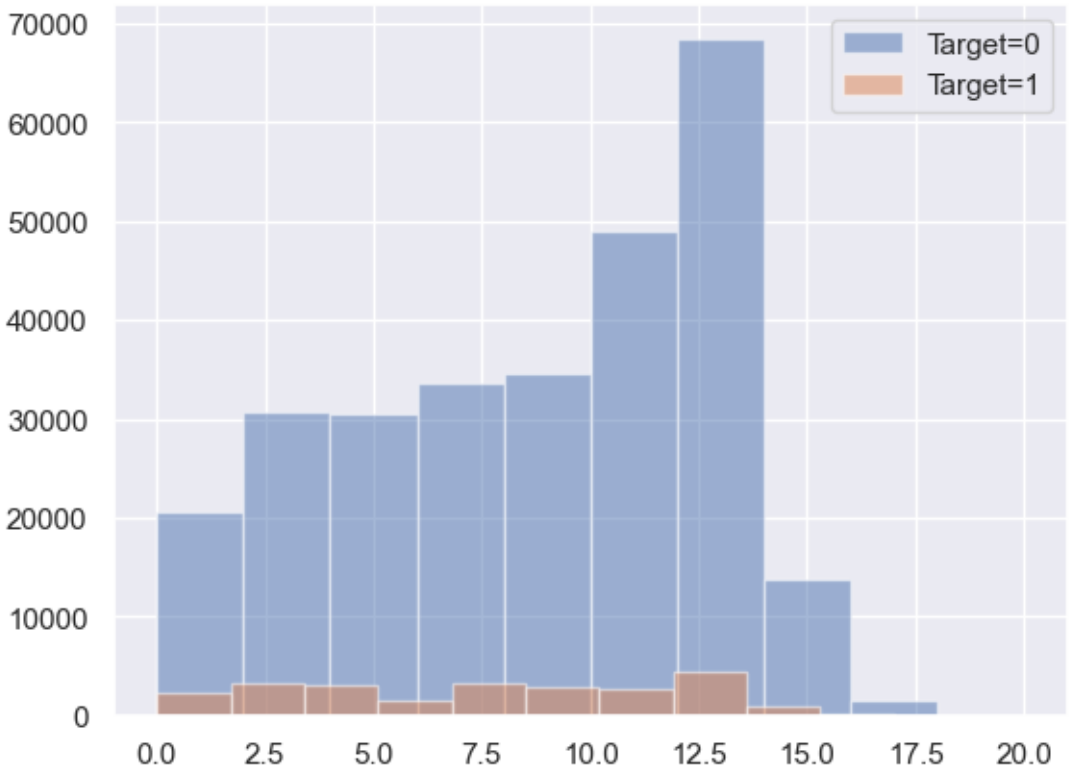


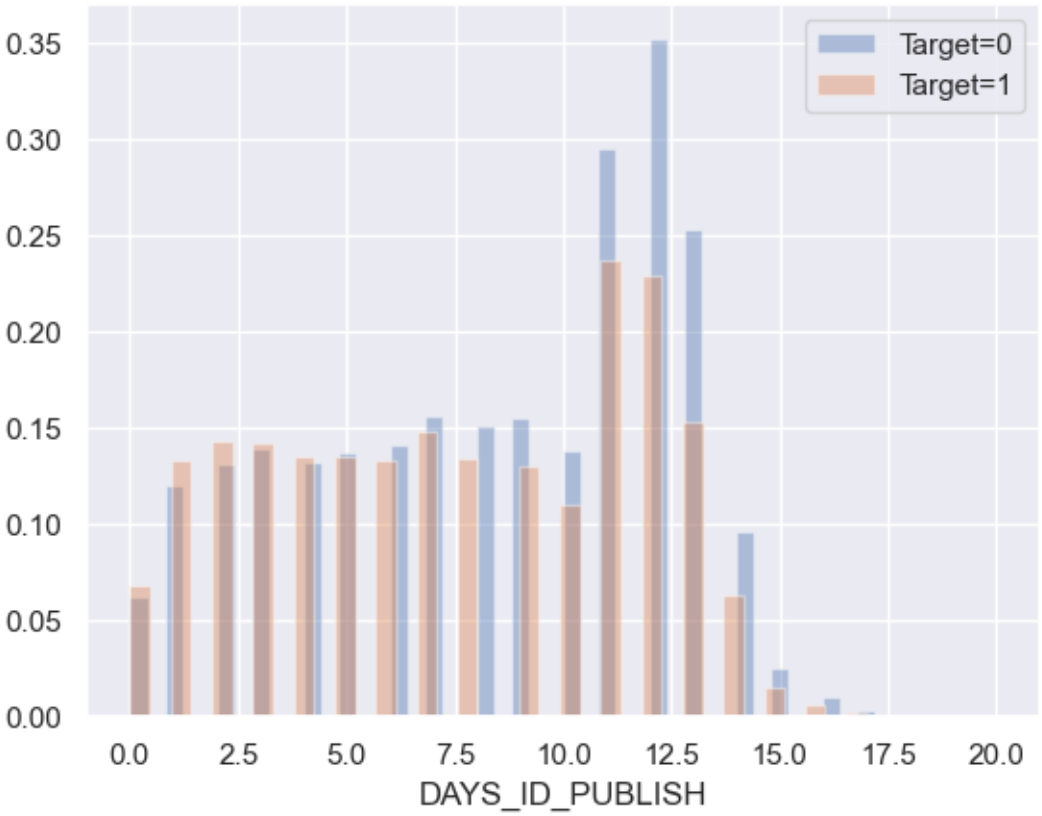
Plot of DAYS_REGISTRATION


Target=0
Target=1
80000
70000
60000
50000
40000
30000
20000
10000
0
0
10
20
30
40
50
60
70

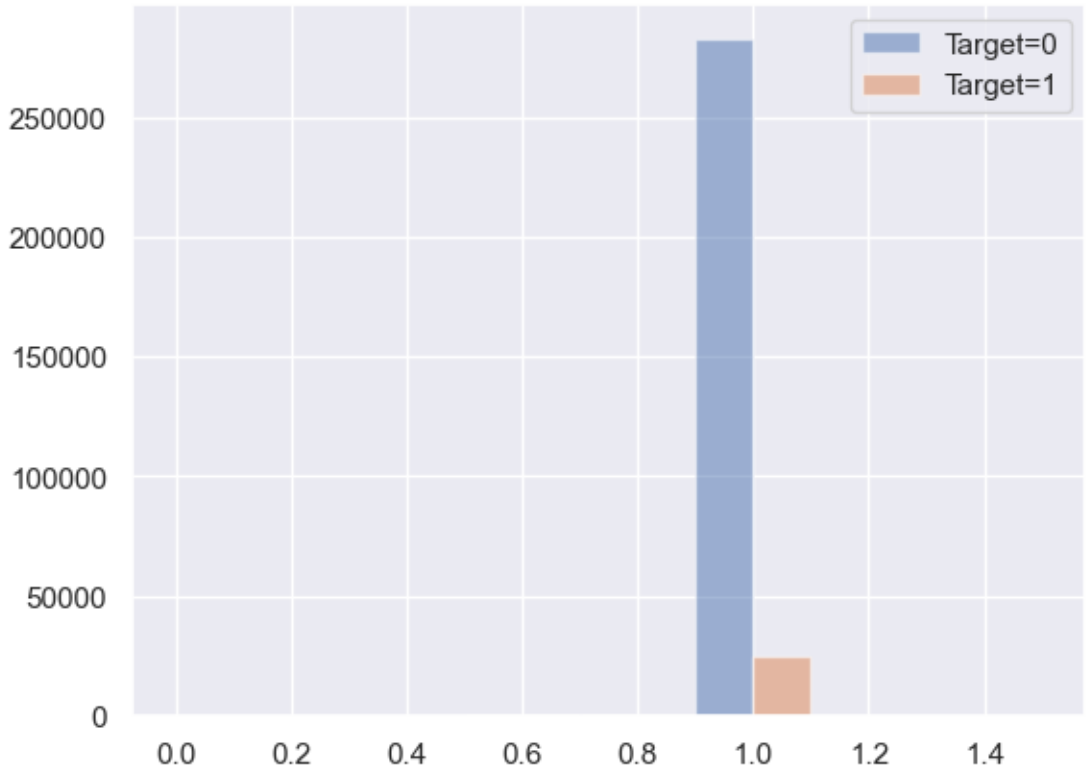


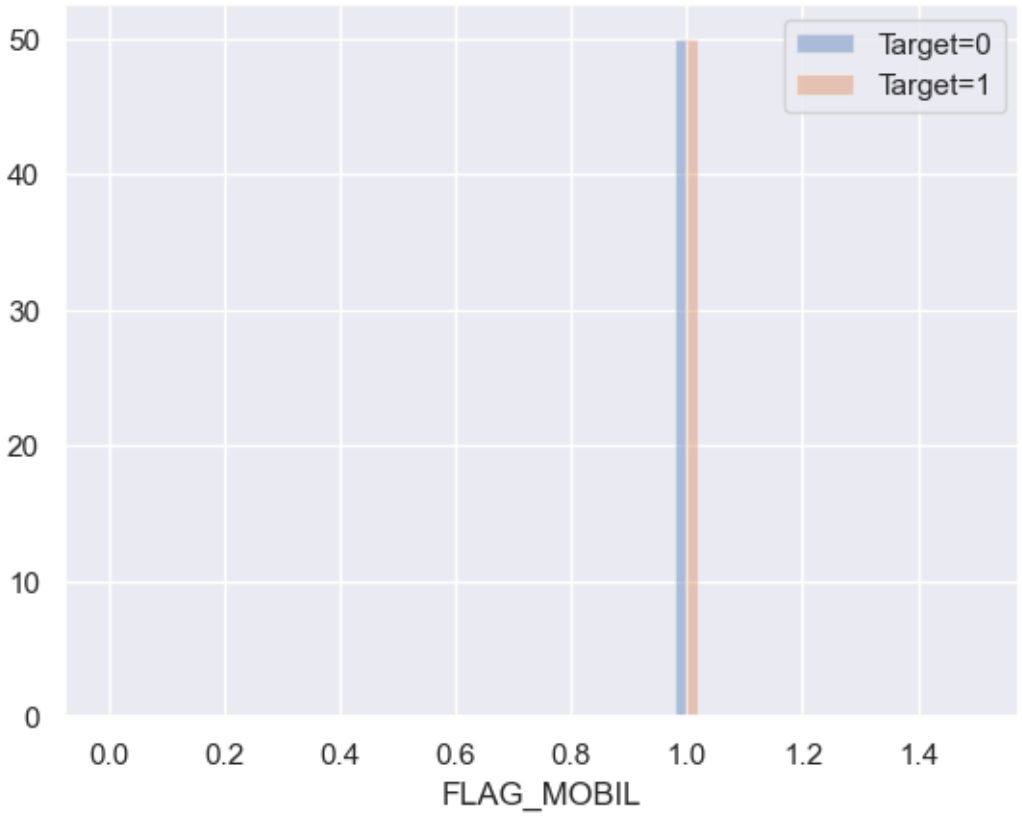
Plot of DAYS_ID_PUBLISH


Target=0
Target=1
70000
60000
50000
40000
30000
20000
10000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

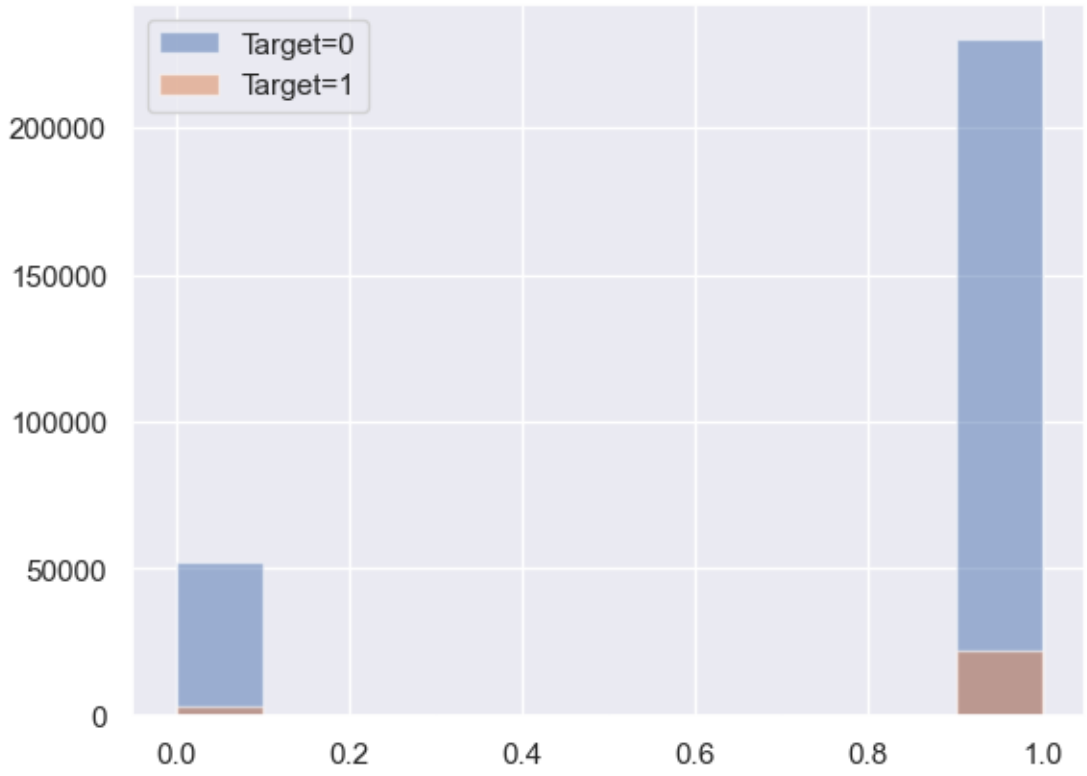


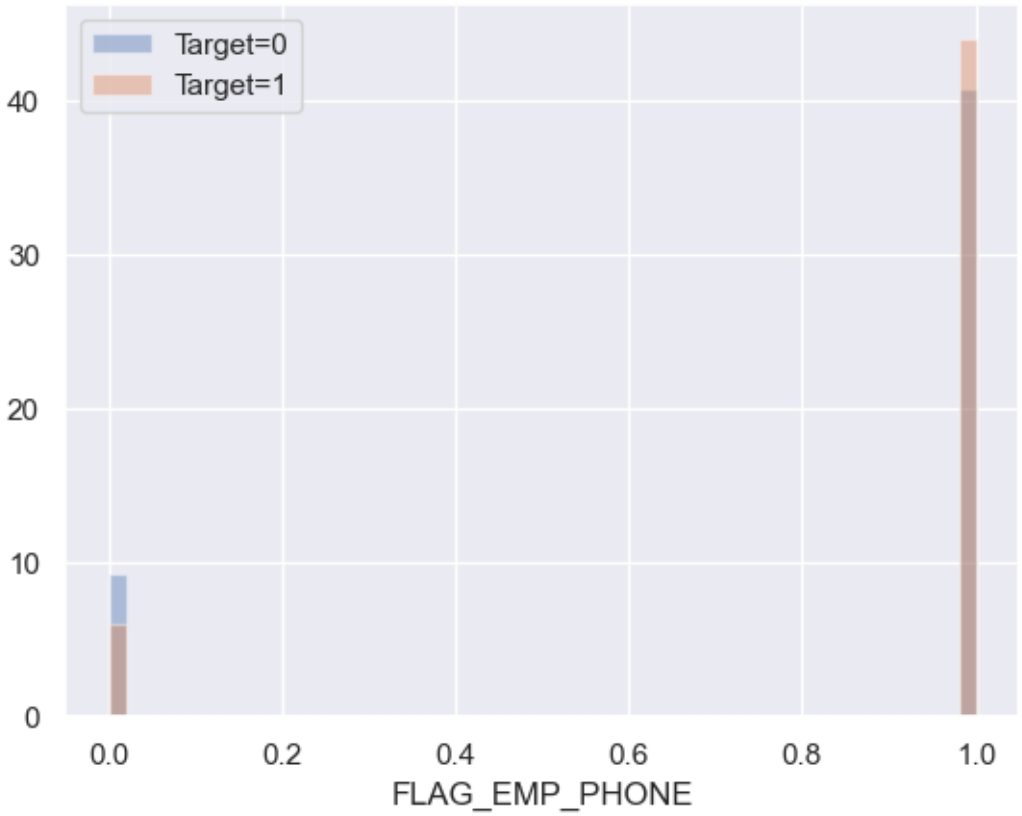
Plot of FLAG_MOBIL


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

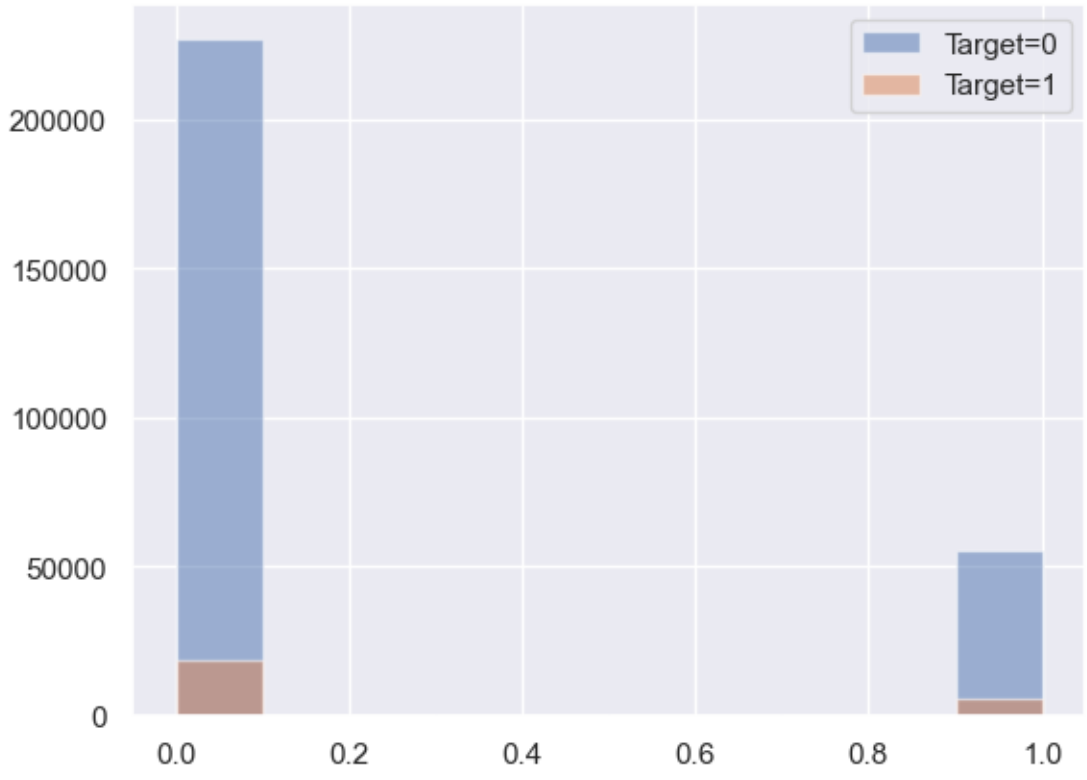


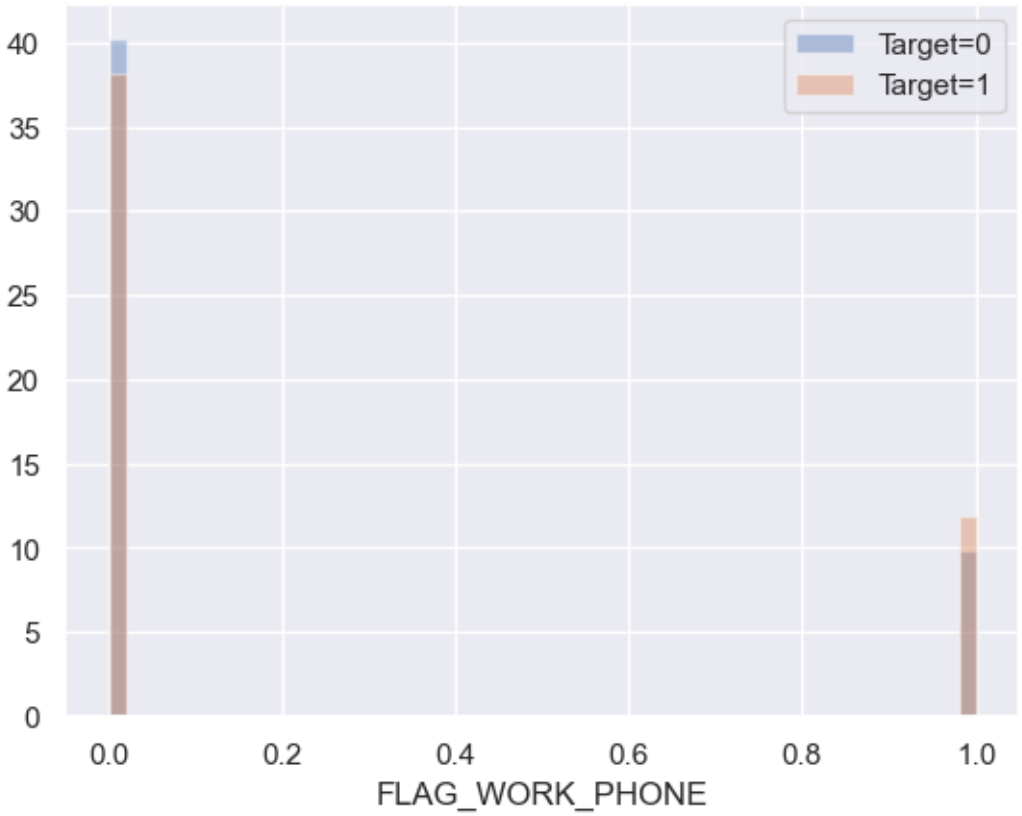
Plot of FLAG_EMP_PHONE


Target=0
Target=1
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

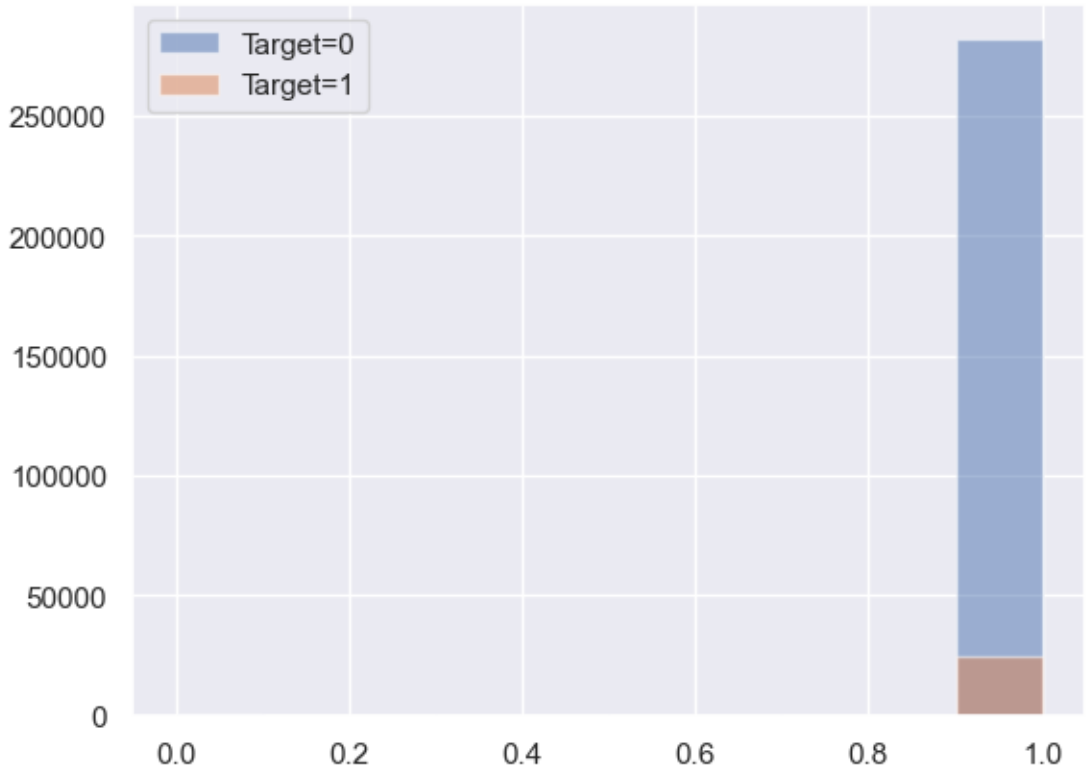


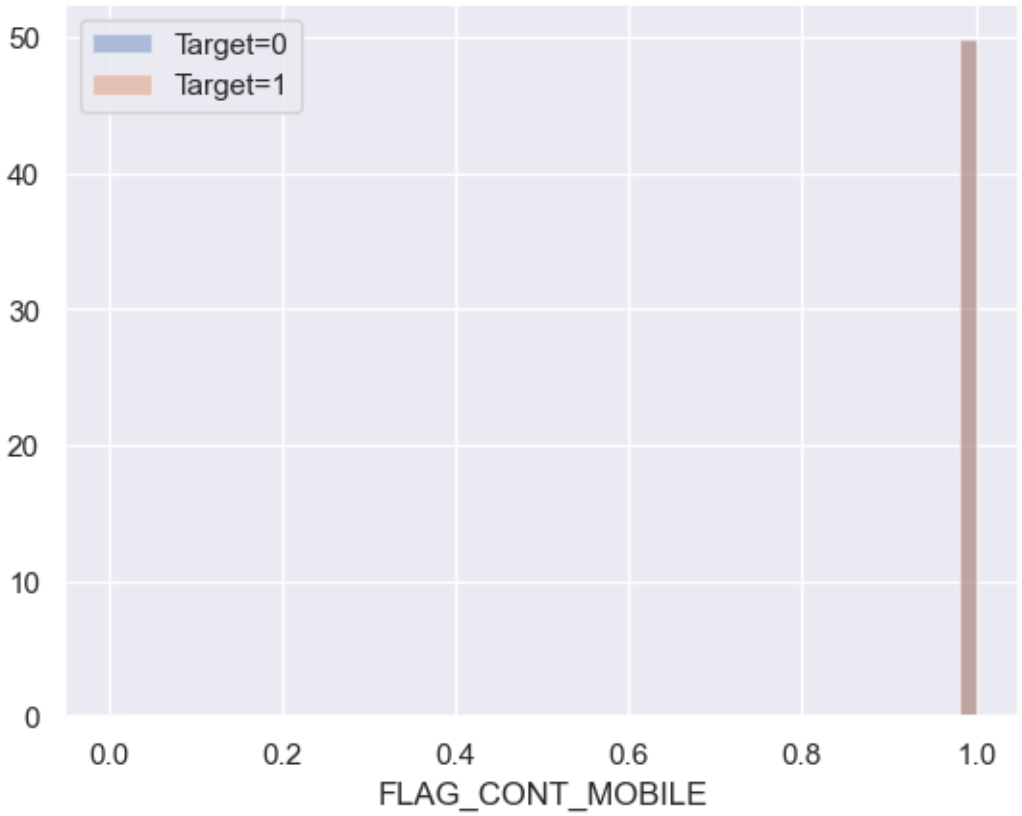
Plot of FLAG_WORK_PHONE


Target=0
Target=1
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

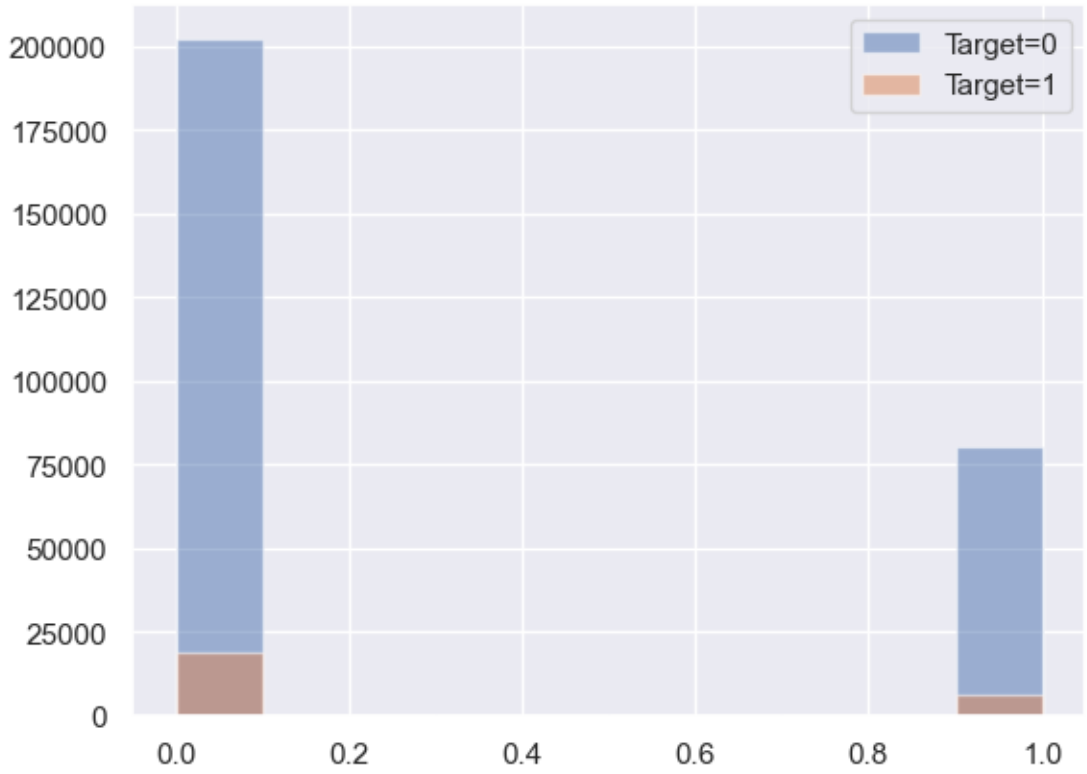


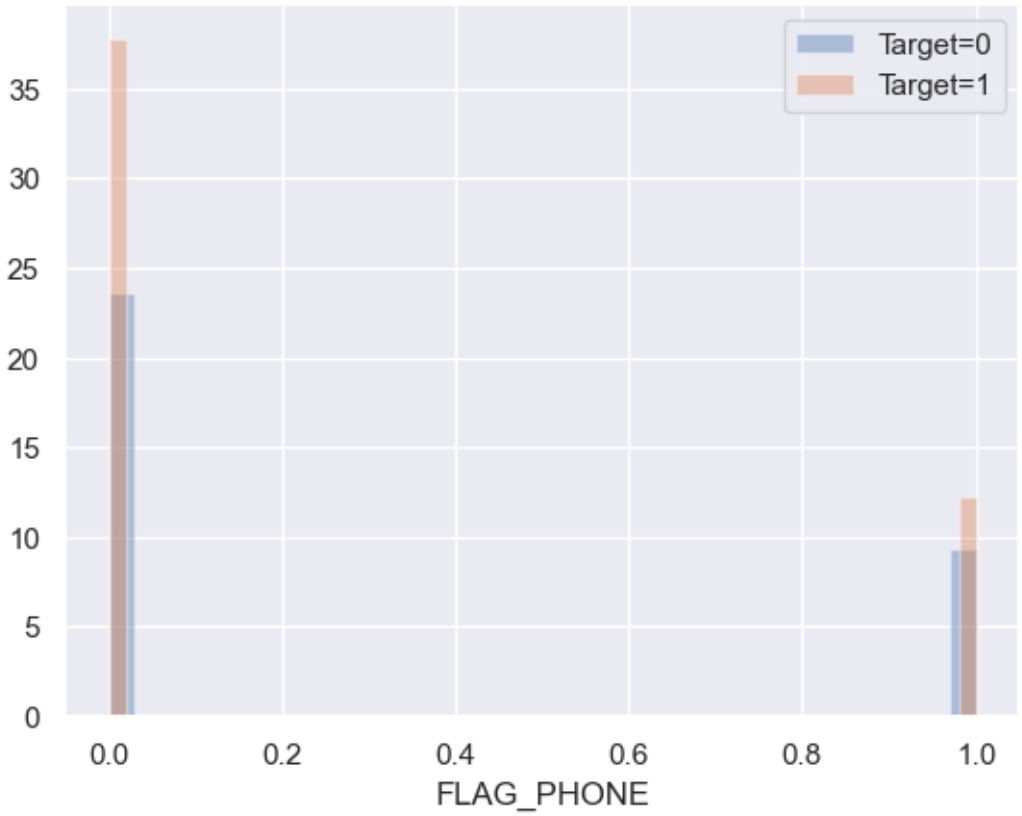
Plot of FLAG_CONT_MOBILE


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

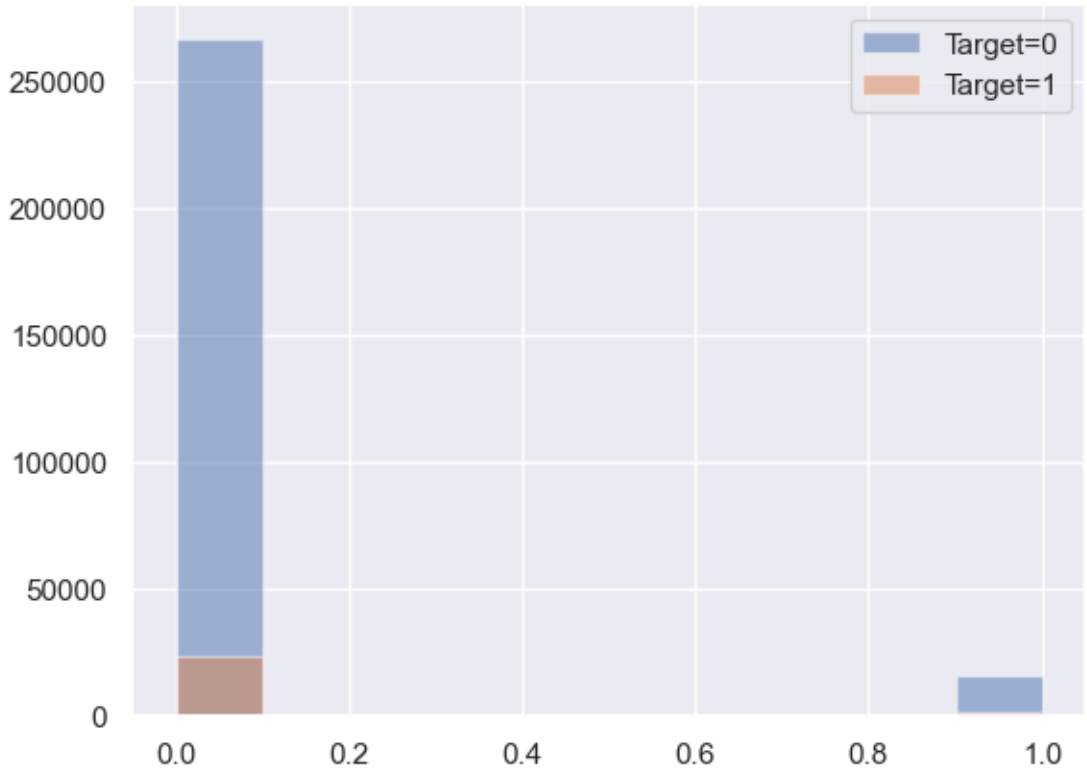


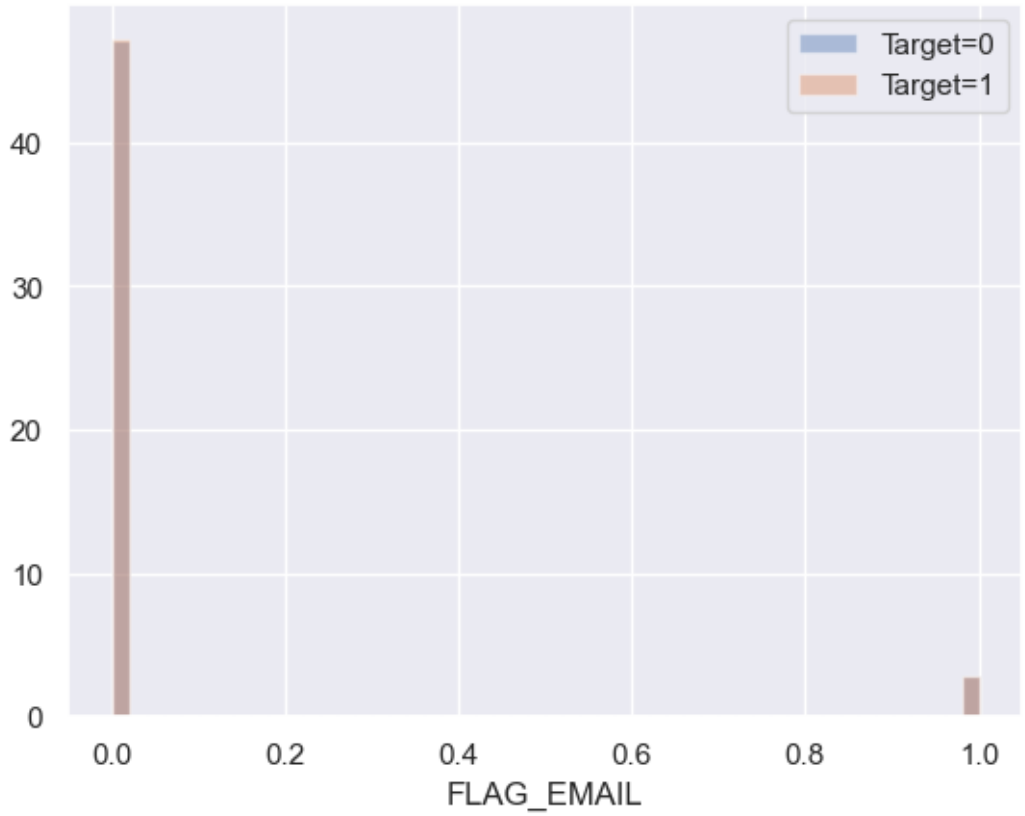
Plot of FLAG_PHONE


Target=0
Target=1
200000
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.2
0.4
0.6
0.8
1.0

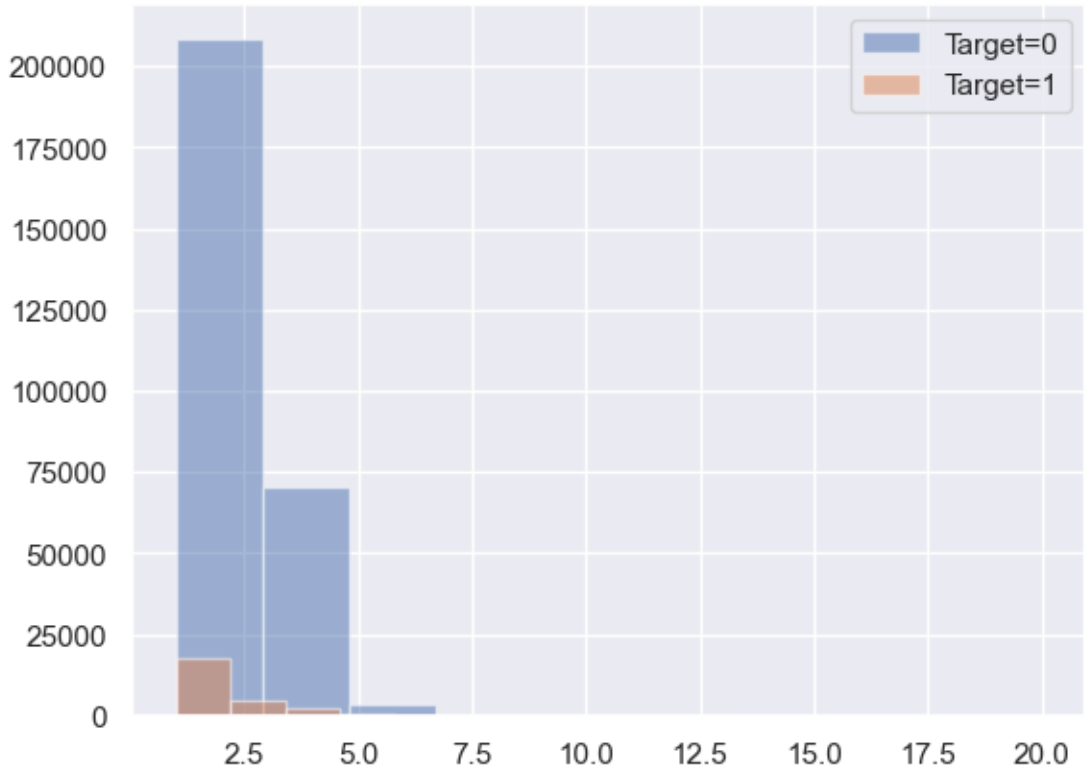


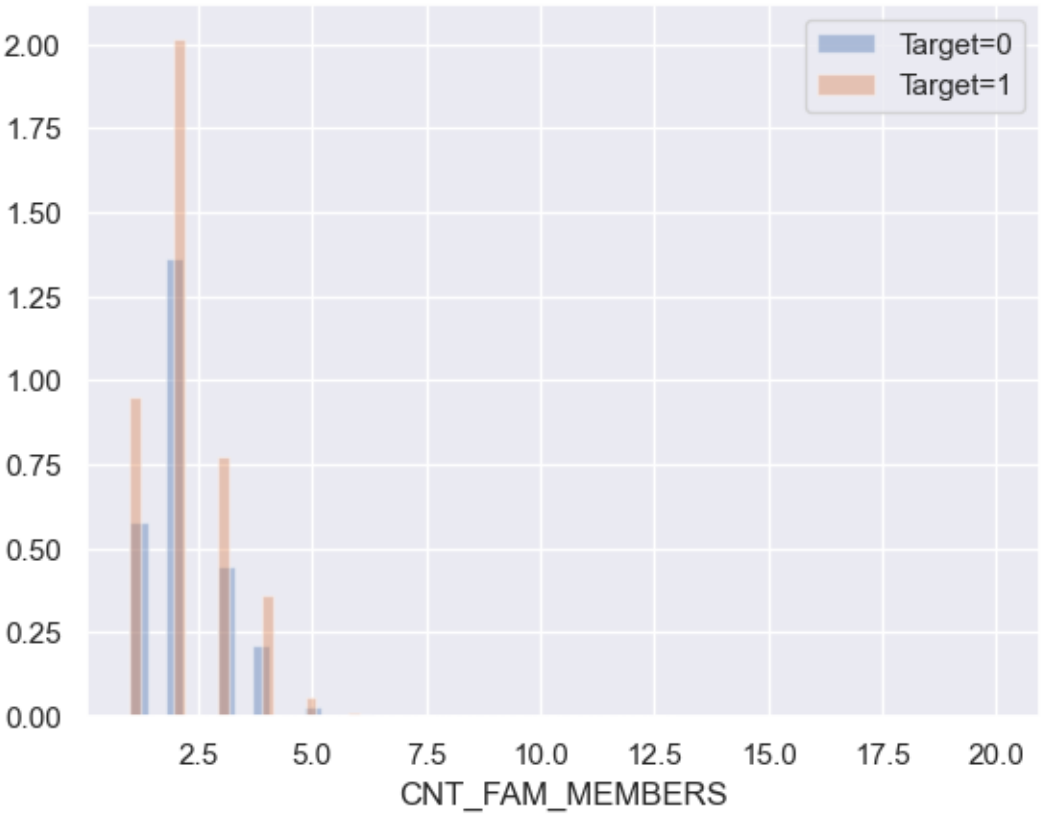

```
----------------------------------------------------------------------
Plot of FLAG_EMAIL
```


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

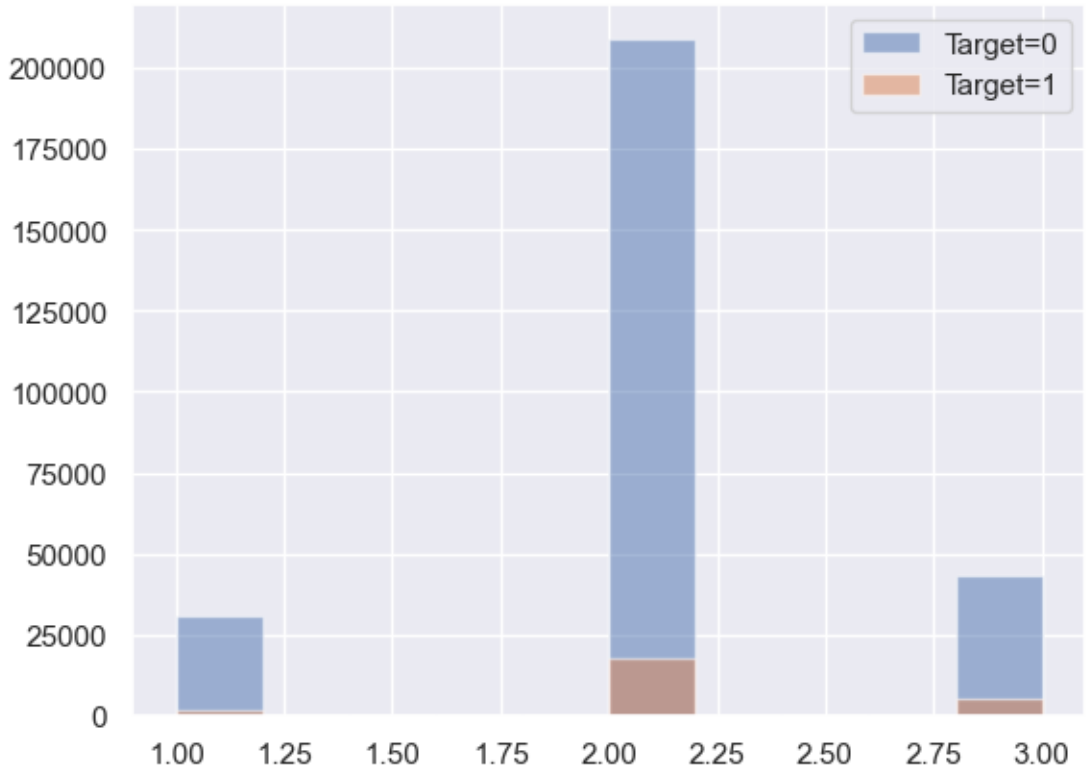


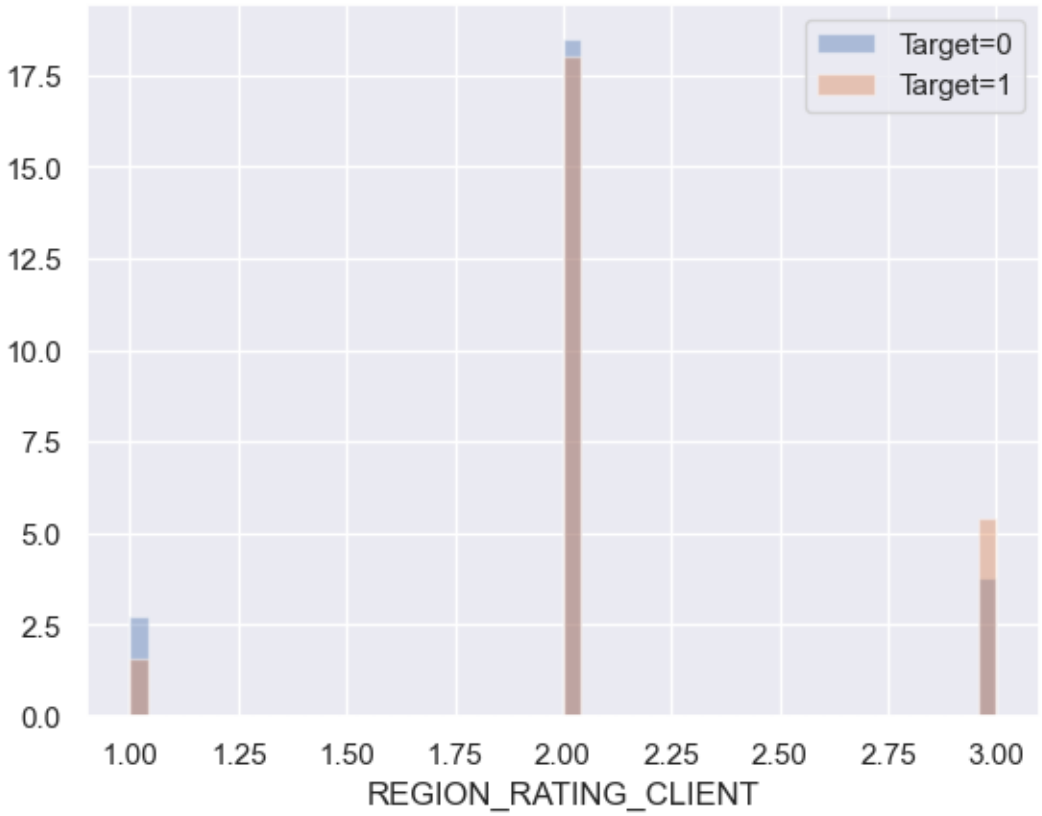
Plot of CNT_FAM_MEMBERS


Target=0
Target=1
200000
175000
150000
125000
100000
75000
50000
25000
0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

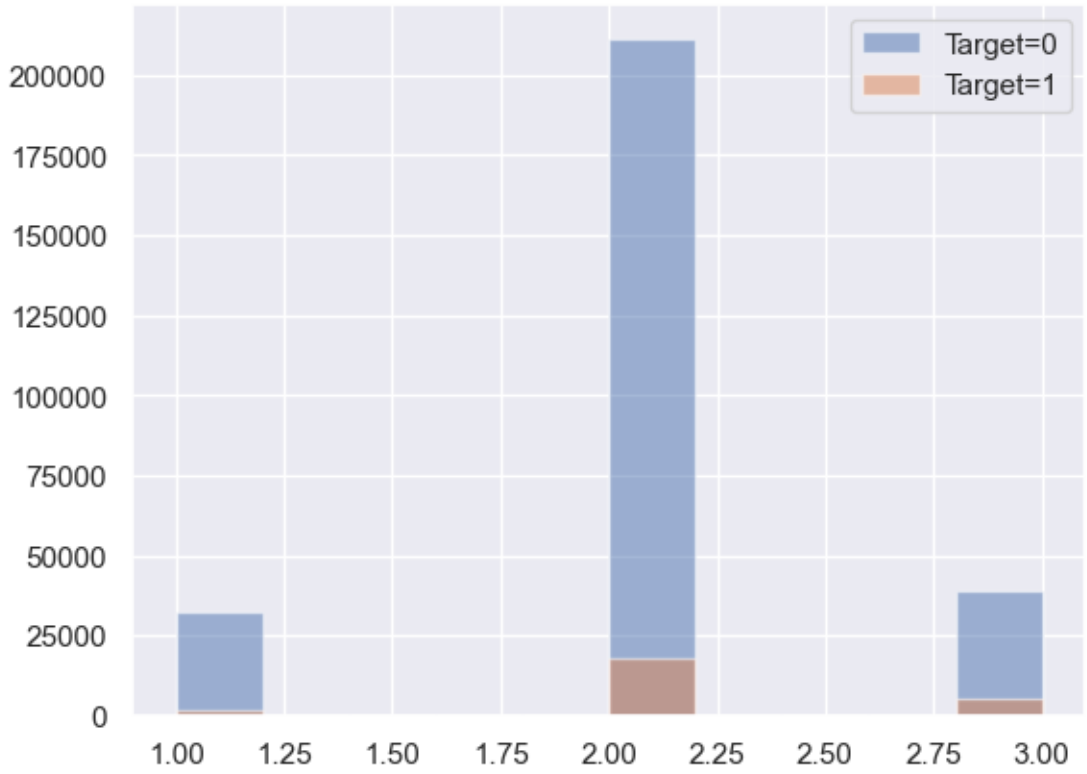


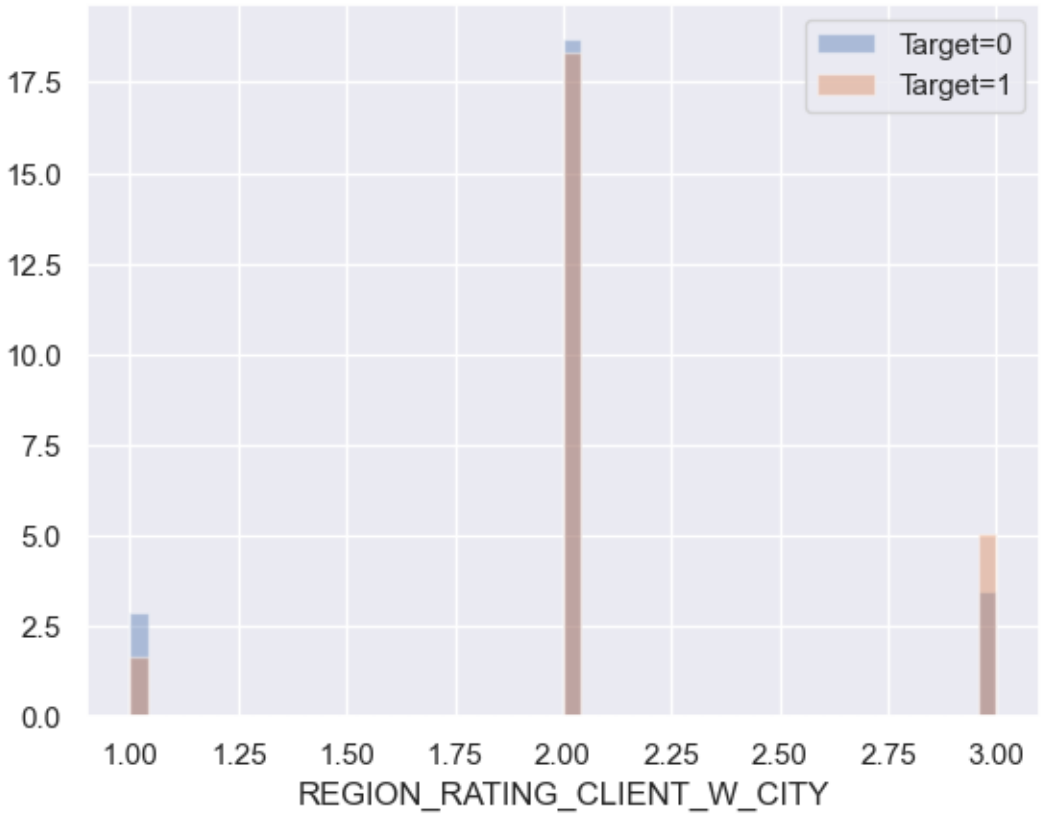
Plot of REGION_RATING_CLIENT


Target=0
Target=1
200000
175000
150000
125000
100000
75000
50000
25000
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

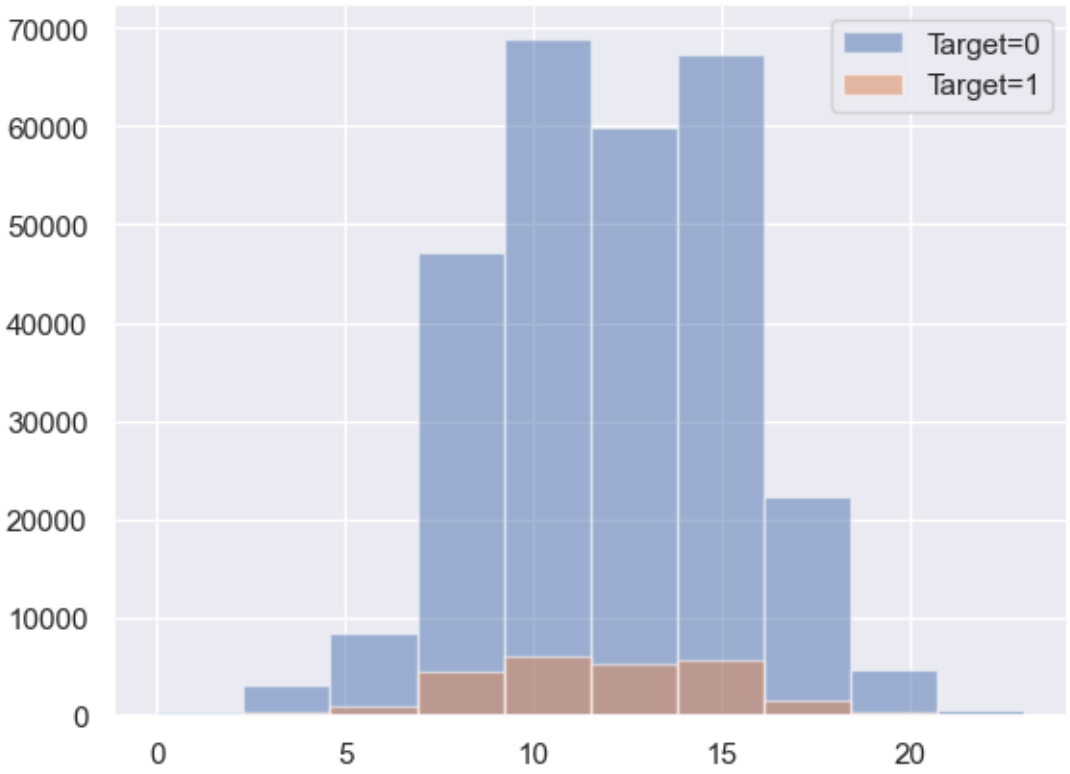


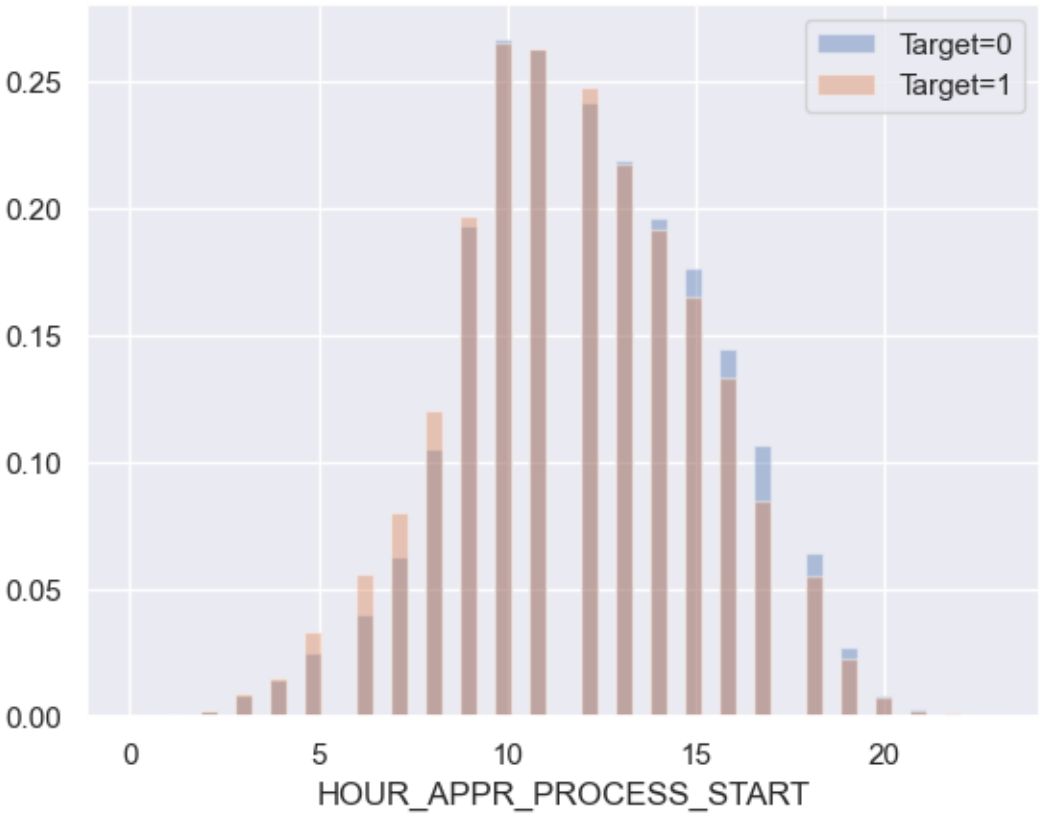
Plot of REGION_RATING_CLIENT_W_CITY


Target=0
Target=1
200000
175000
150000
125000
100000
75000
50000
25000
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

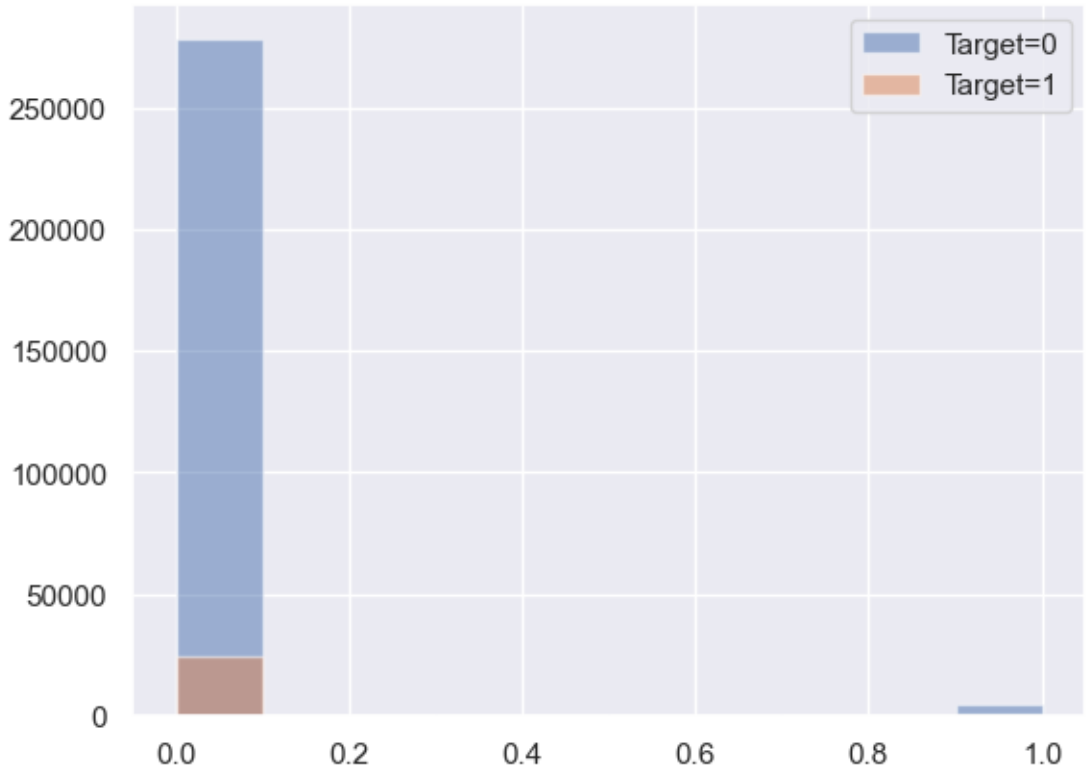


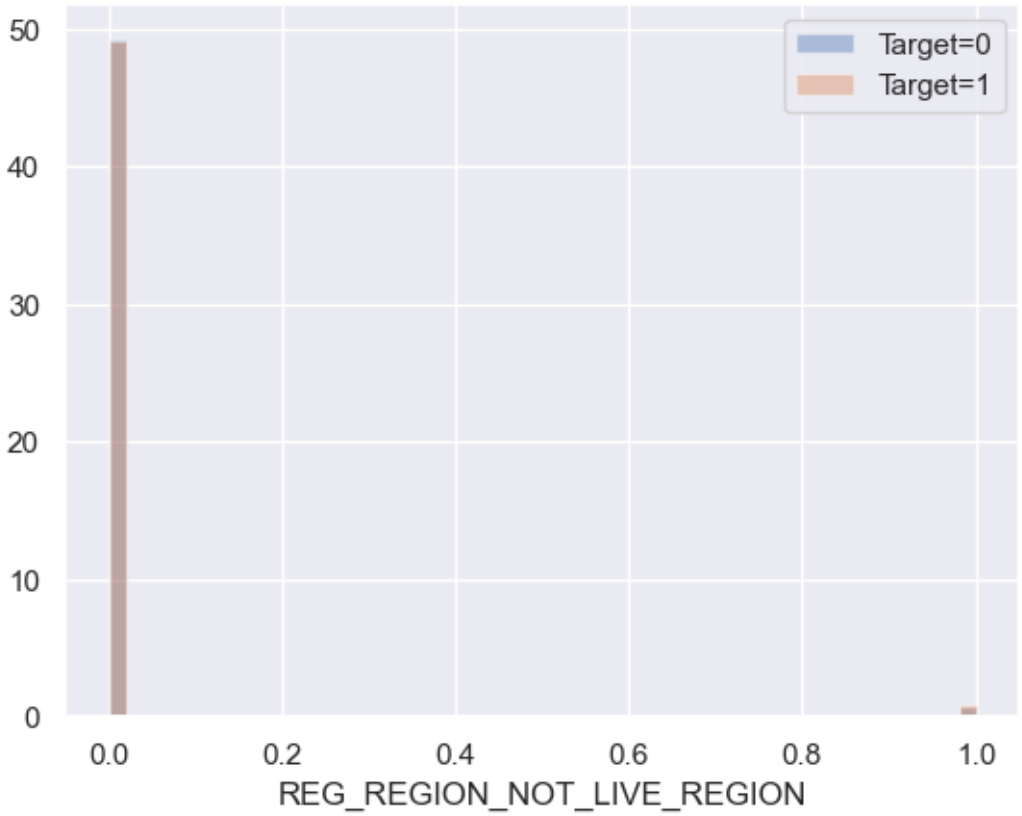
Plot of HOUR_APPR_PROCESS_START


70000
60000
50000
40000
30000
20000
10000
0
0
5
10
15
20
Target=0
Target=1

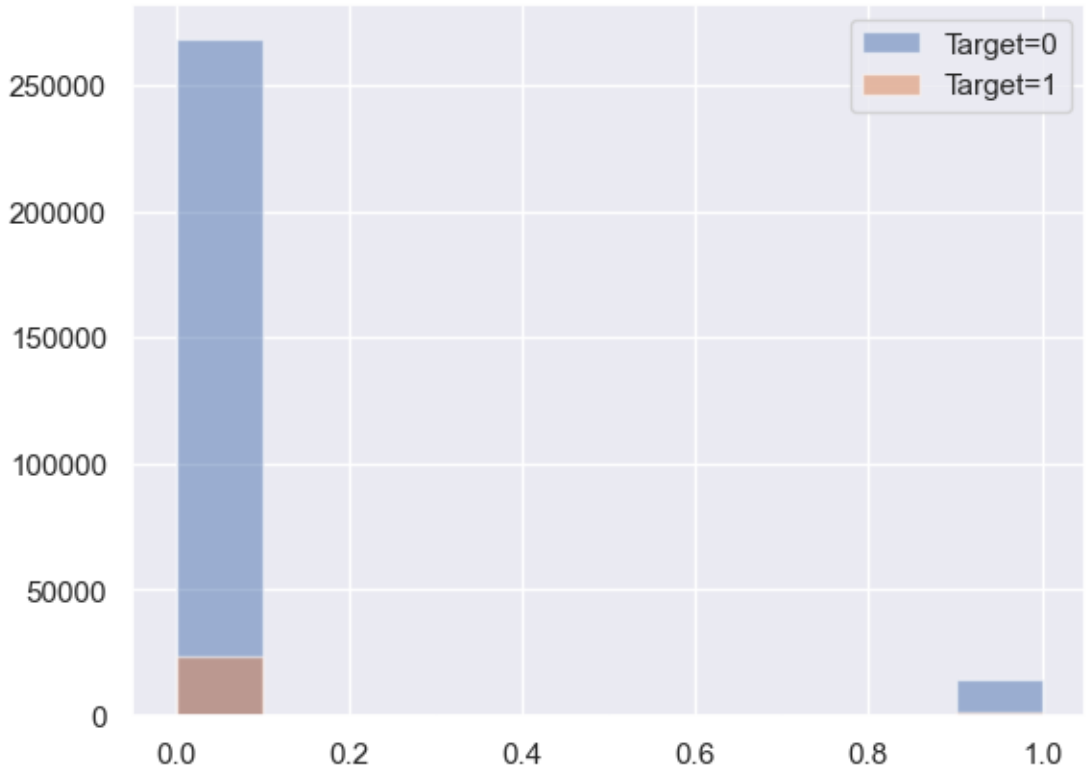


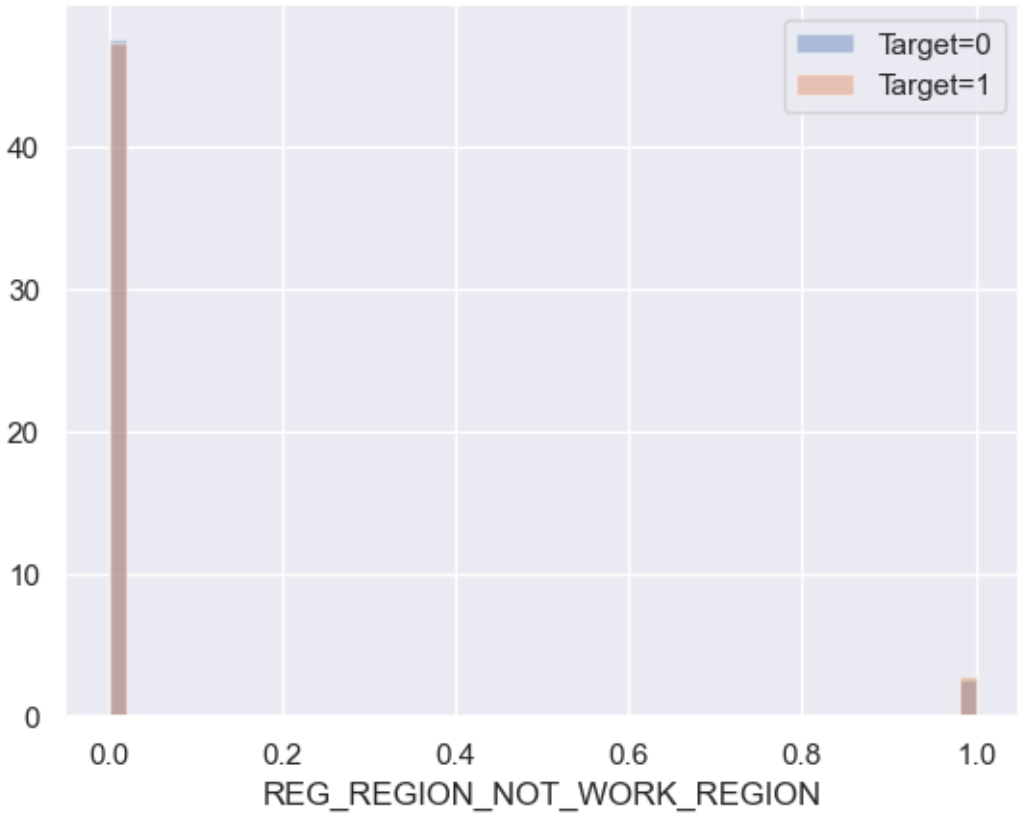
Plot of REG_REGION_NOT_LIVE_REGION


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

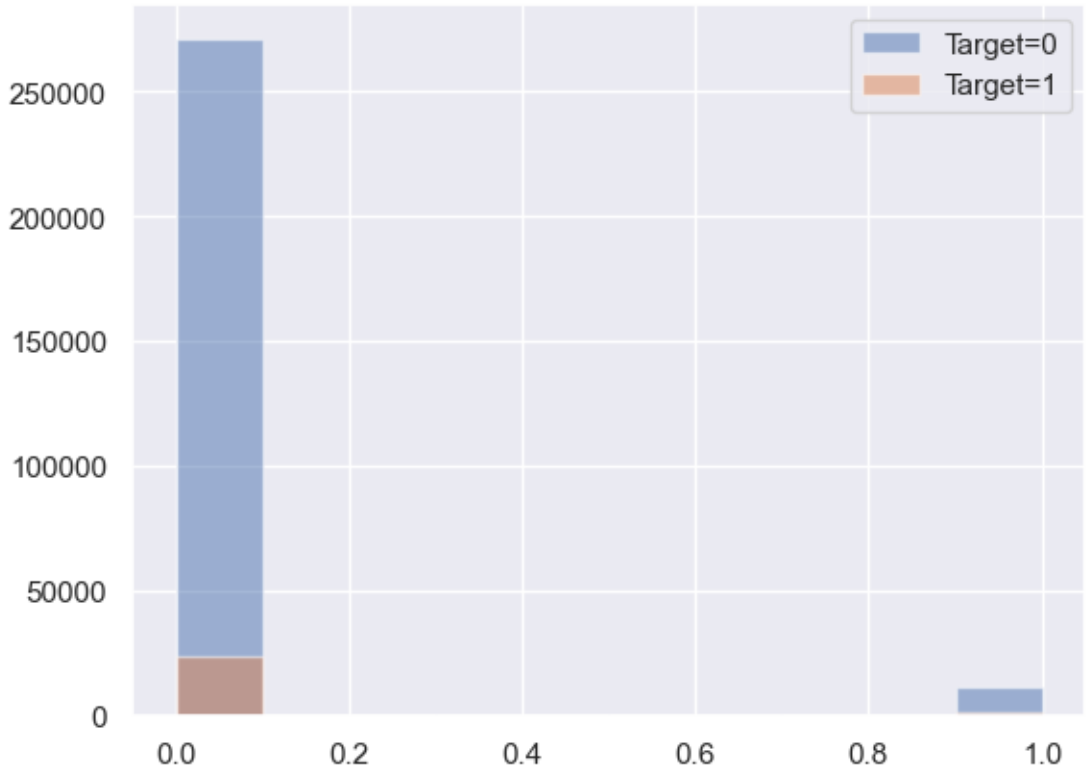


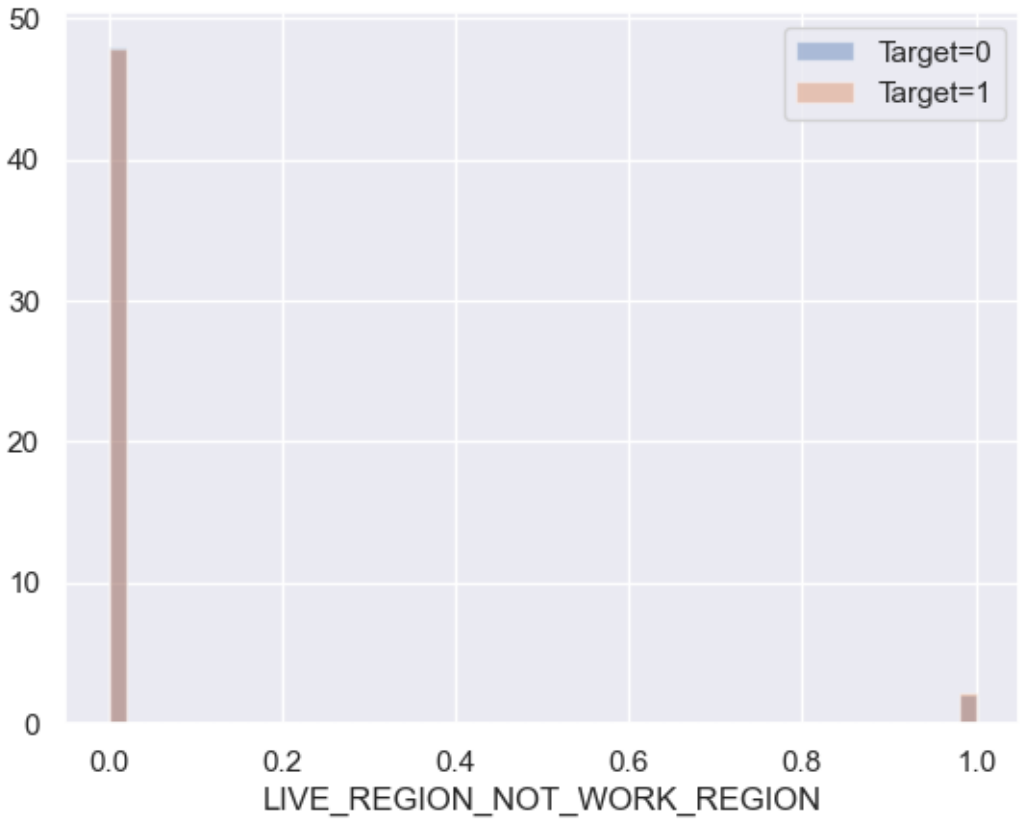
Plot of REG_REGION_NOT_WORK_REGION


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

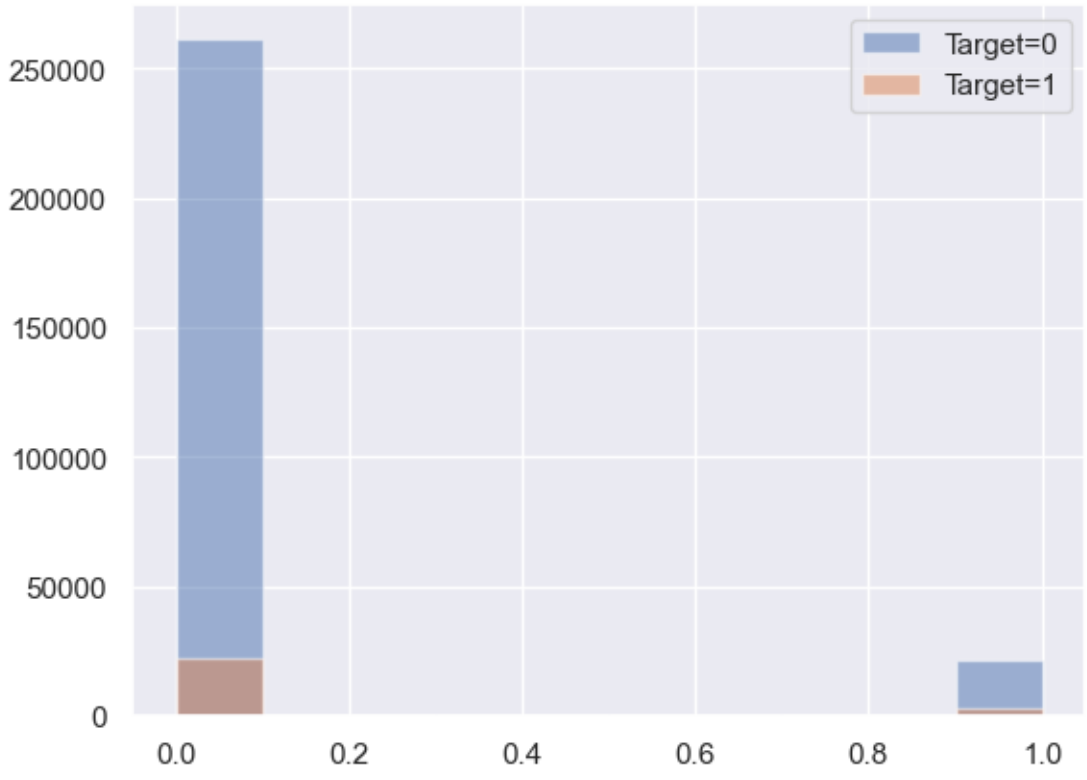


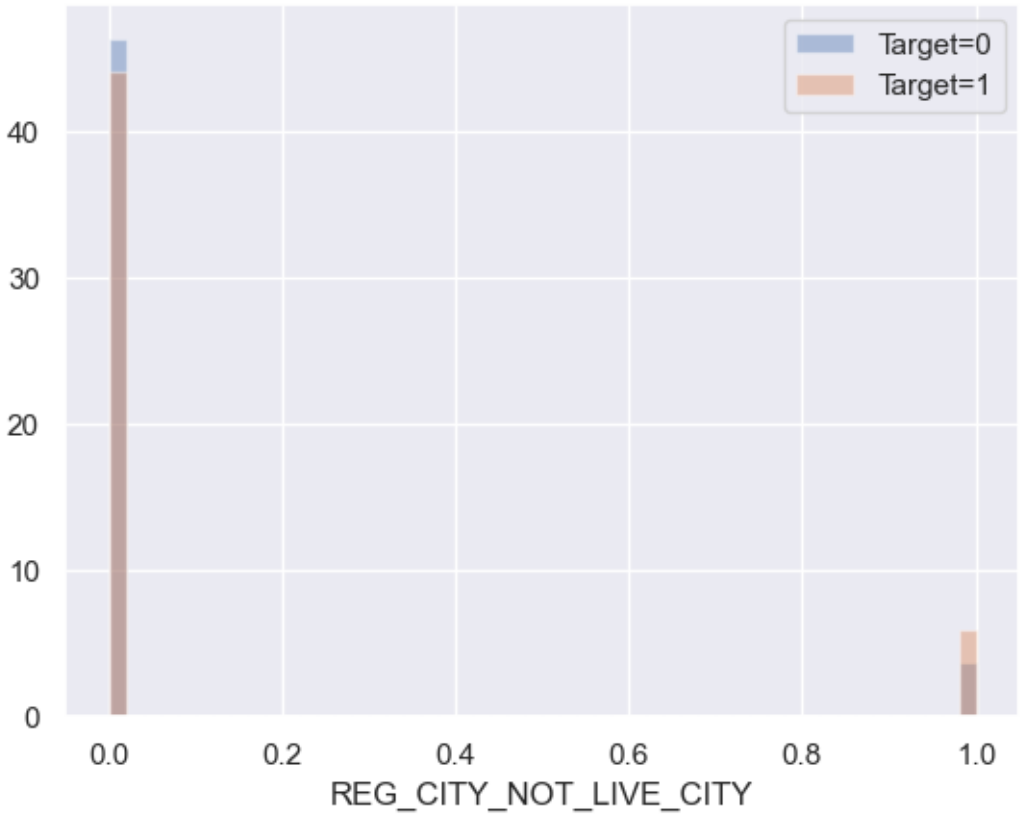

```
----------------------------------------------------------------------
Plot of LIVE_REGION_NOT_WORK_REGION
```


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

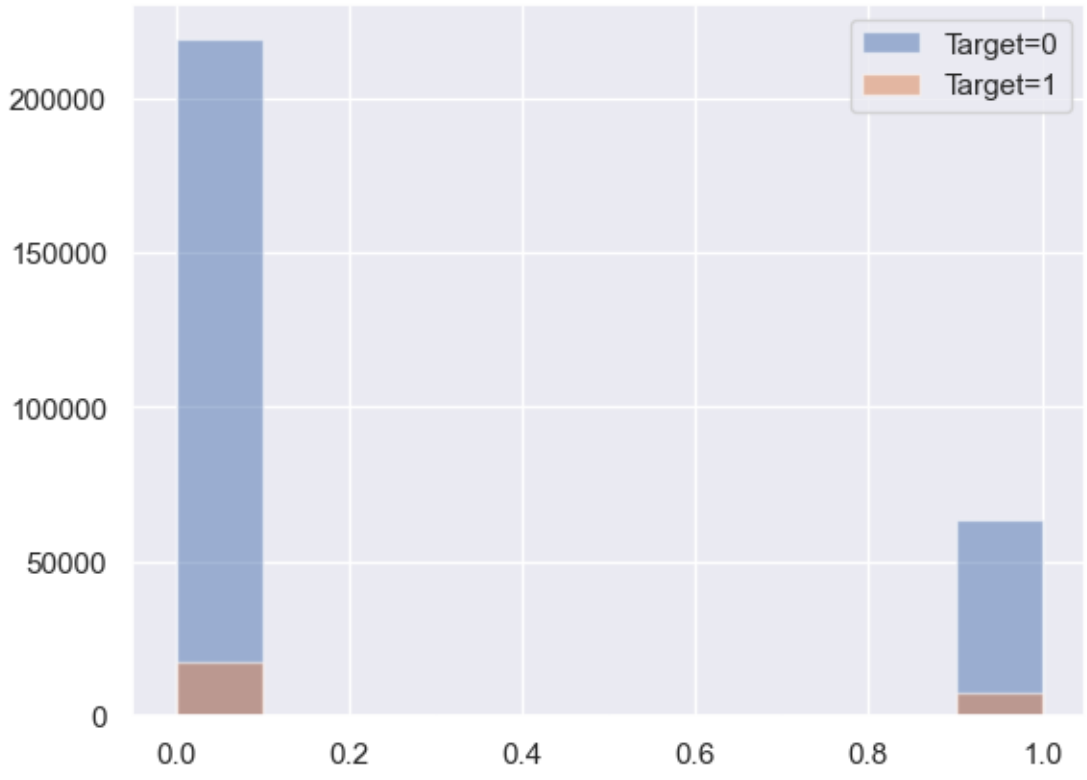


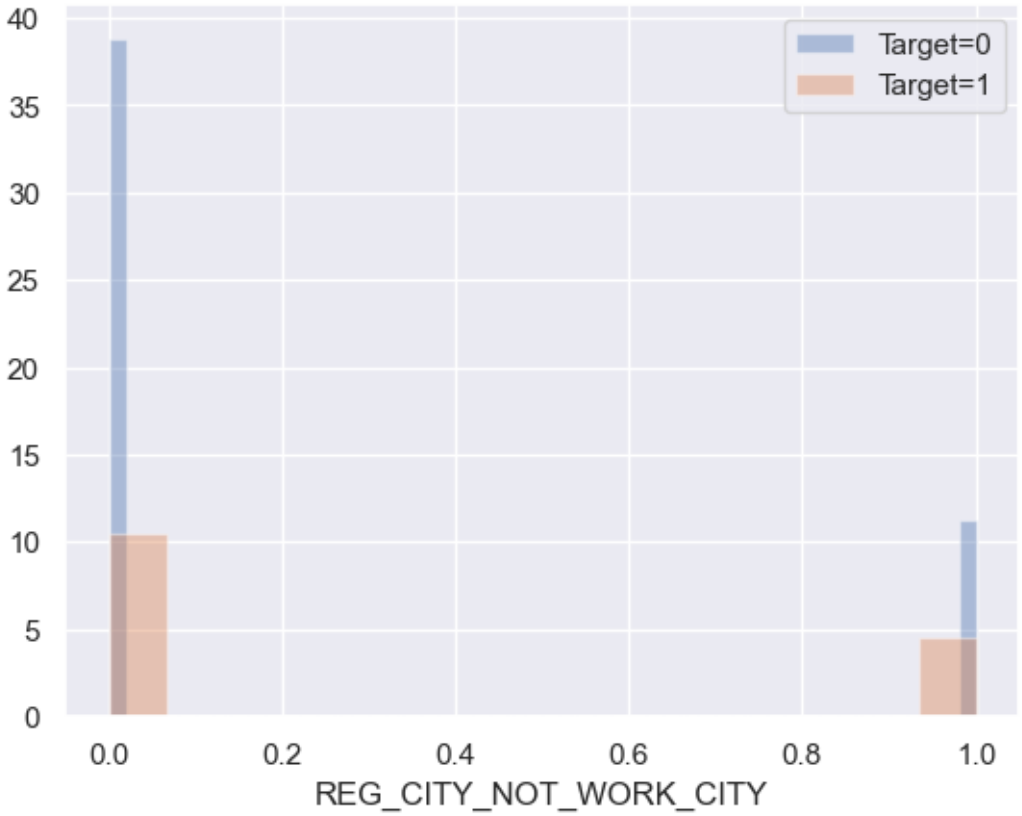
Plot of REG_CITY_NOT_LIVE_CITY


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

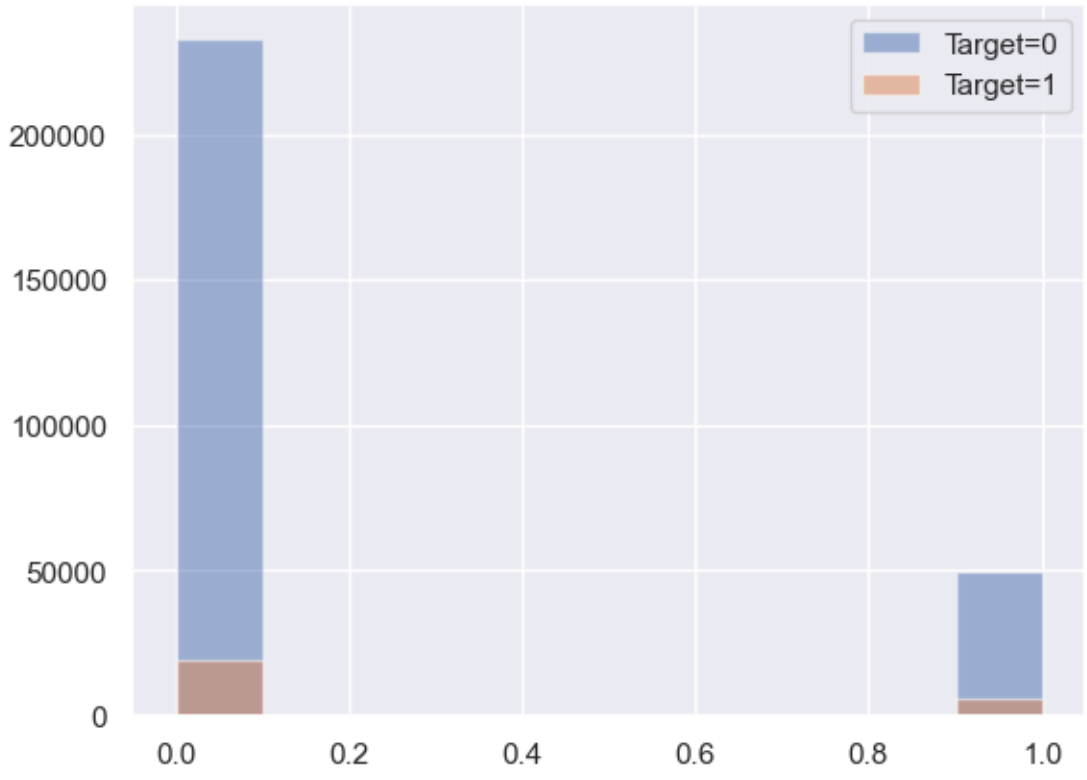


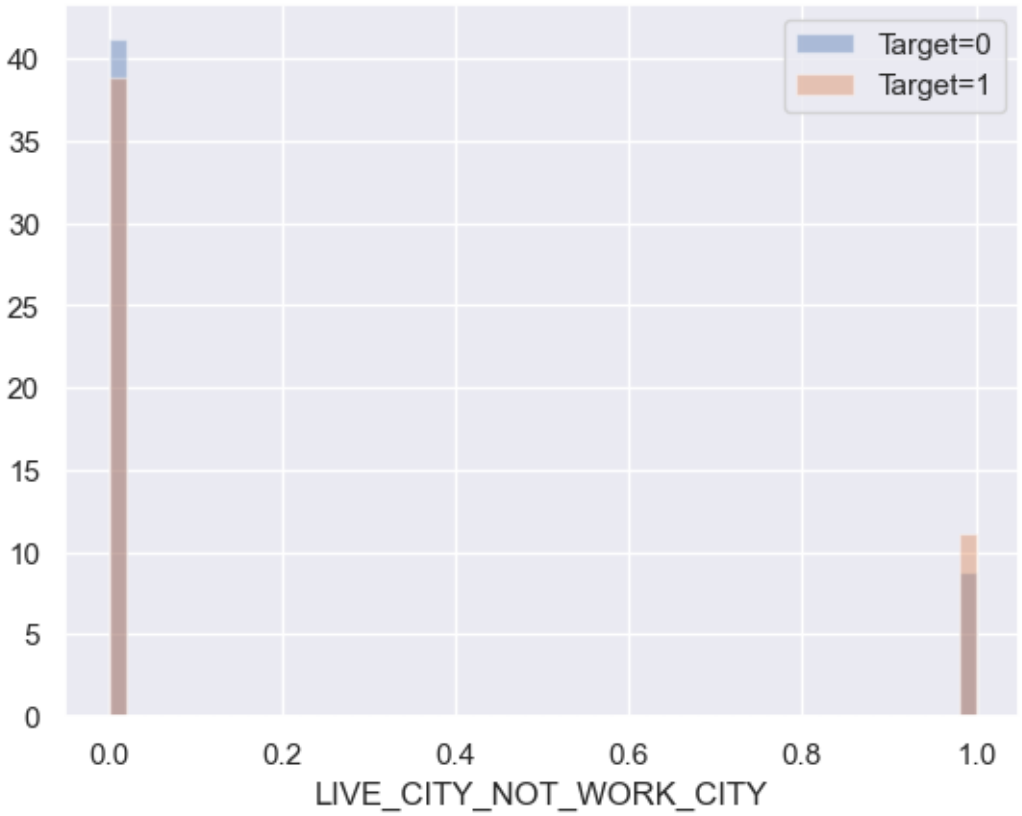
Plot of REG_CITY_NOT_WORK_CITY


Target=0
Target=1
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

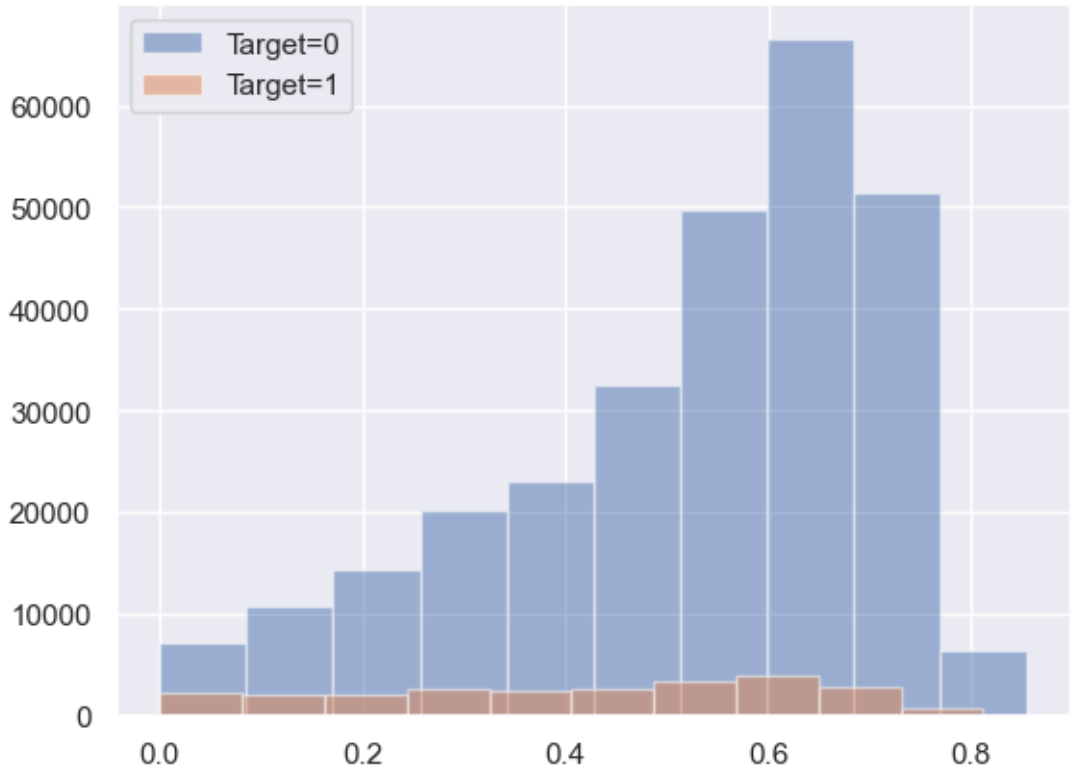


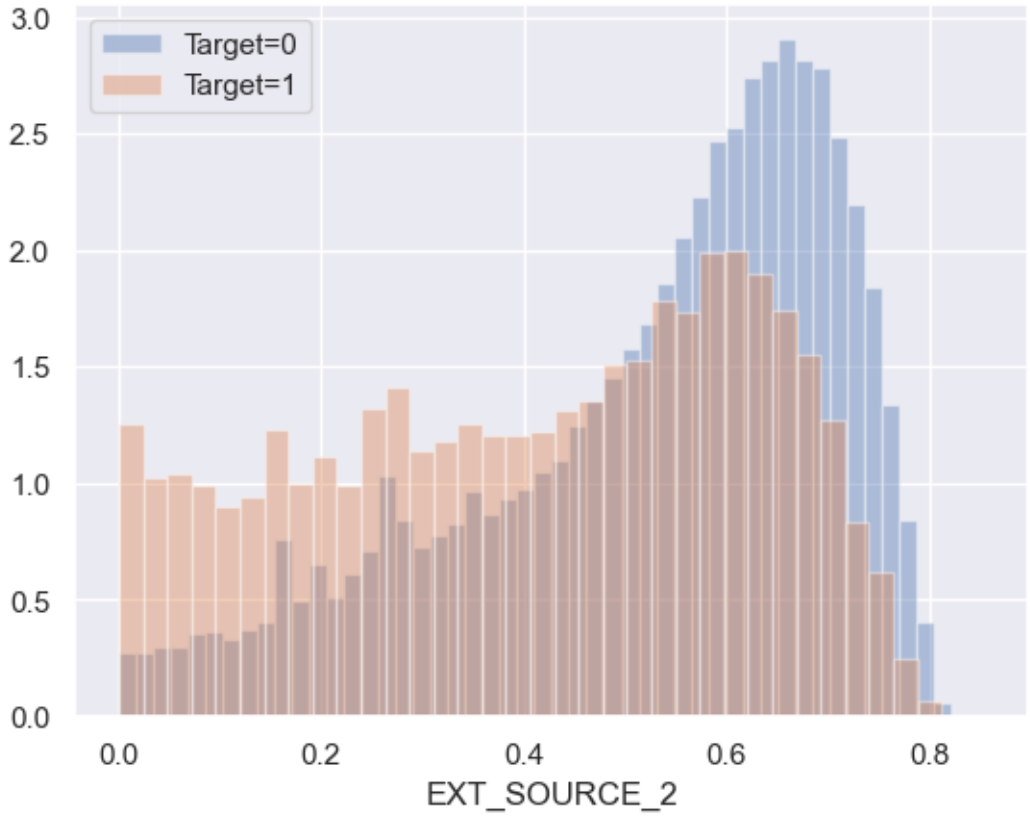
Plot of LIVE_CITY_NOT_WORK_CITY


Target=0
Target=1
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

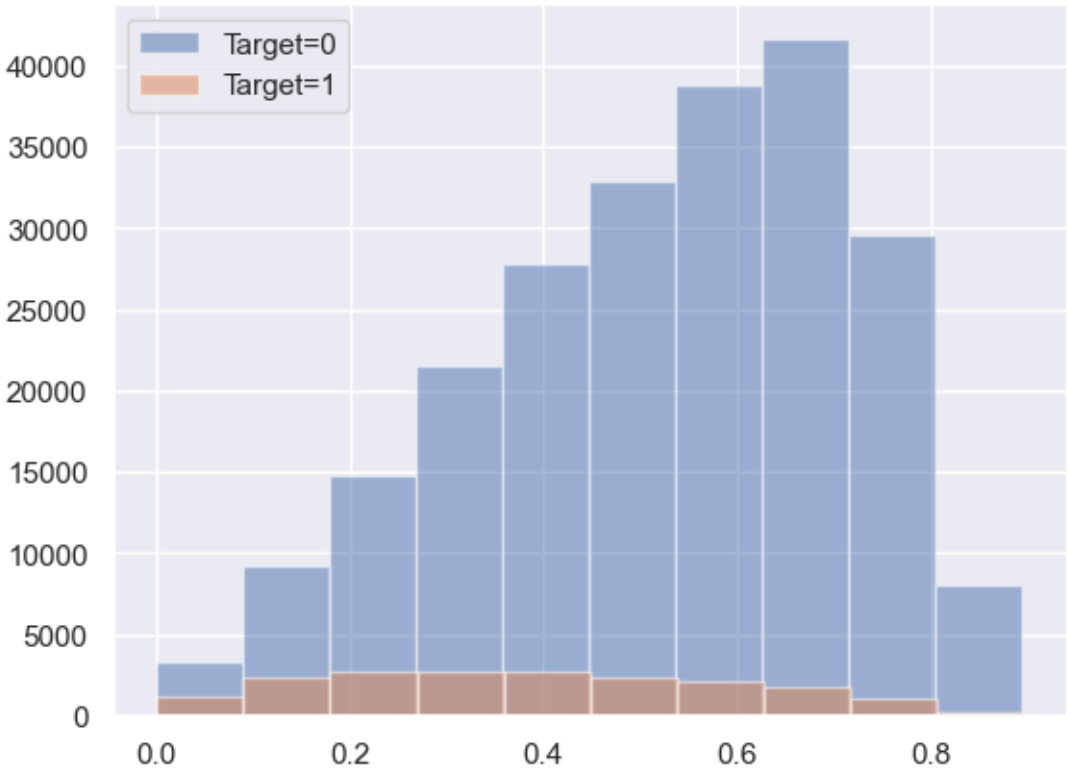


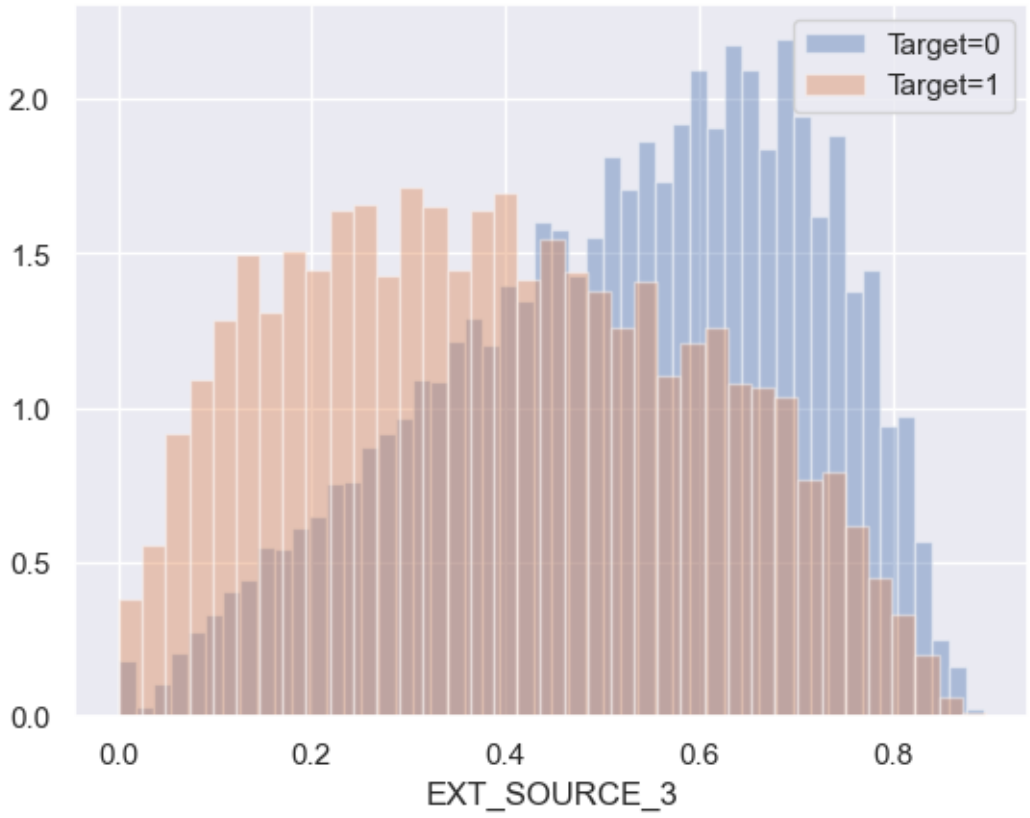
Plot of EXT_SOURCE_2


Target=0
Target=1
60000
50000
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8

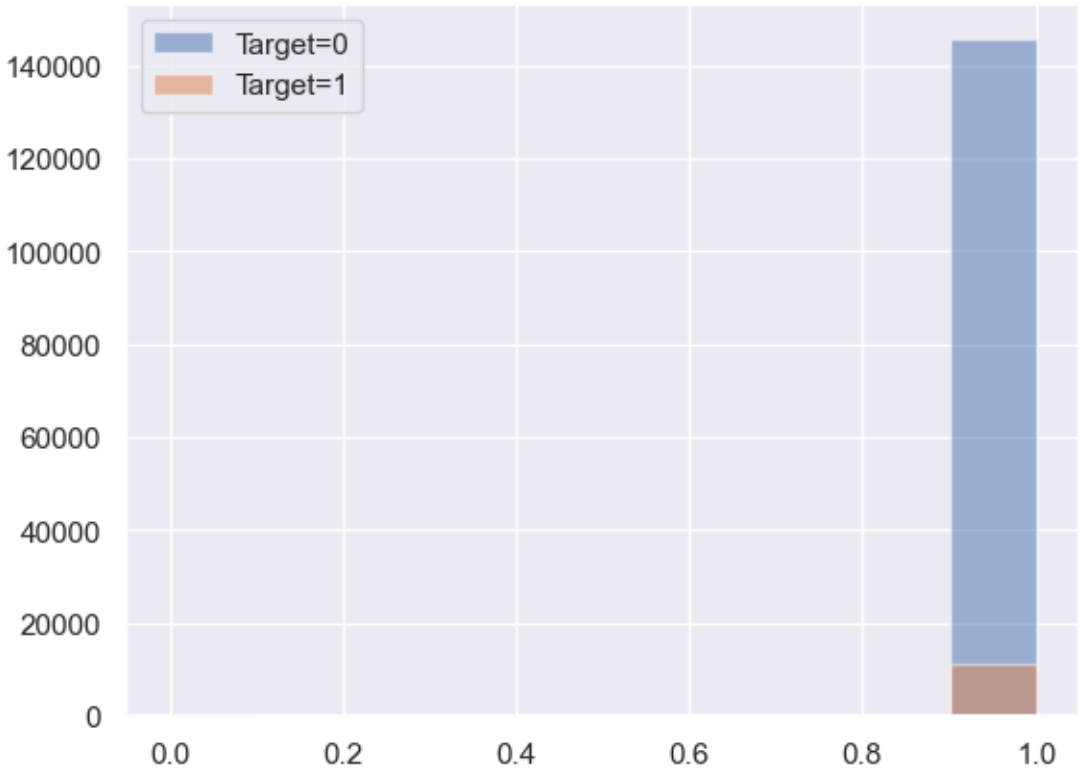


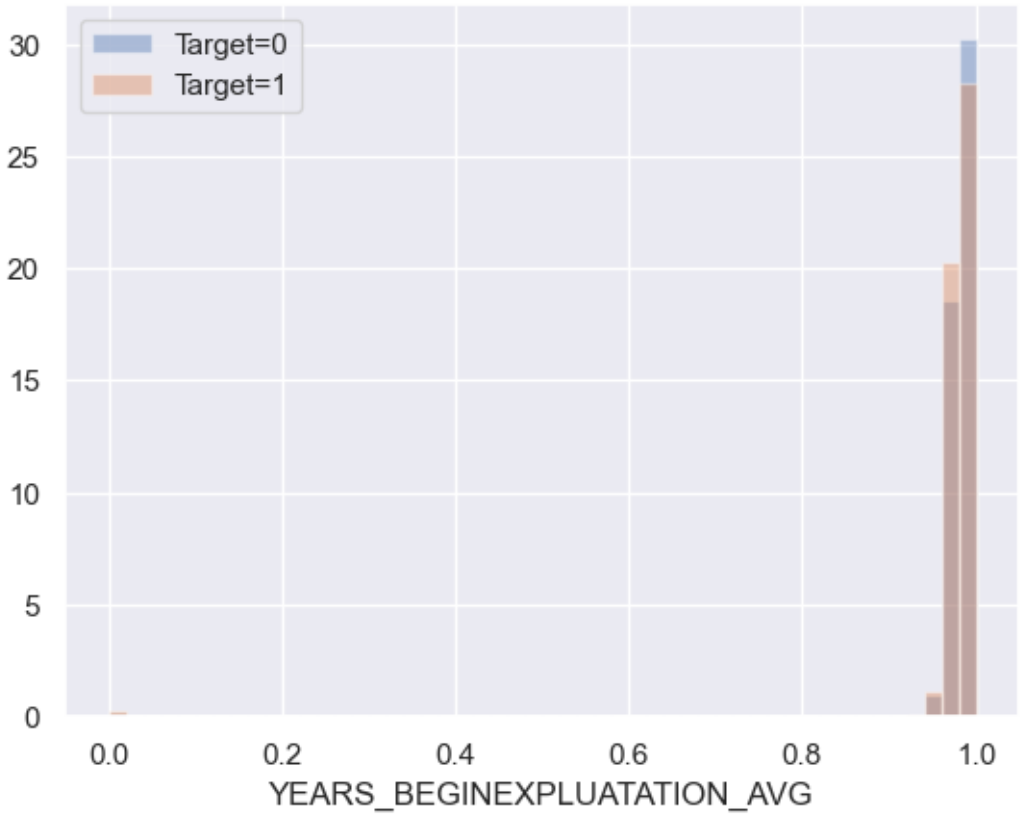
Plot of EXT_SOURCE_3


Target=0
Target=1
40000
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8

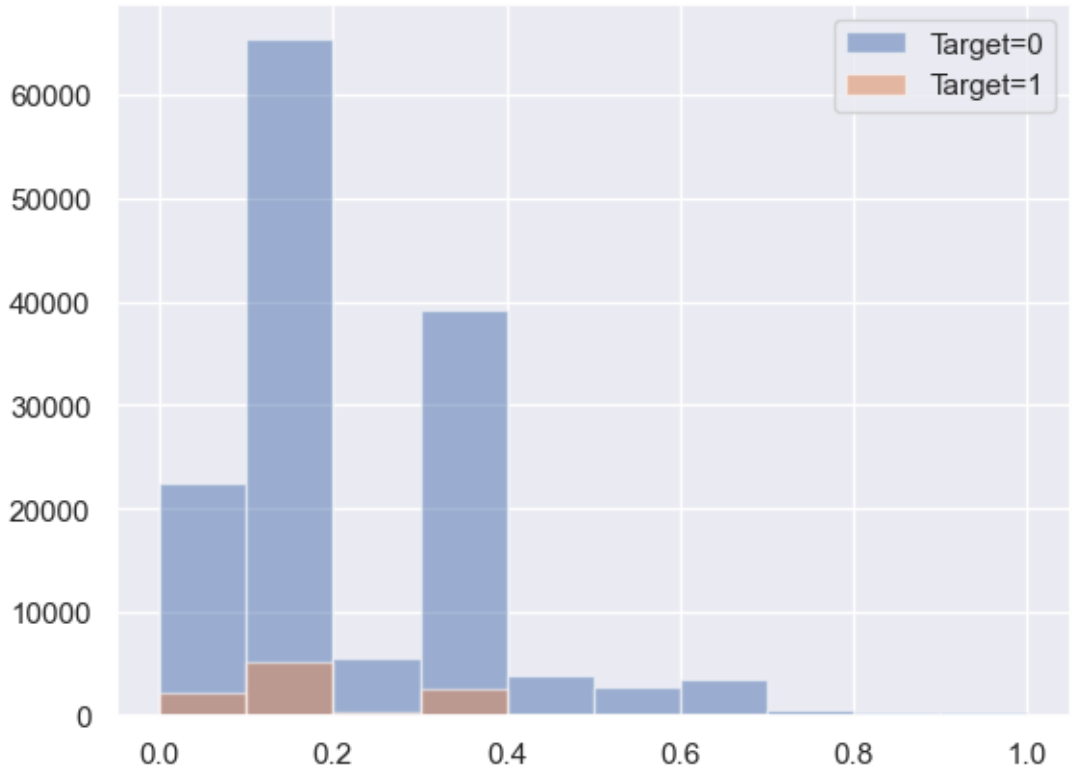


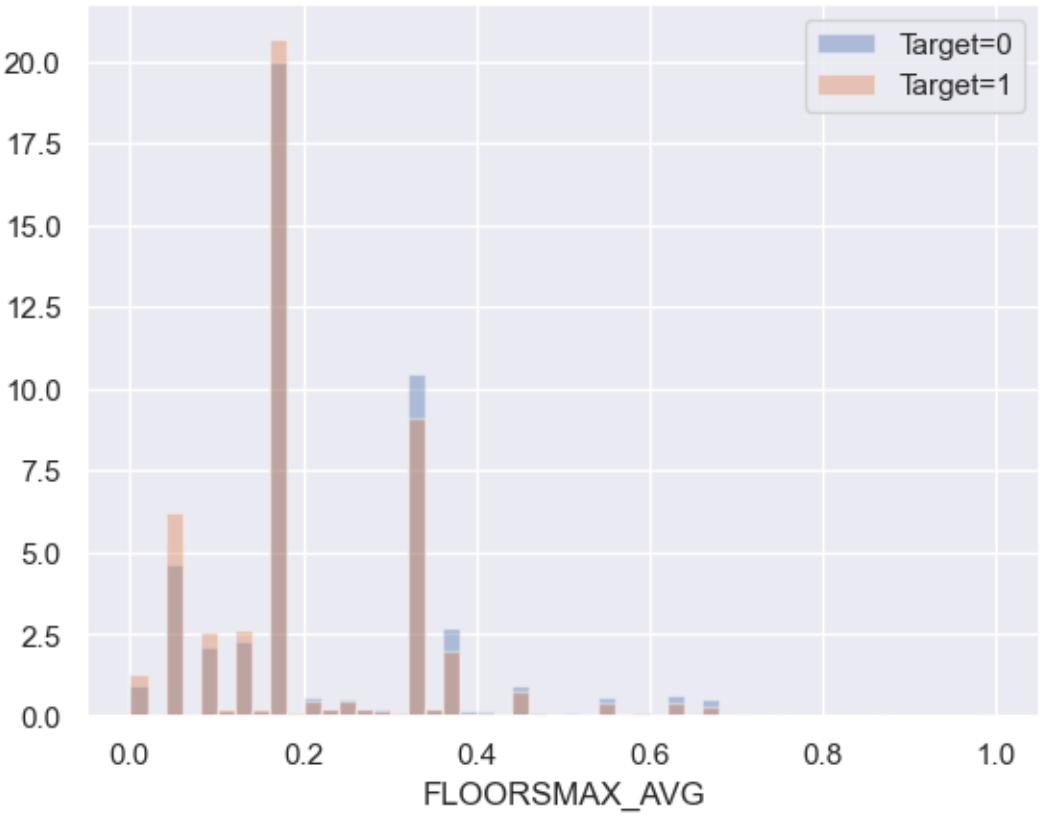
Plot of YEARS_BEGINEXPLUATATION_AVG


Target=0
Target=1
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0

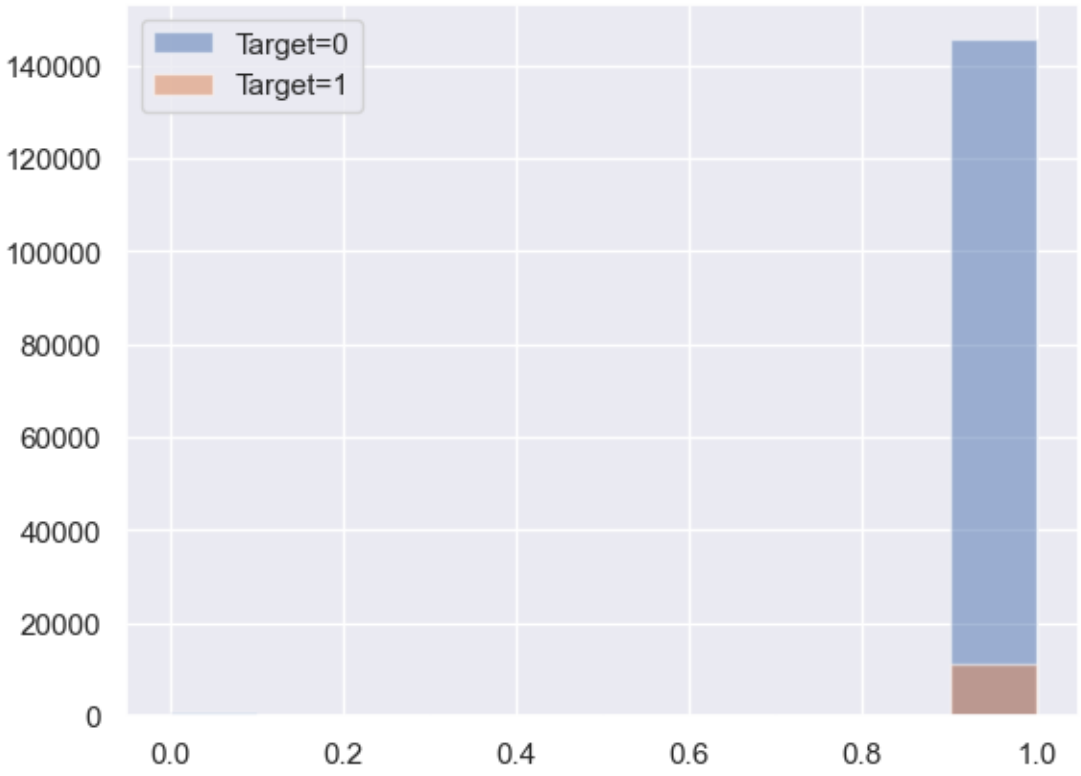


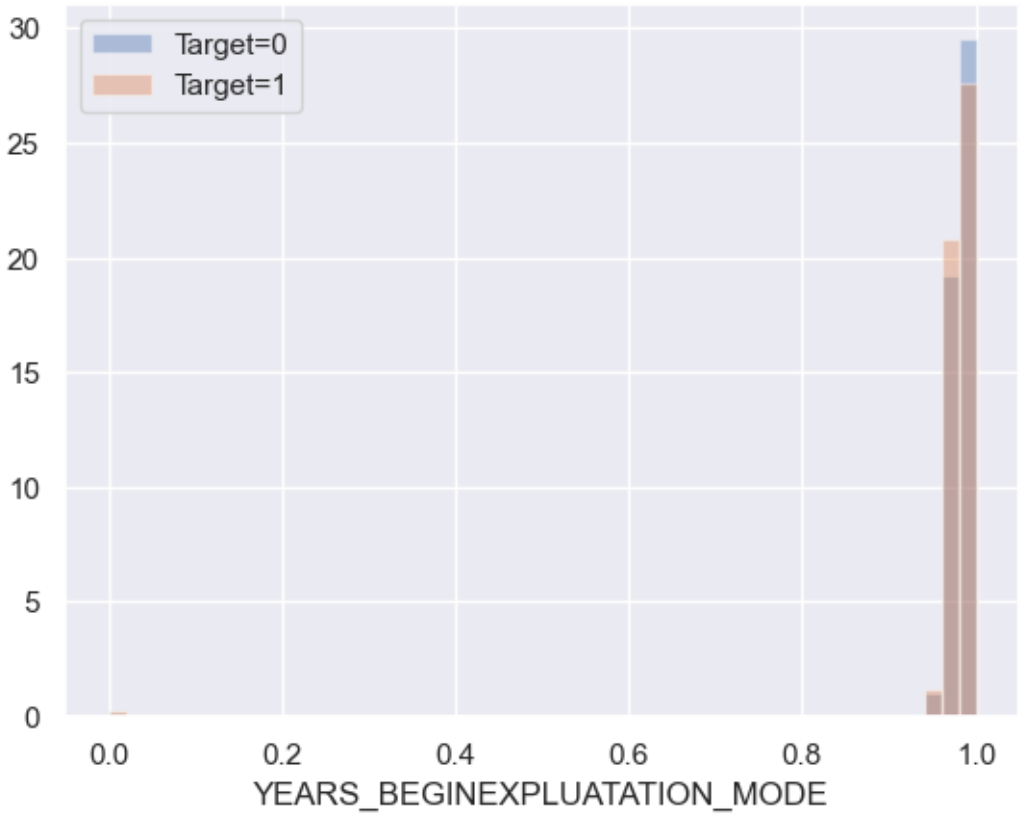
Plot of FLOORSMAX_AVG


Target=0
Target=1
60000
50000
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0

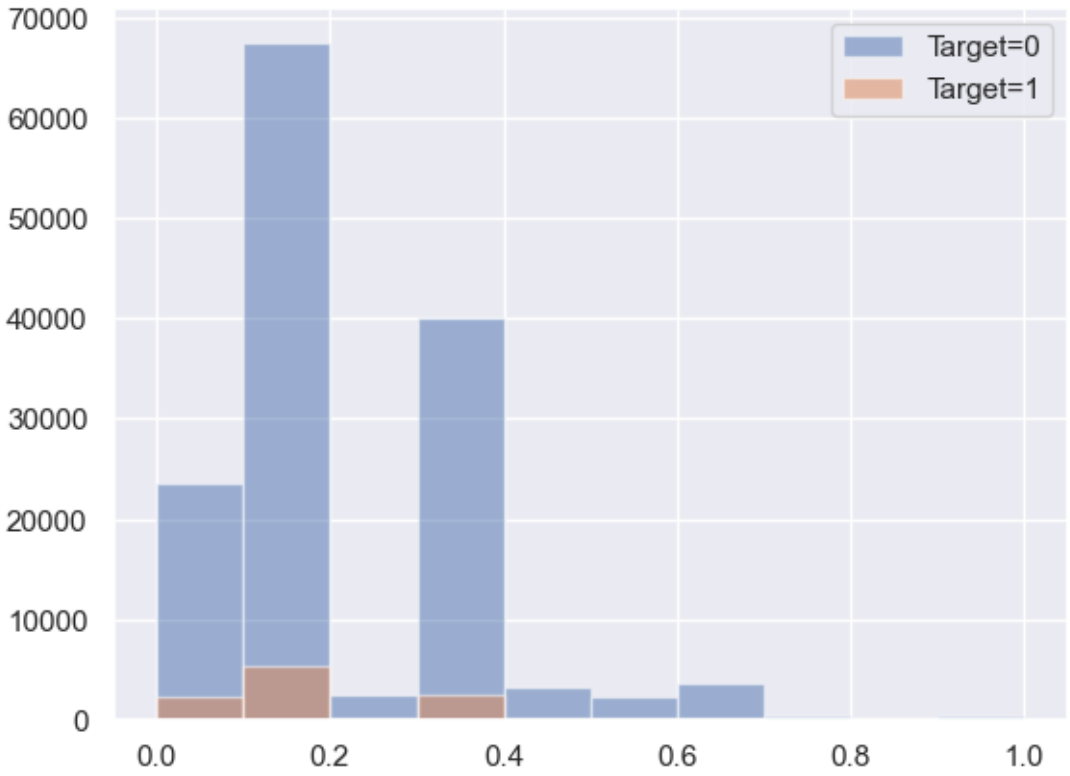


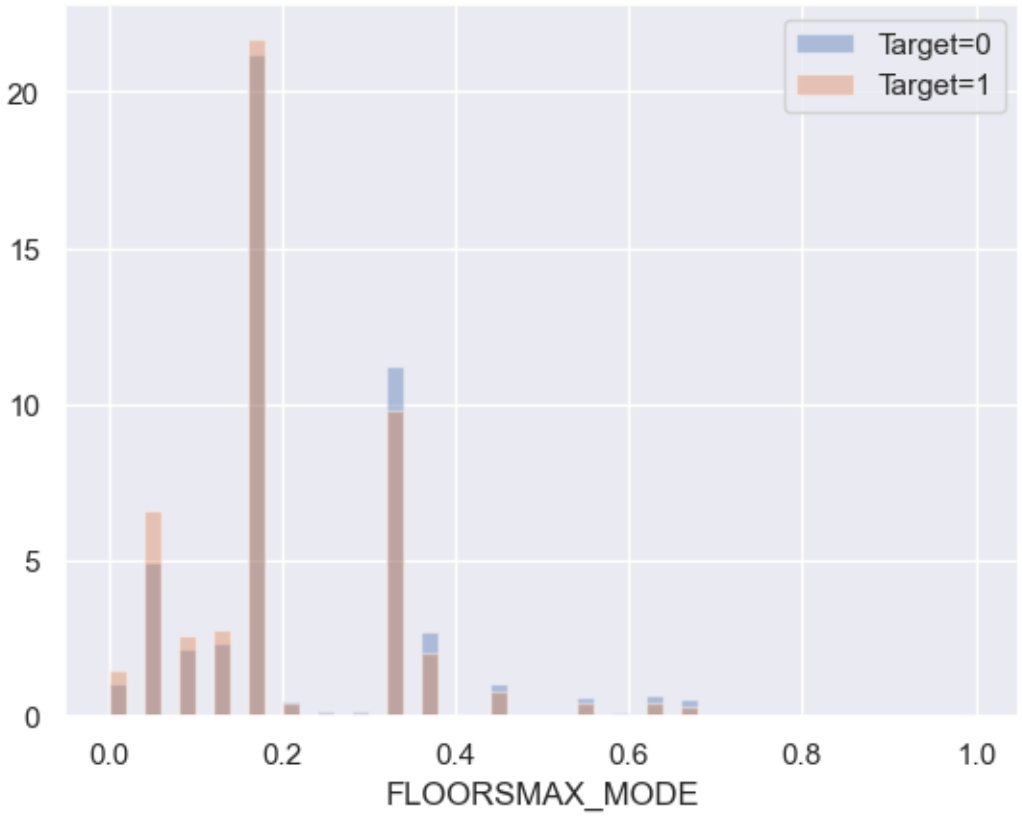
Plot of YEARS_BEGINEXPLUATATION_MODE


Target=0
Target=1
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0

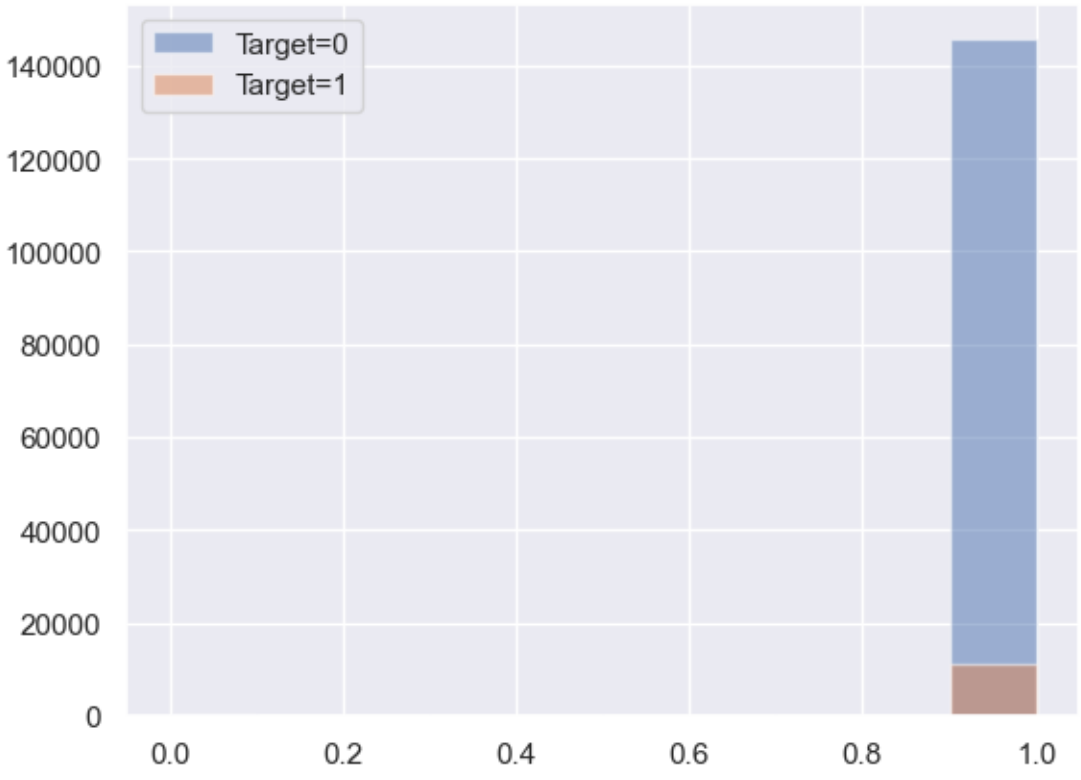


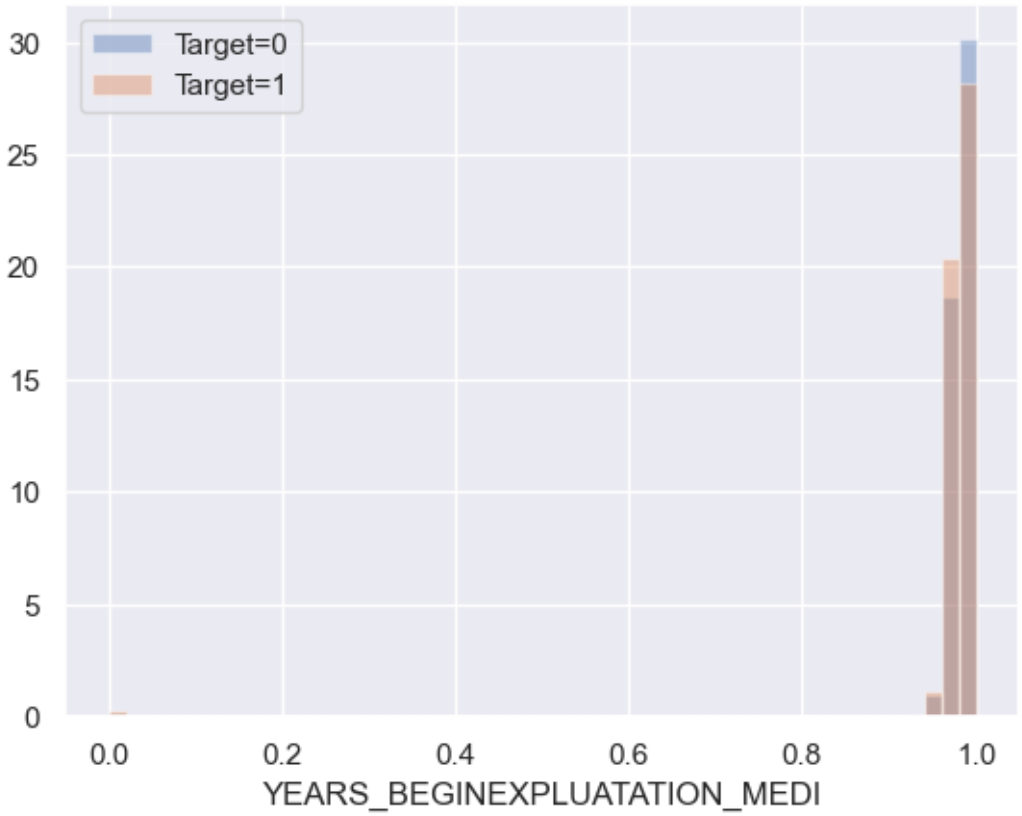
Plot of FLOORSMAX_MODE


70000
60000
50000
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0
Target=0
Target=1

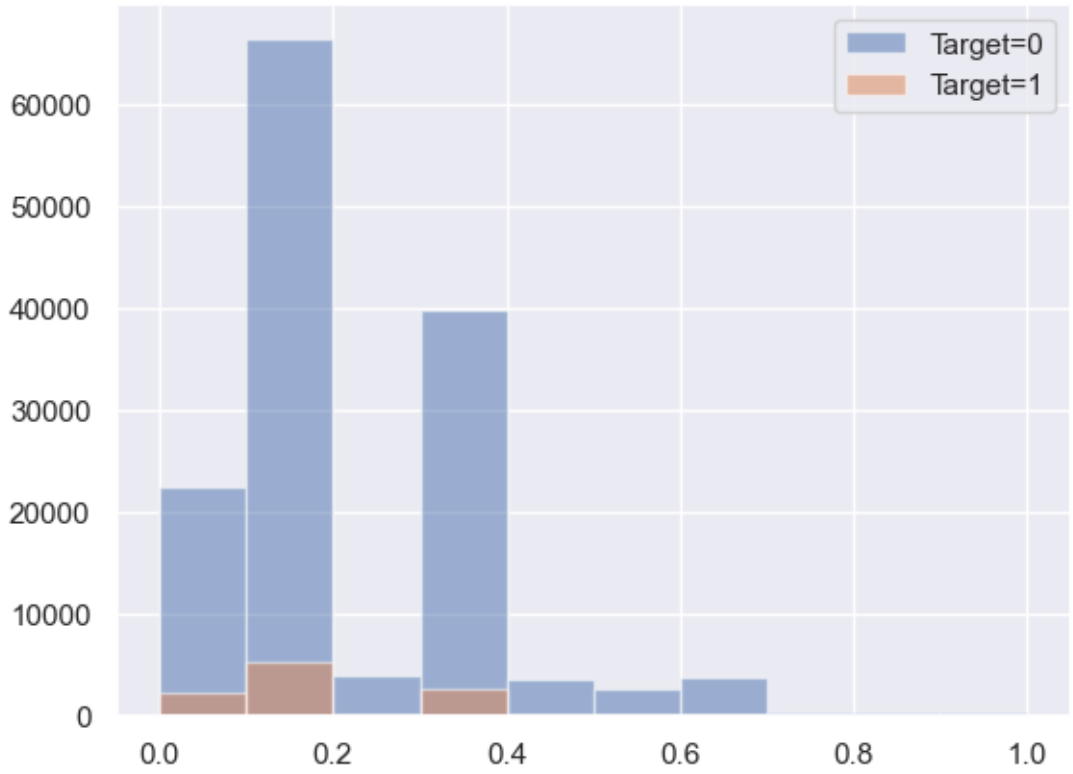


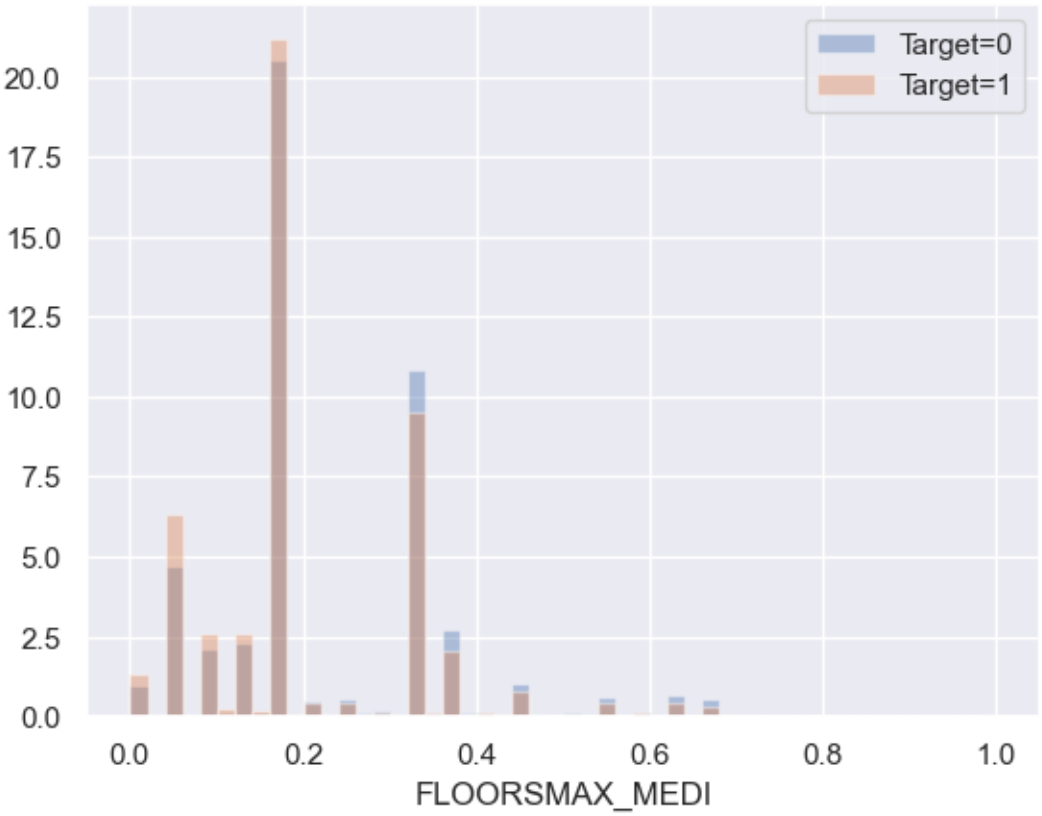
Plot of YEARS_BEGINEXPLUATATION_MEDI


Target=0
Target=1
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0

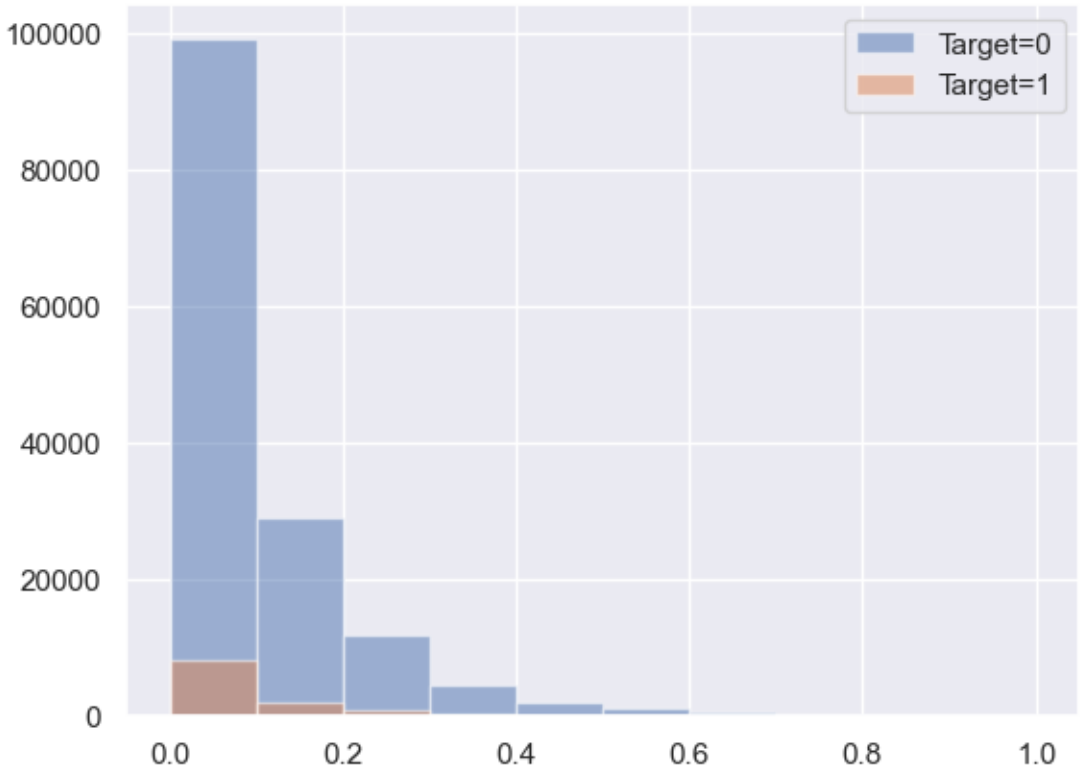


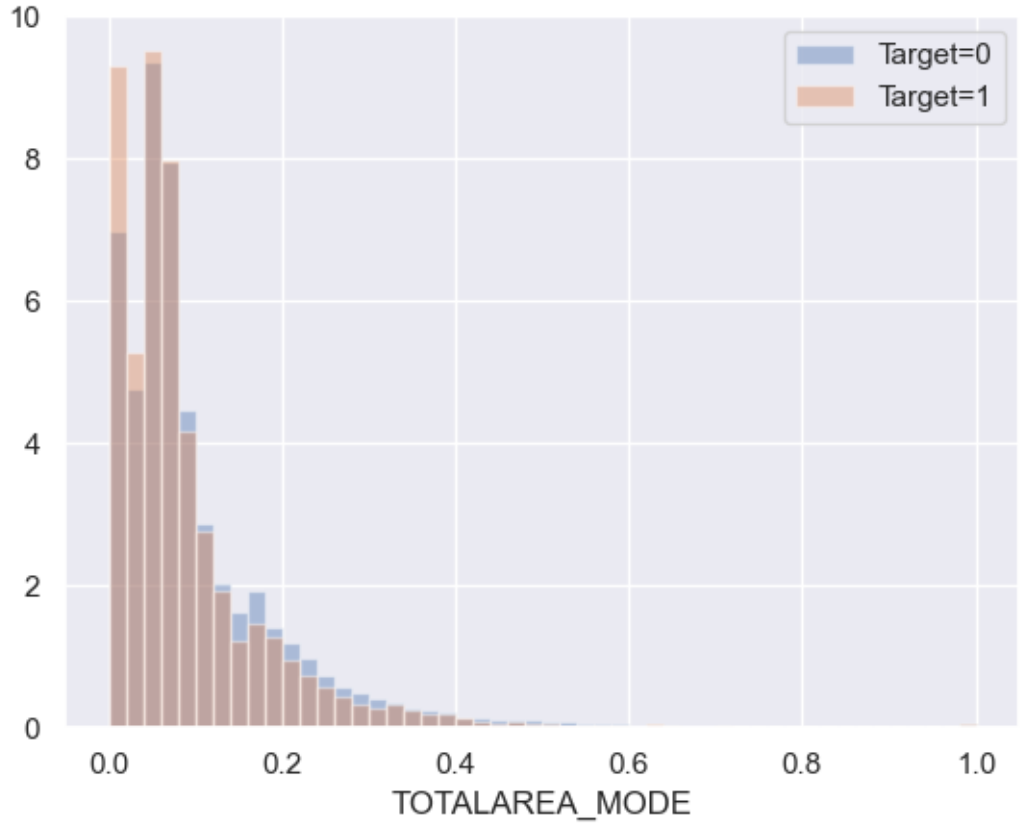
Plot of FLOORSMAX_MEDI


Target=0
Target=1
60000
50000
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0

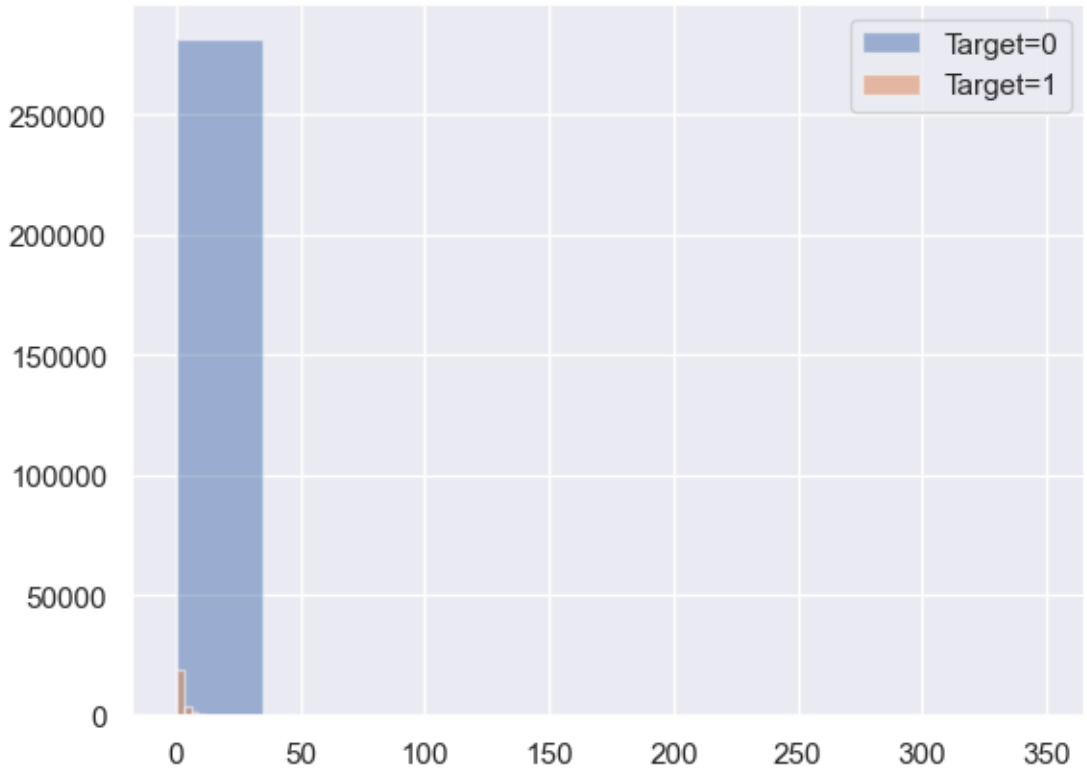


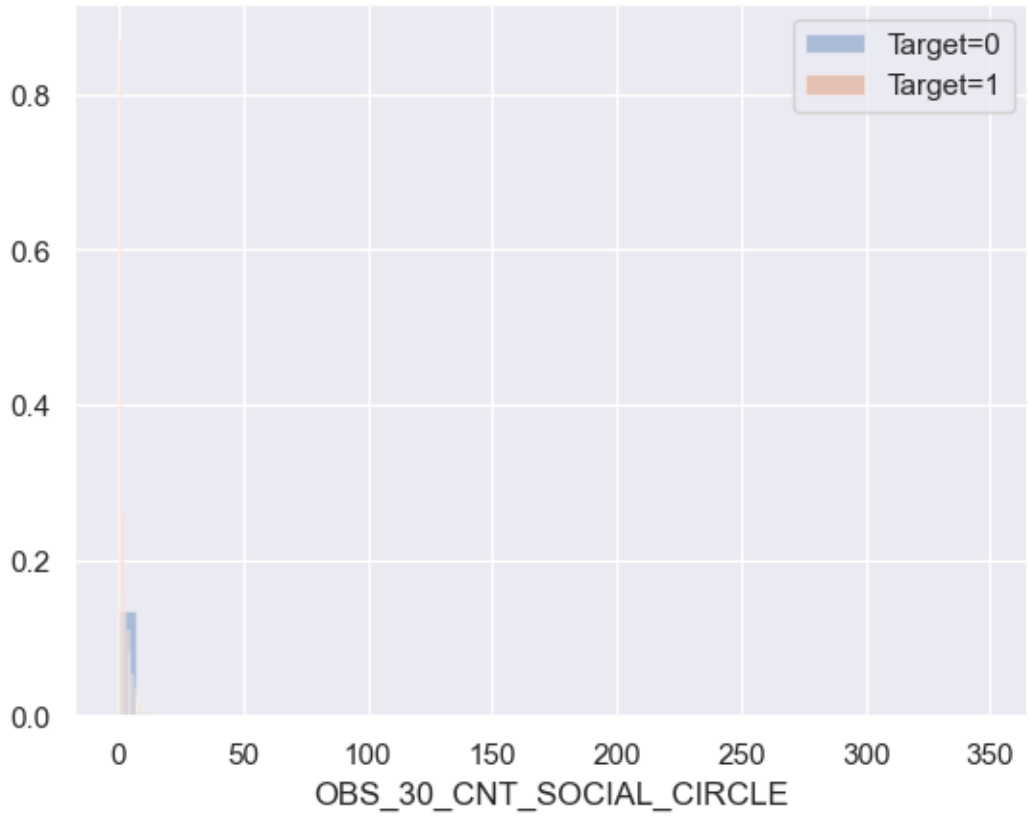
Plot of TOTALAREA_MODE


100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0
Target=0
Target=1

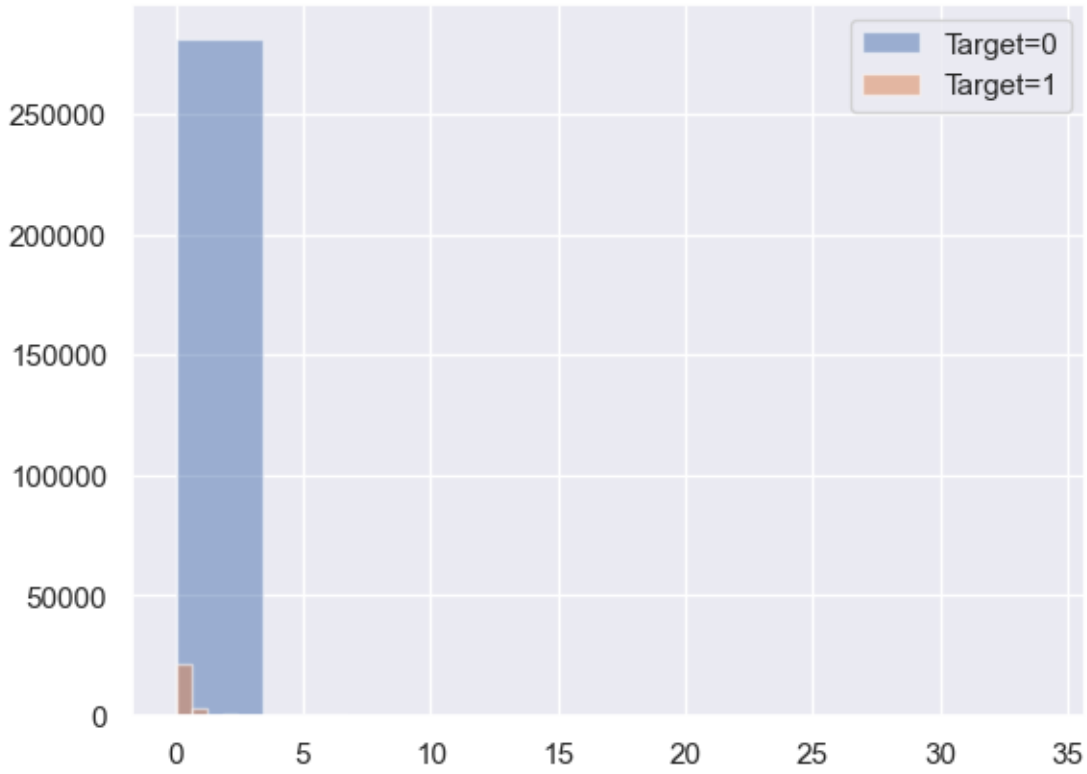


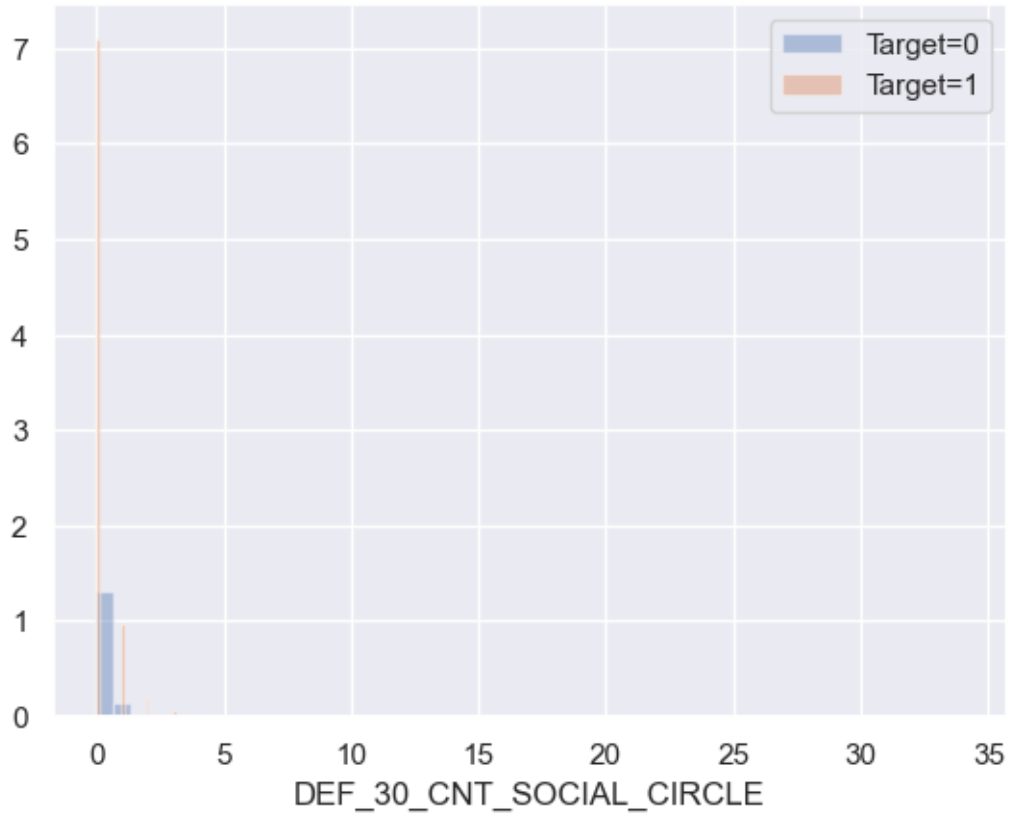
Plot of OBS_30_CNT_SOCIAL_CIRCLE


Target=0
Target=1
250000
200000
150000
100000
50000
0
0
50
100
150
200
250
300
350

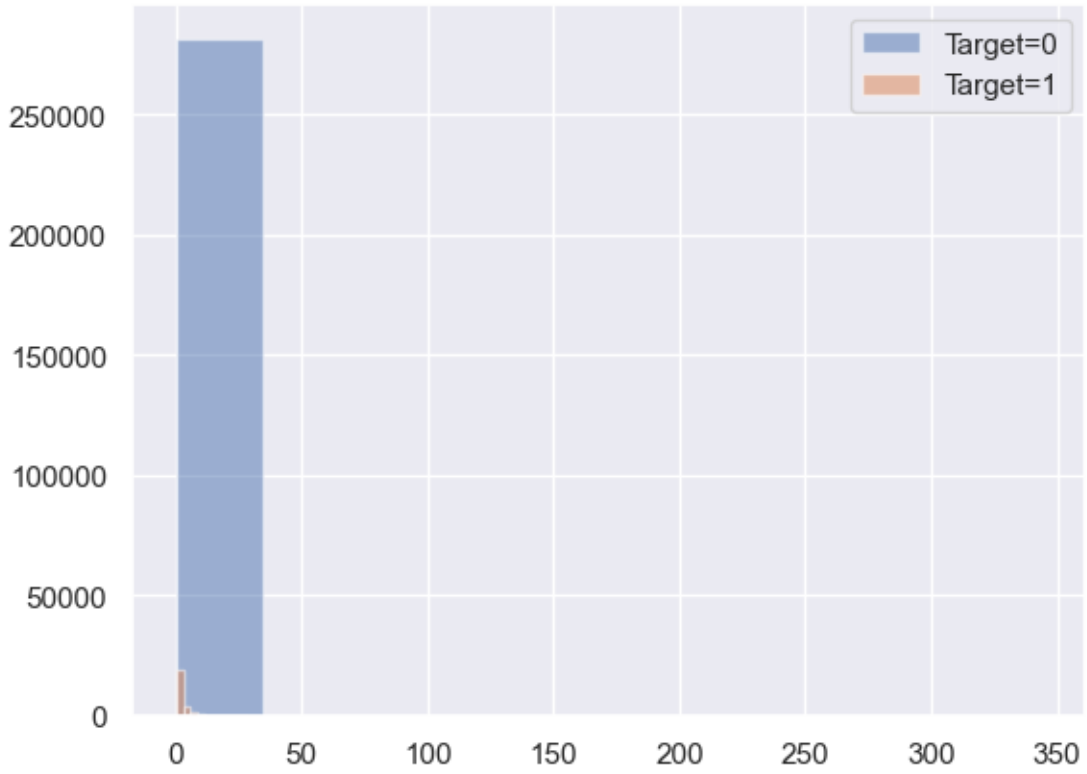


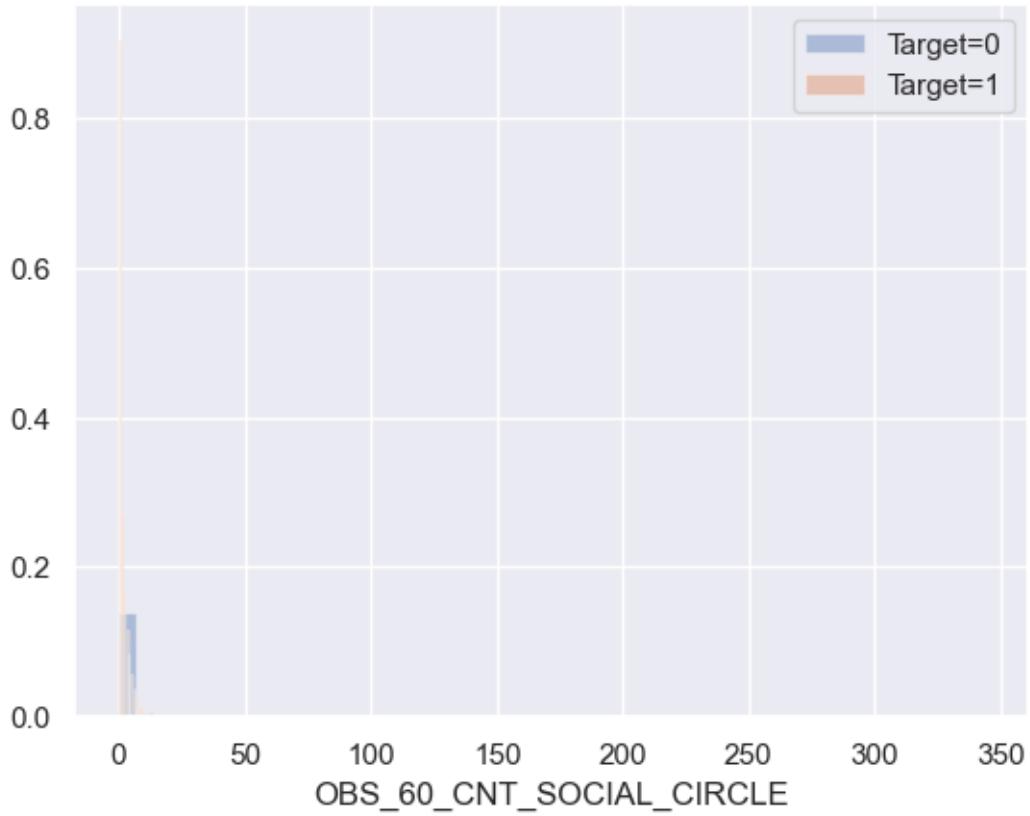

```
----------------------------------------------------------------------
Plot of DEF_30_CNT_SOCIAL_CIRCLE
```


Target=0
Target=1
250000
200000
150000
100000
50000
0
0
5
10
15
20
25
30
35

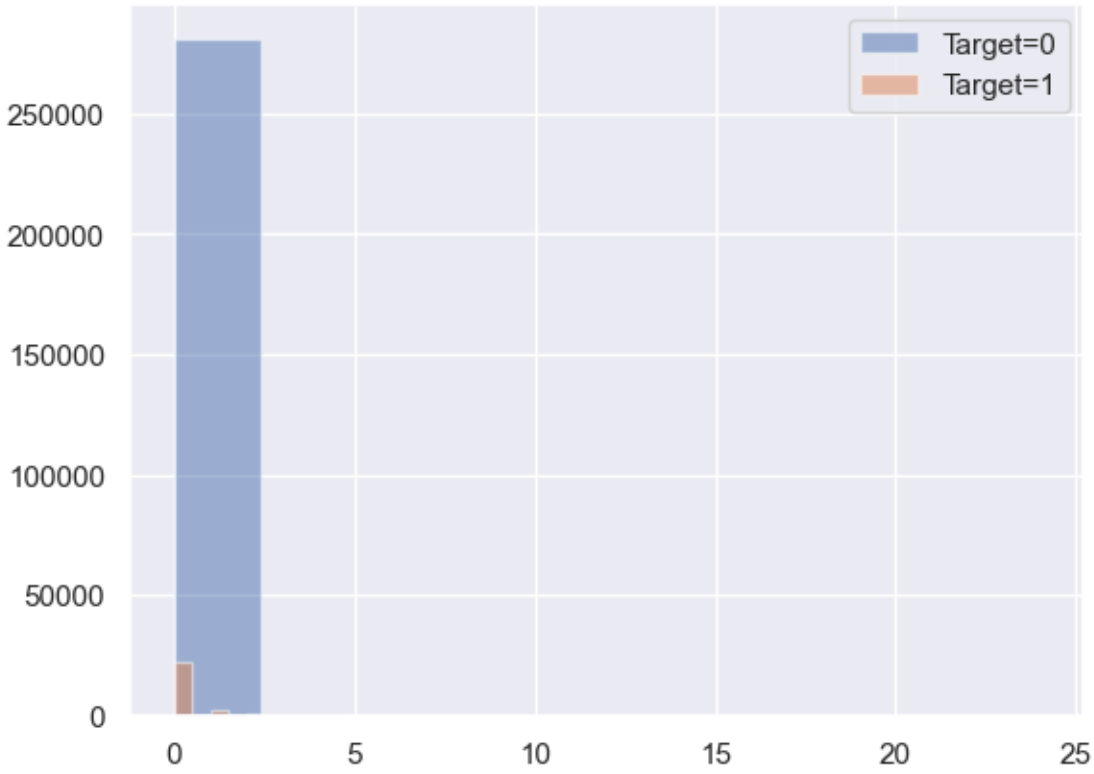


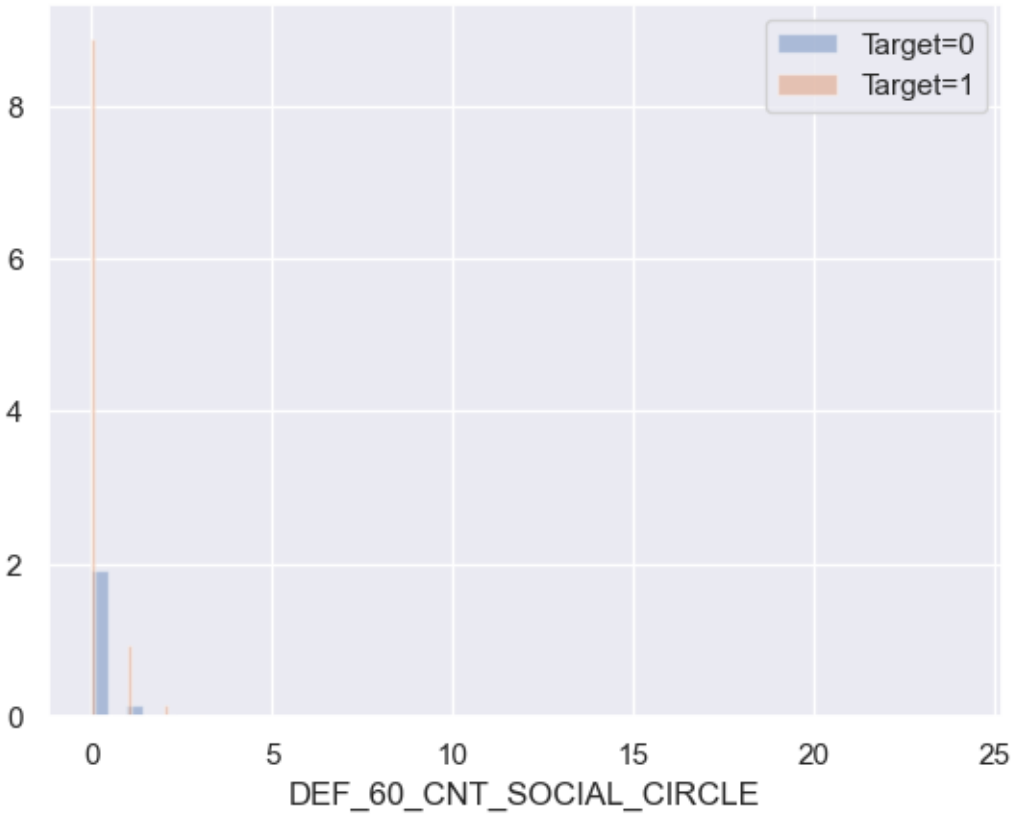
Plot of OBS_60_CNT_SOCIAL_CIRCLE


Target=0
Target=1
250000
200000
150000
100000
50000
0
0
50
100
150
200
250
300
350

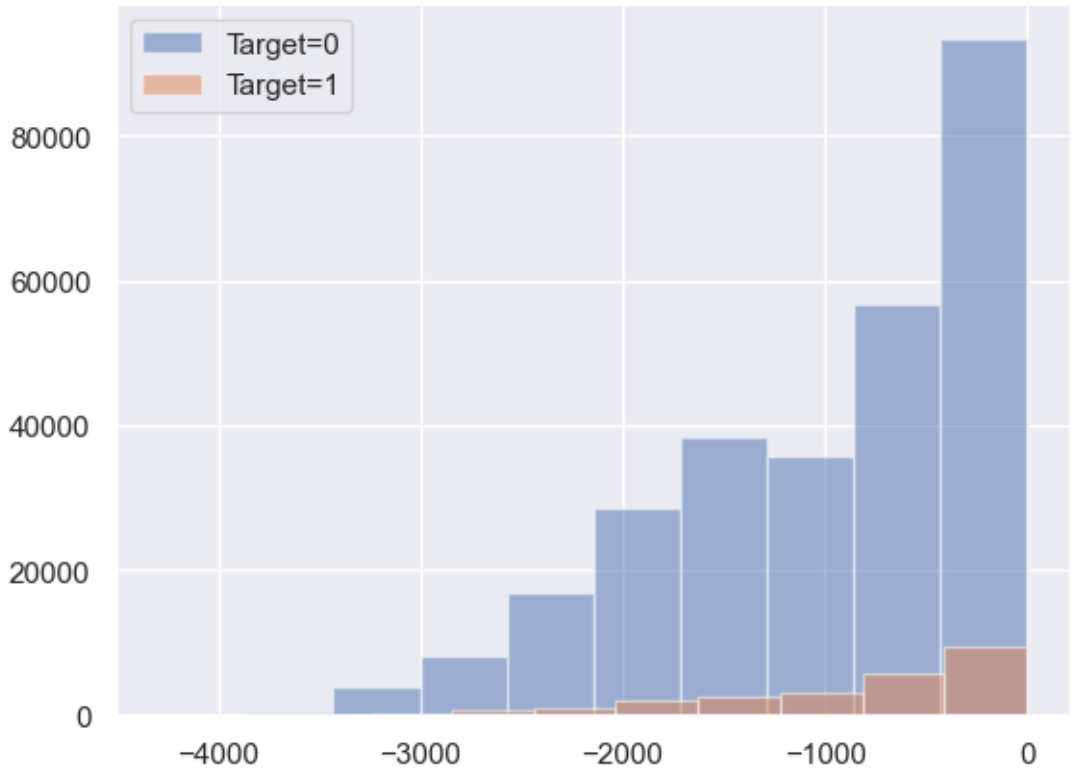


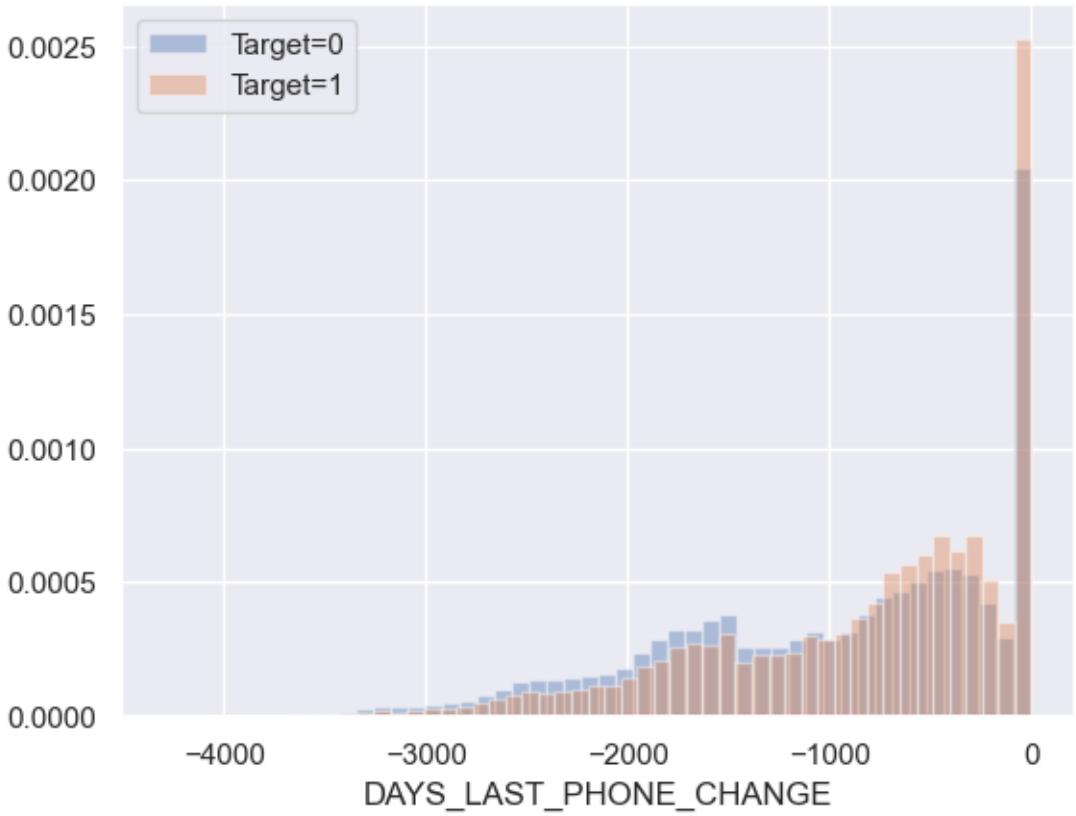
Plot of DEF_60_CNT_SOCIAL_CIRCLE


Target=0
Target=1
250000
200000
150000
100000
50000
0
0
5
10
15
20
25

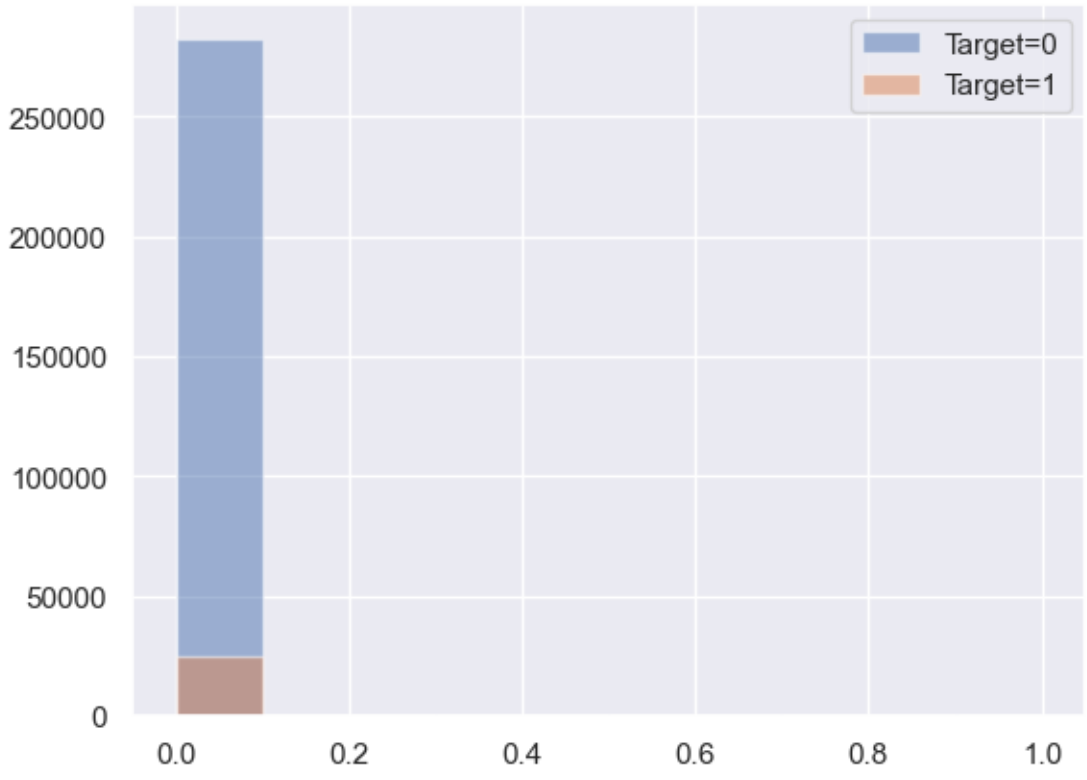


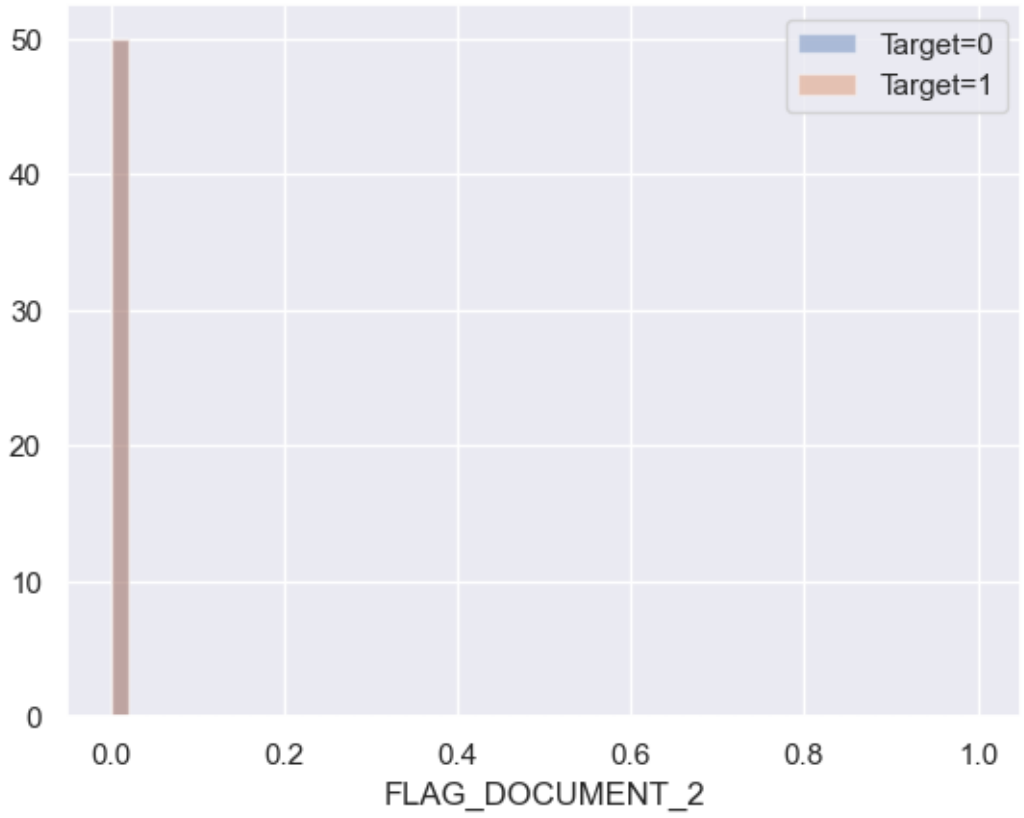
Plot of DAYS_LAST_PHONE_CHANGE


Target=0
Target=1
80000
60000
40000
20000
0
−4000
−3000
−2000
−1000
0

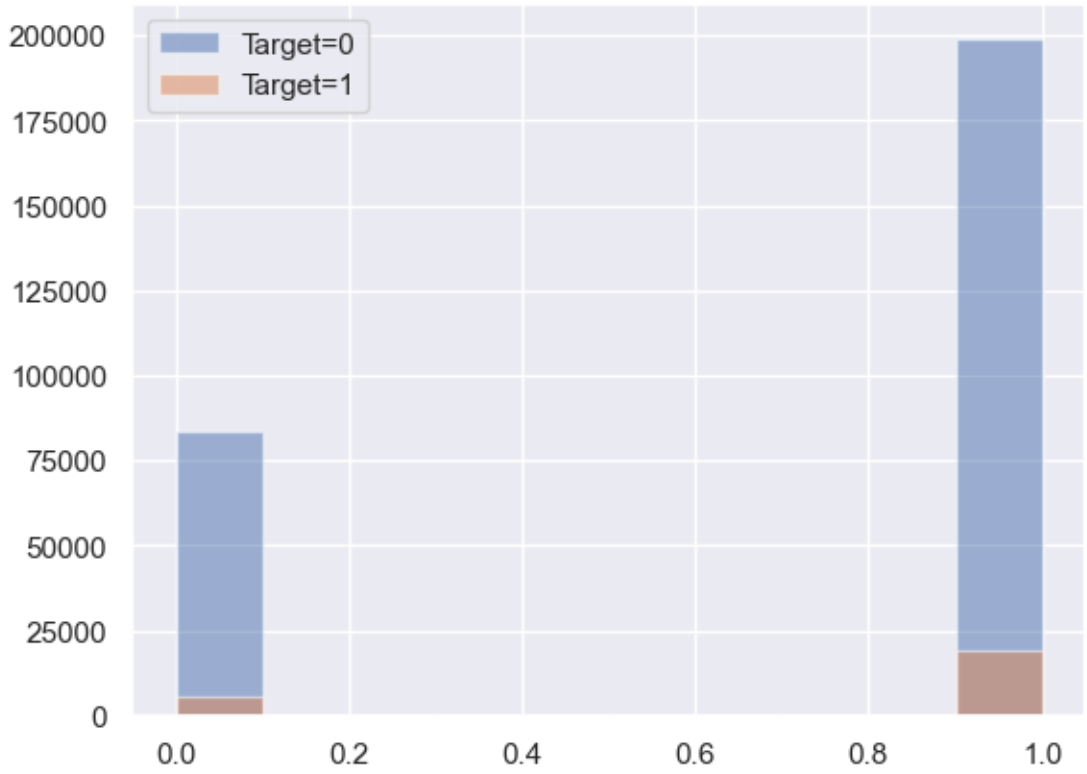


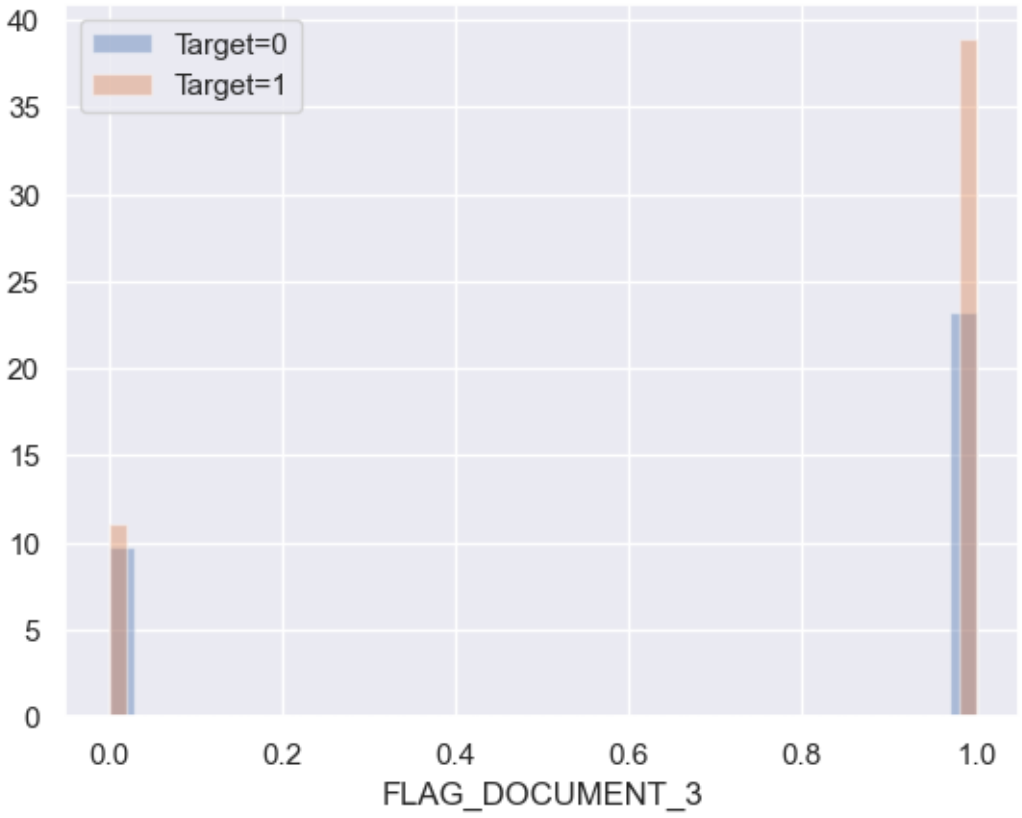
Plot of FLAG_DOCUMENT_2


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

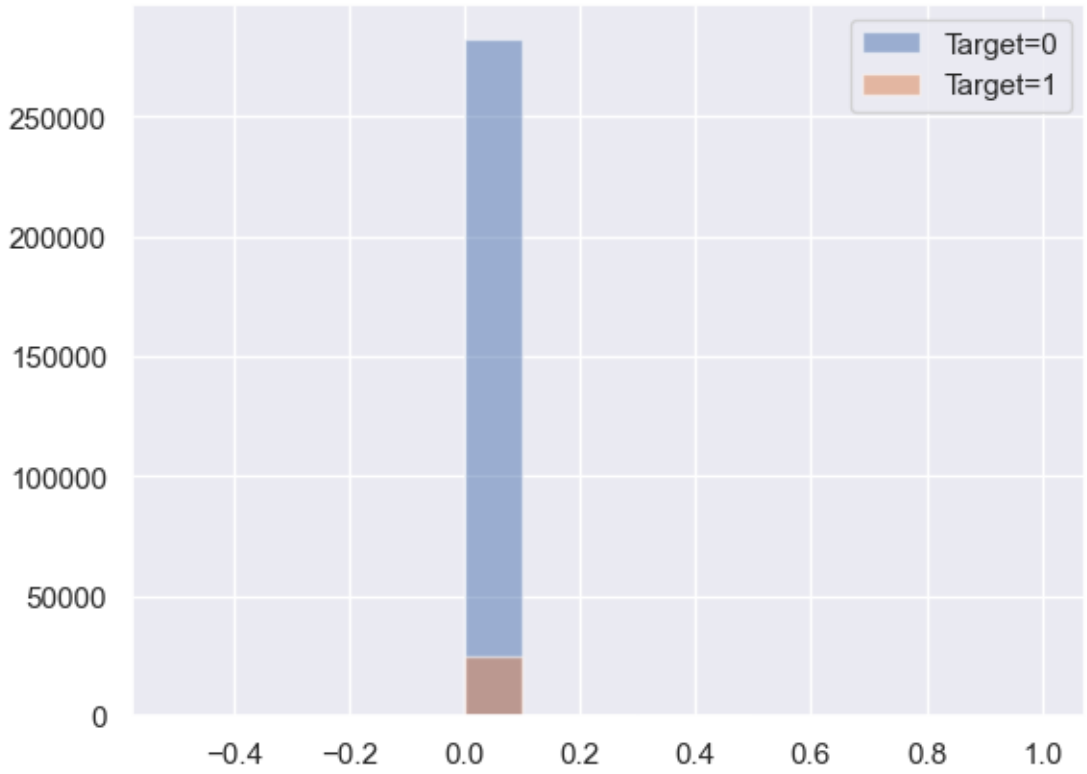


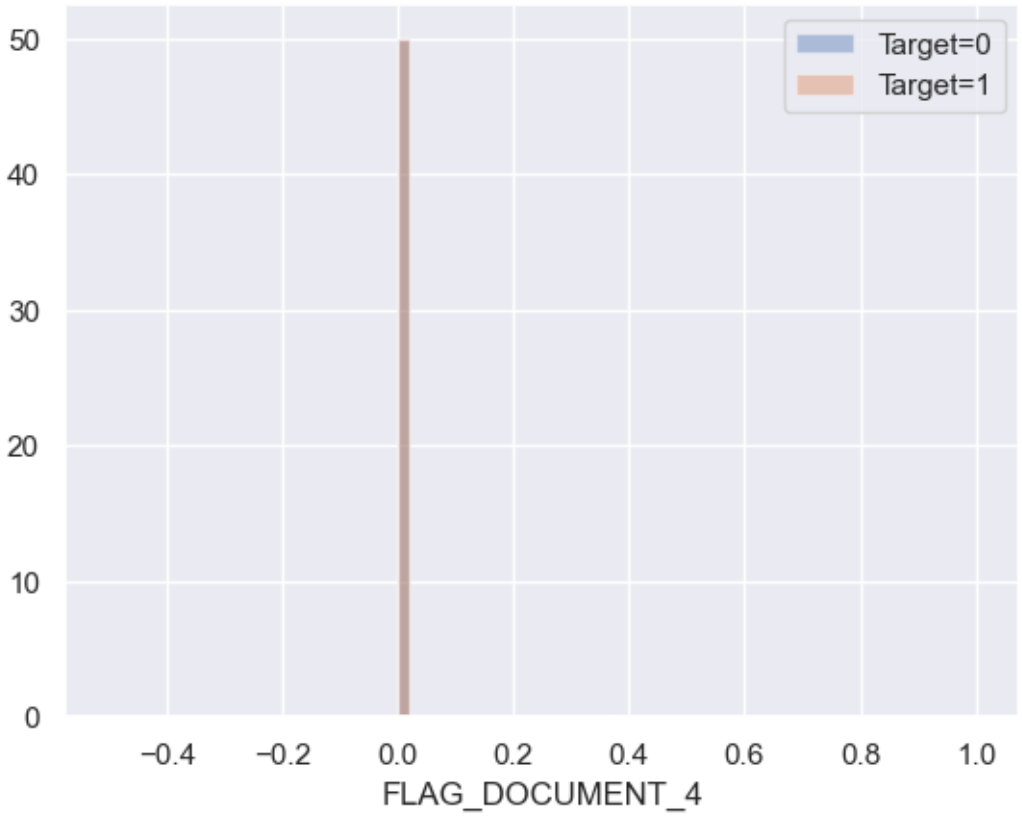
Plot of FLAG_DOCUMENT_3


Target=0
Target=1
200000
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.2
0.4
0.6
0.8
1.0

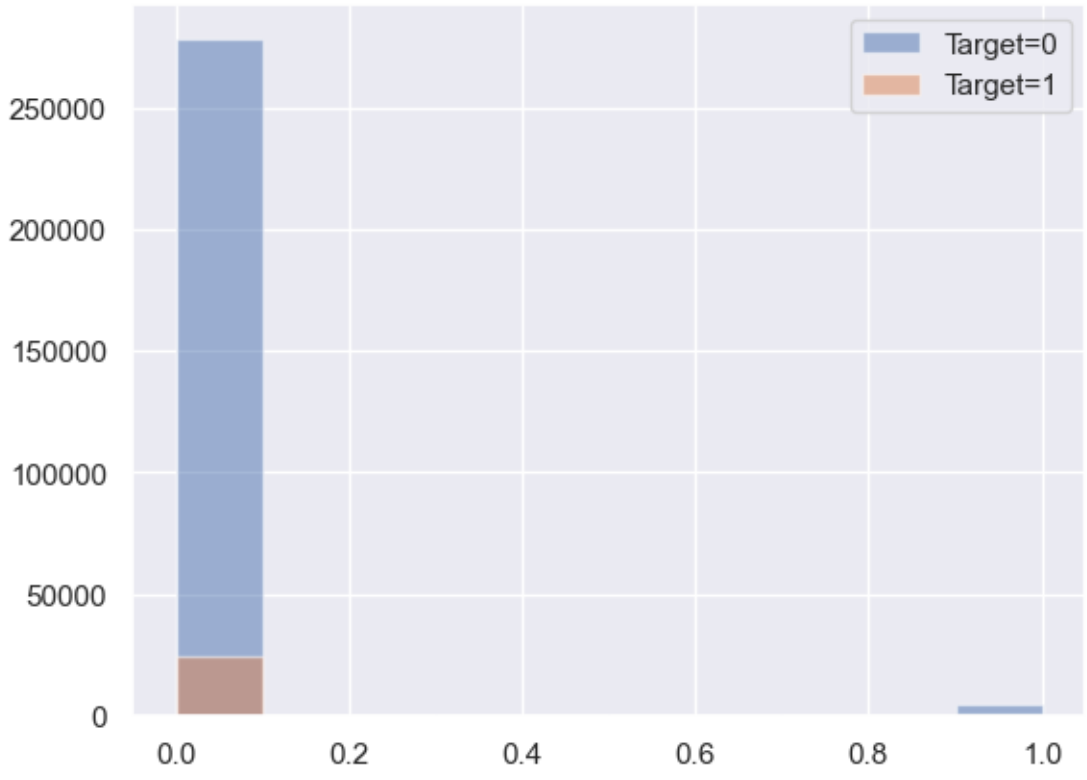


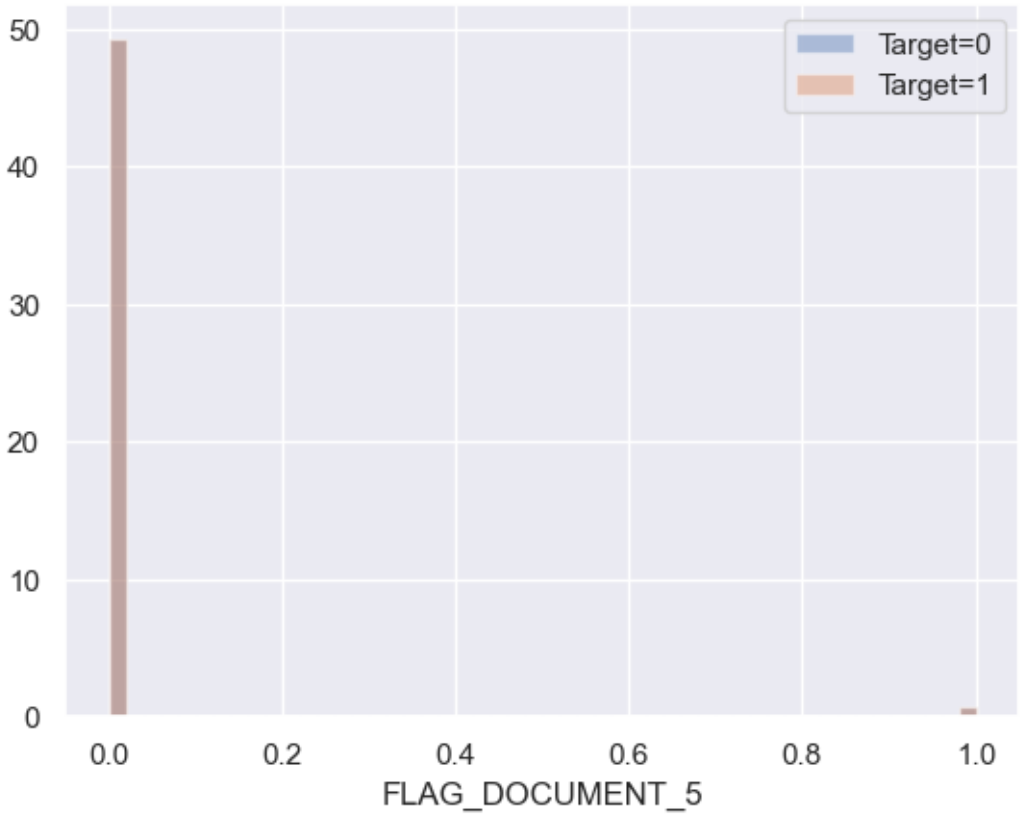
Plot of FLAG_DOCUMENT_4


Target=0
Target=1
250000
200000
150000
100000
50000
0
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

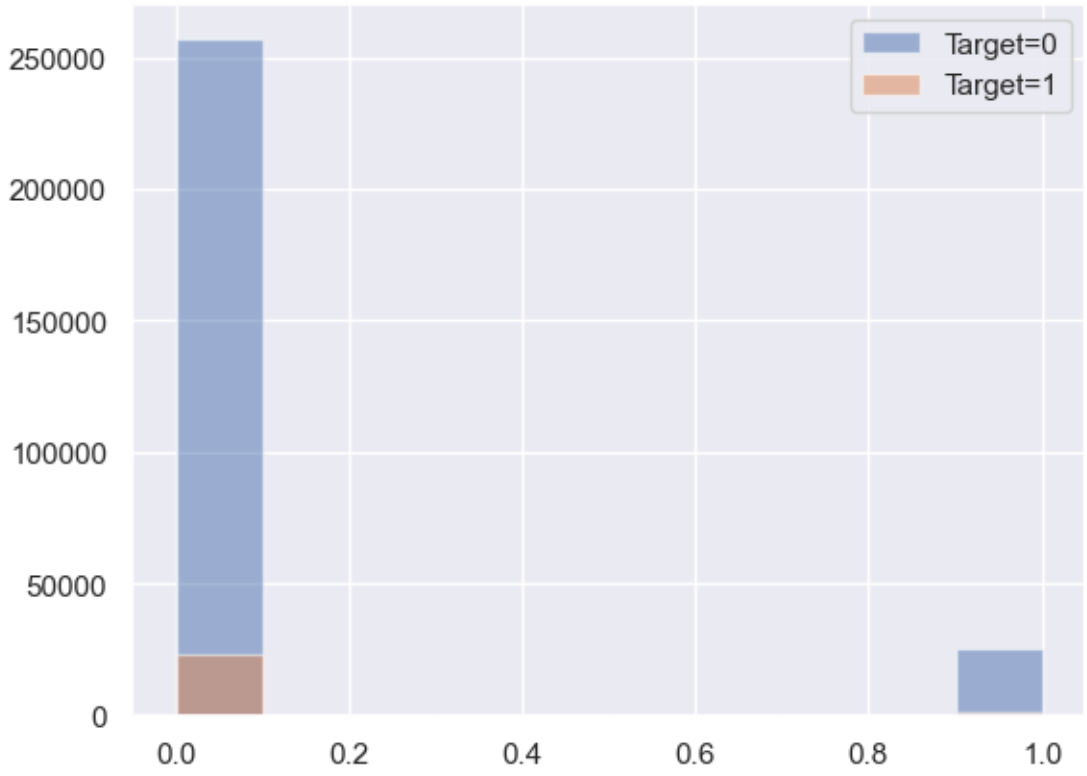


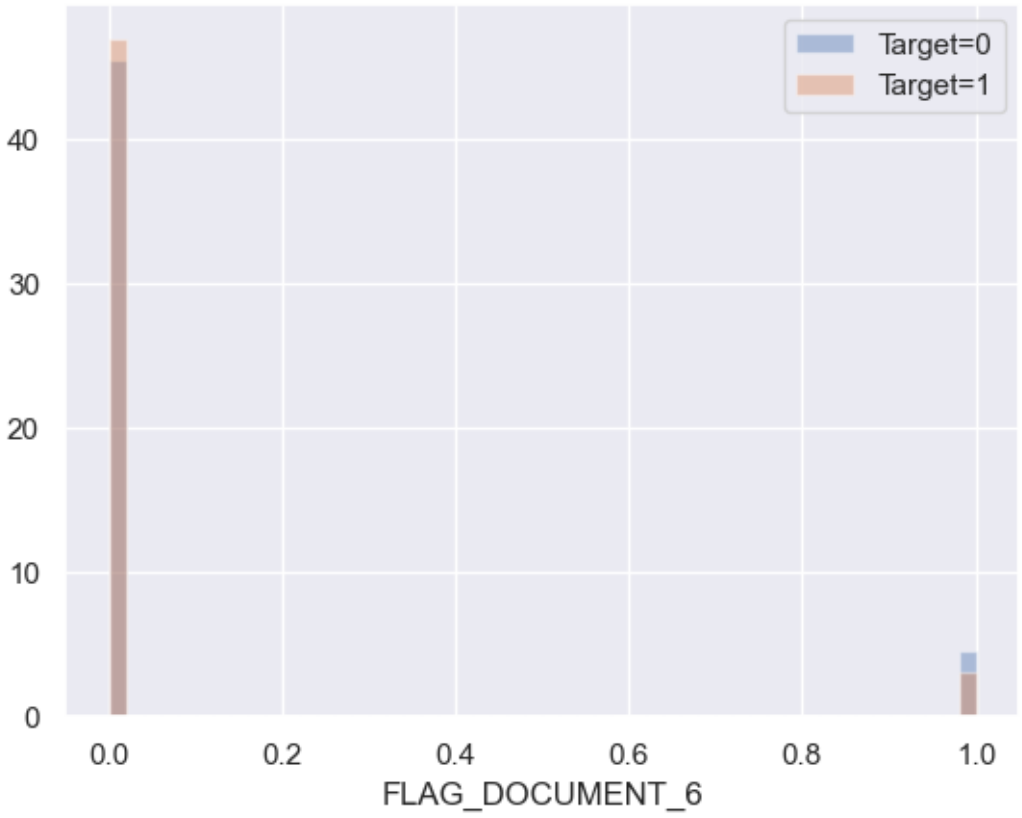

```
----------------------------------------------------------------------
Plot of FLAG_DOCUMENT_5
```


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

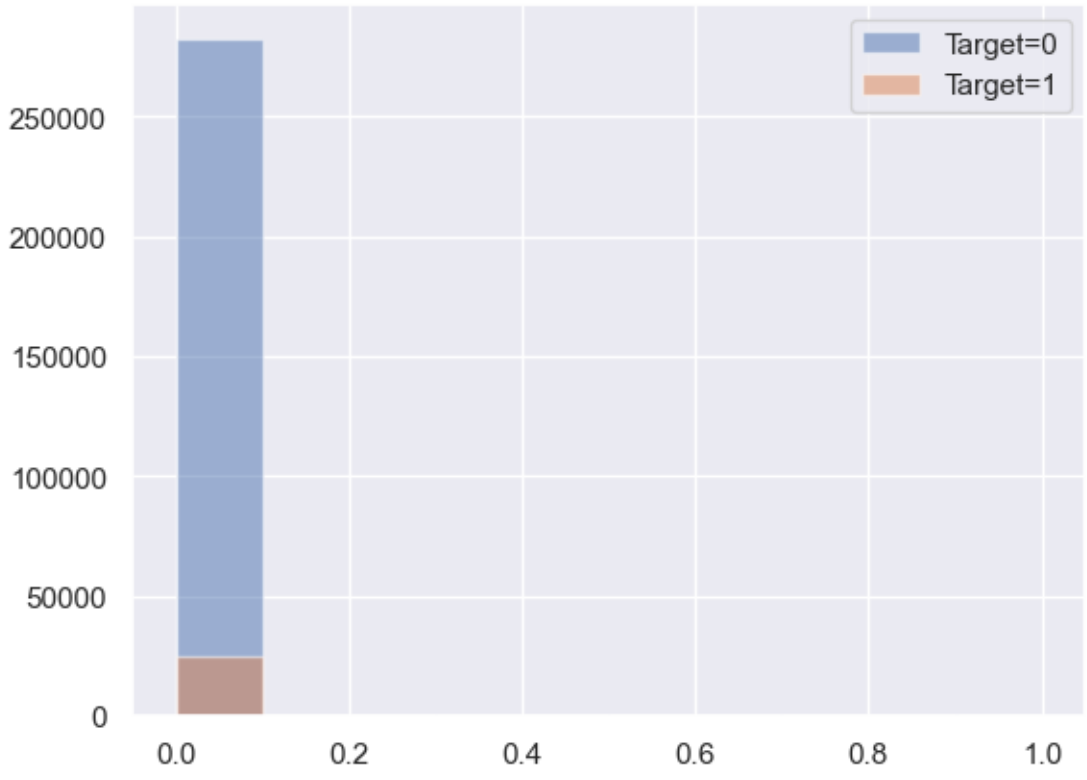


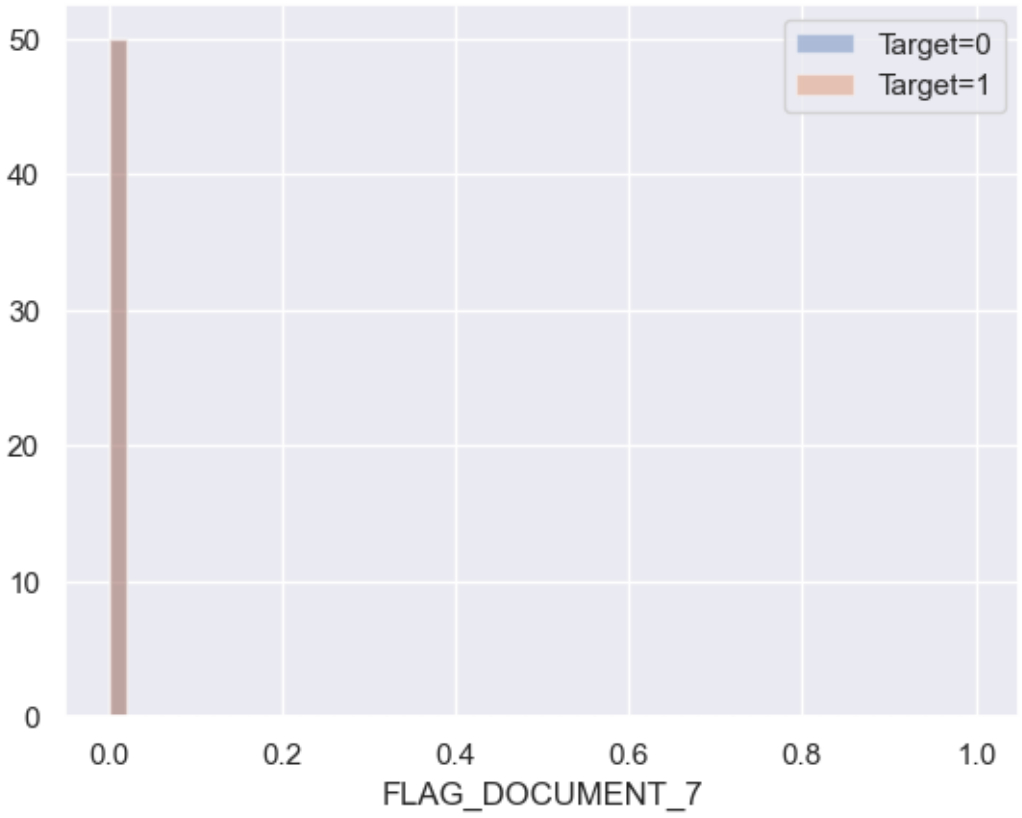
Plot of FLAG_DOCUMENT_6


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

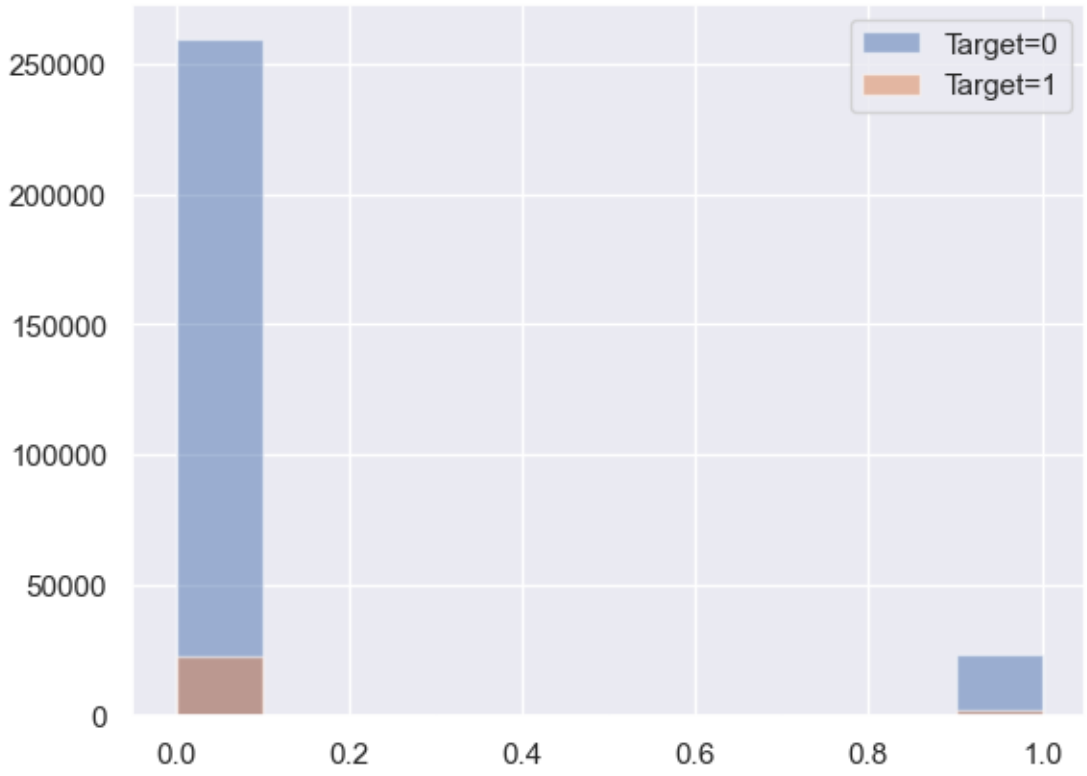


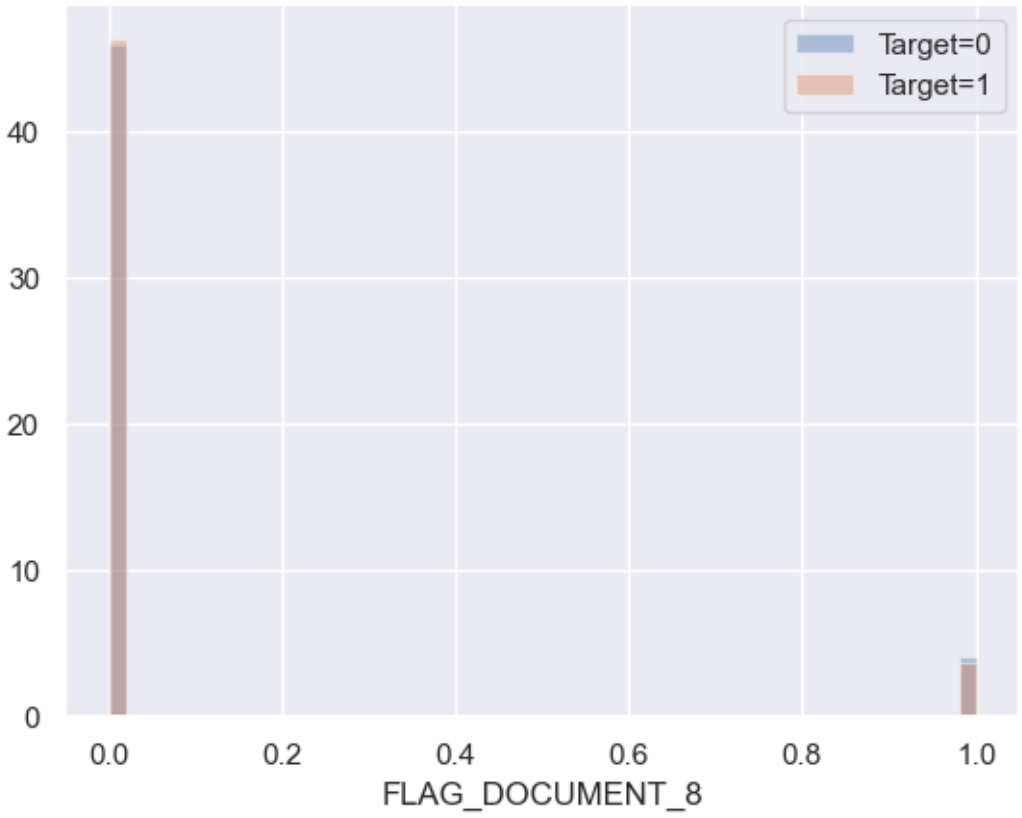
Plot of FLAG_DOCUMENT_7


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

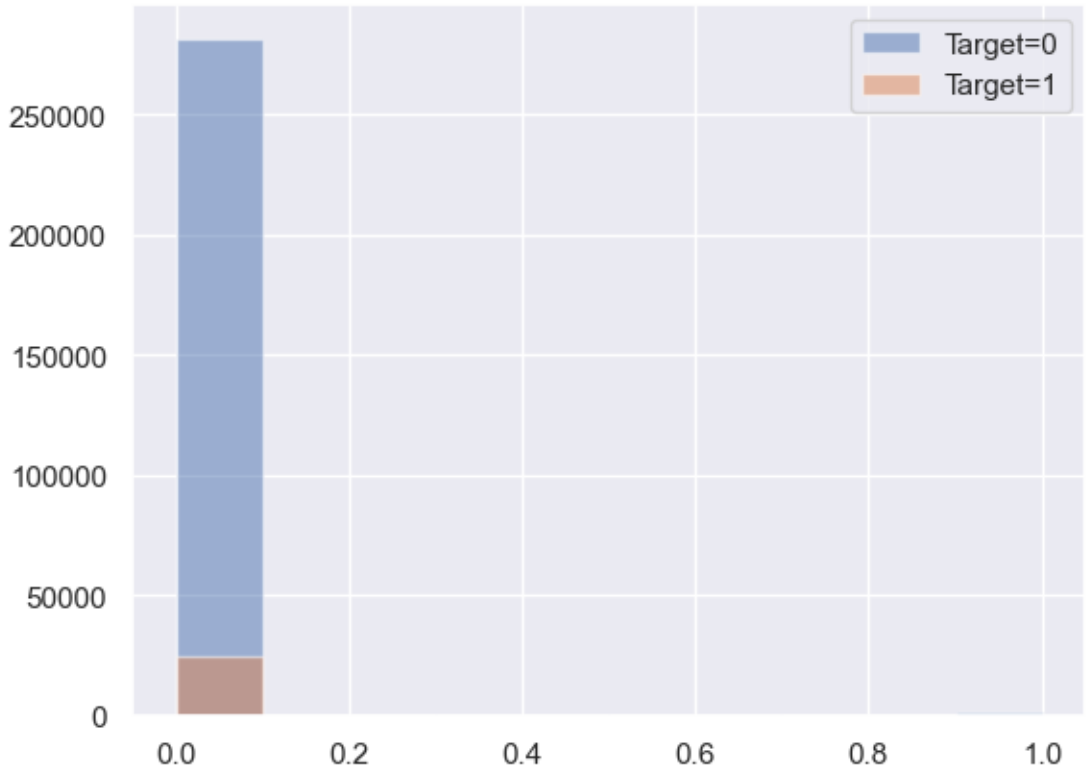


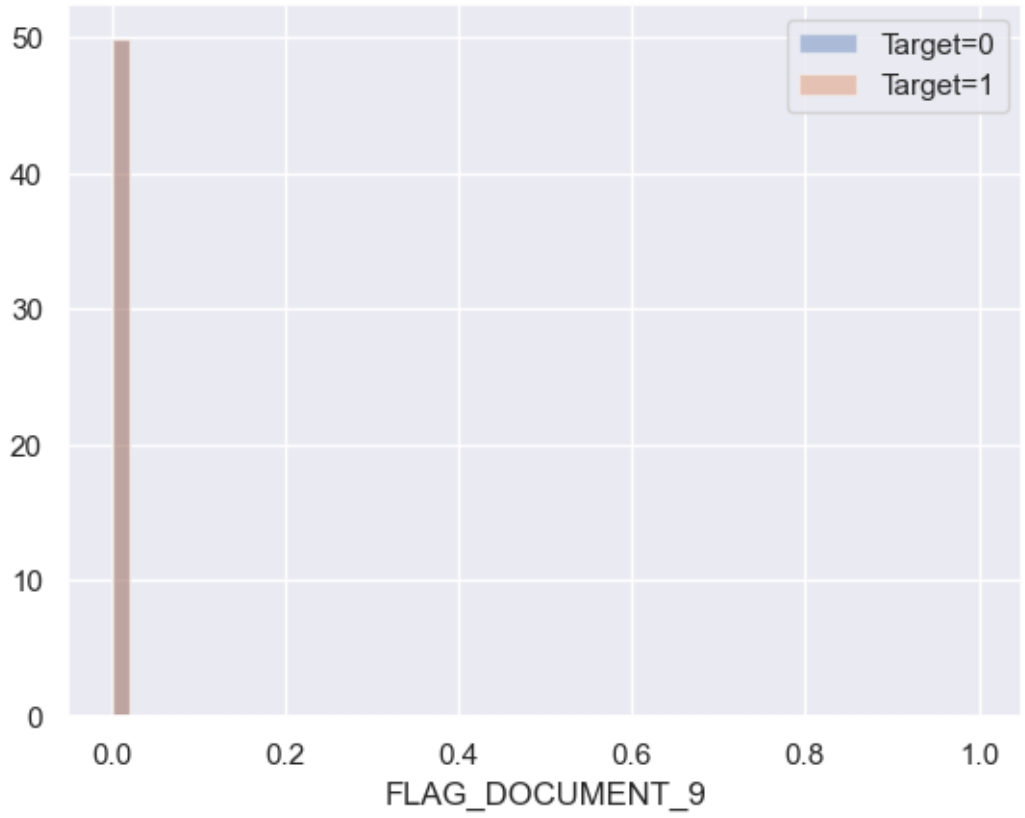
Plot of FLAG_DOCUMENT_8


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

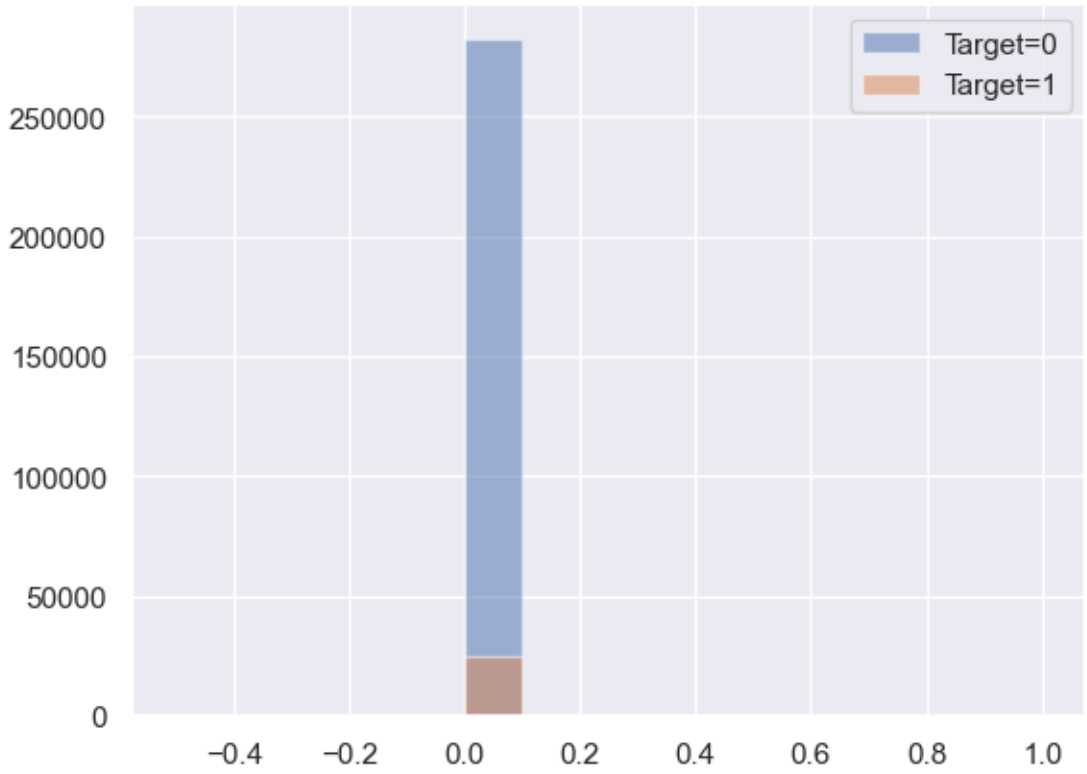


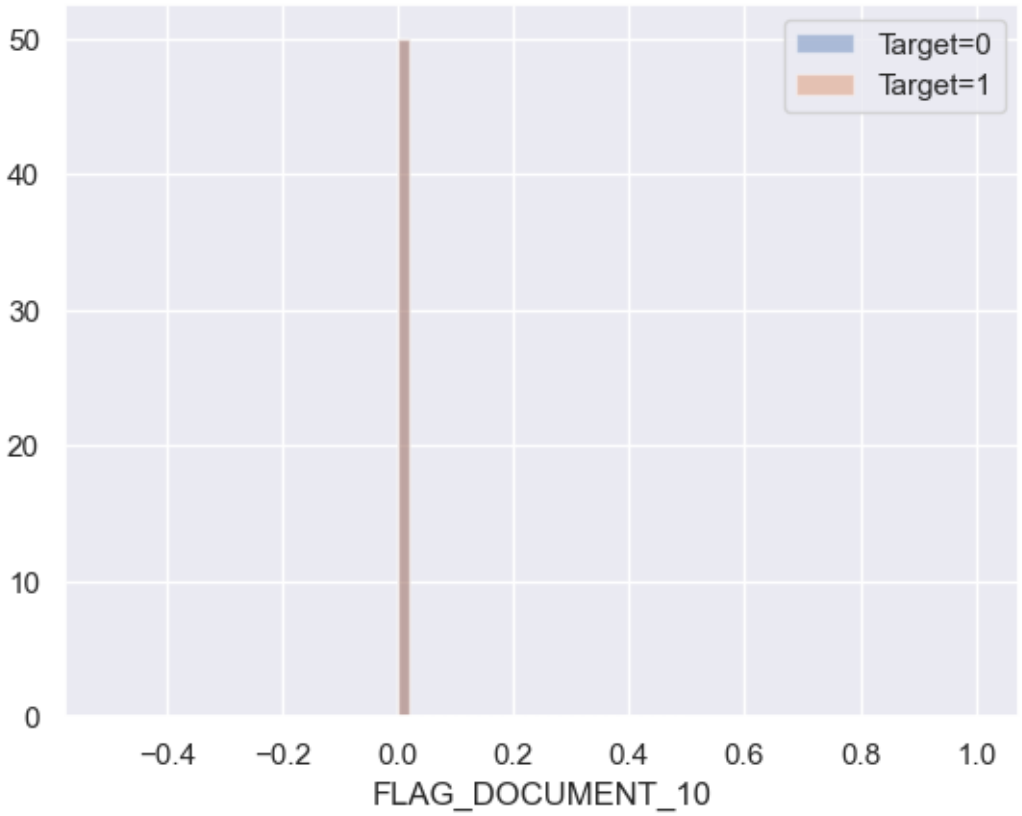
Plot of FLAG_DOCUMENT_9


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

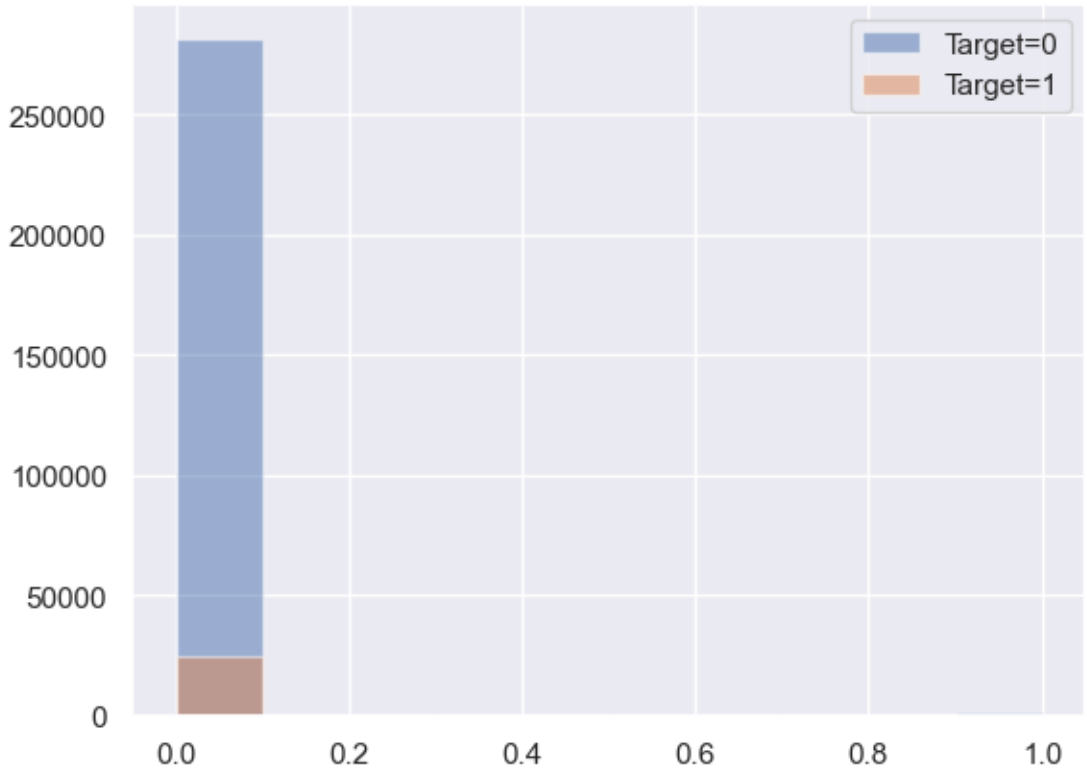


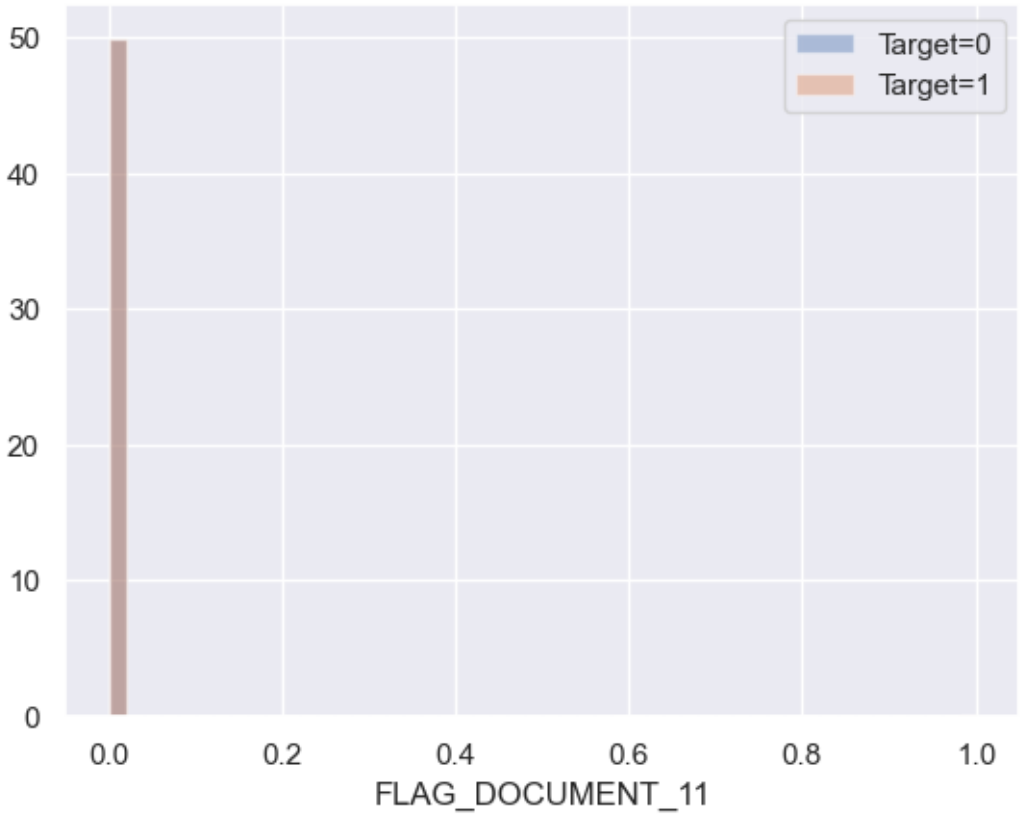
Plot of FLAG_DOCUMENT_10


Target=0
Target=1
250000
200000
150000
100000
50000
0
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

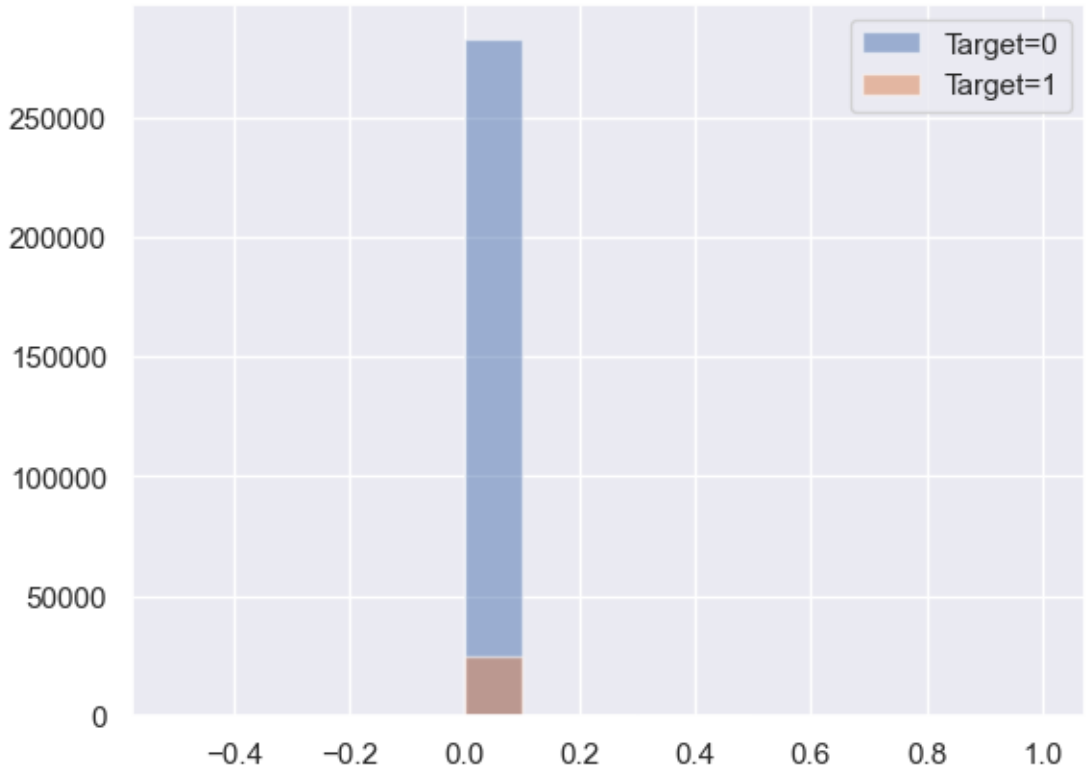


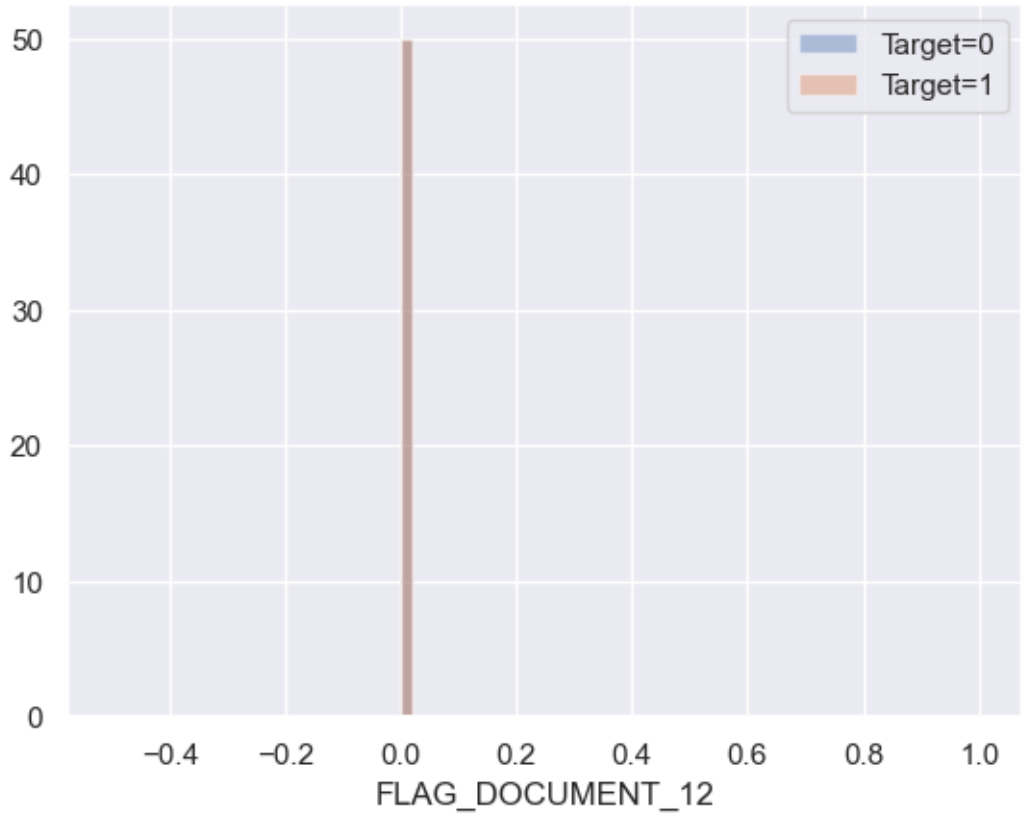
Plot of FLAG_DOCUMENT_11


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

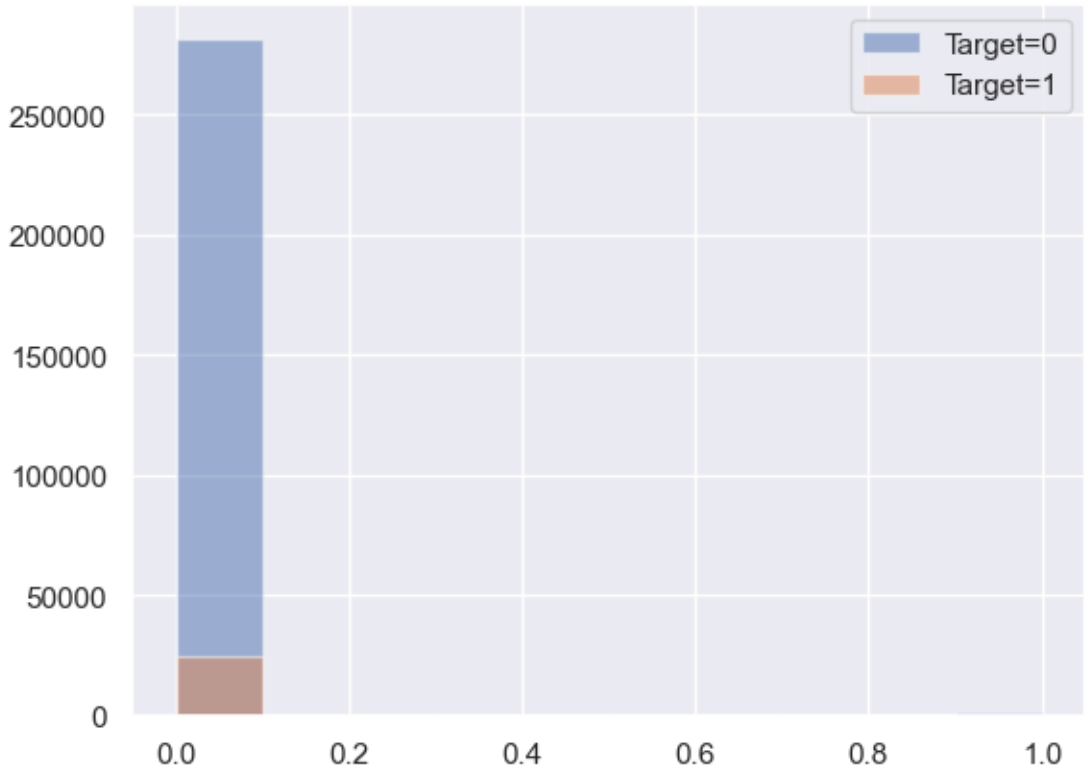


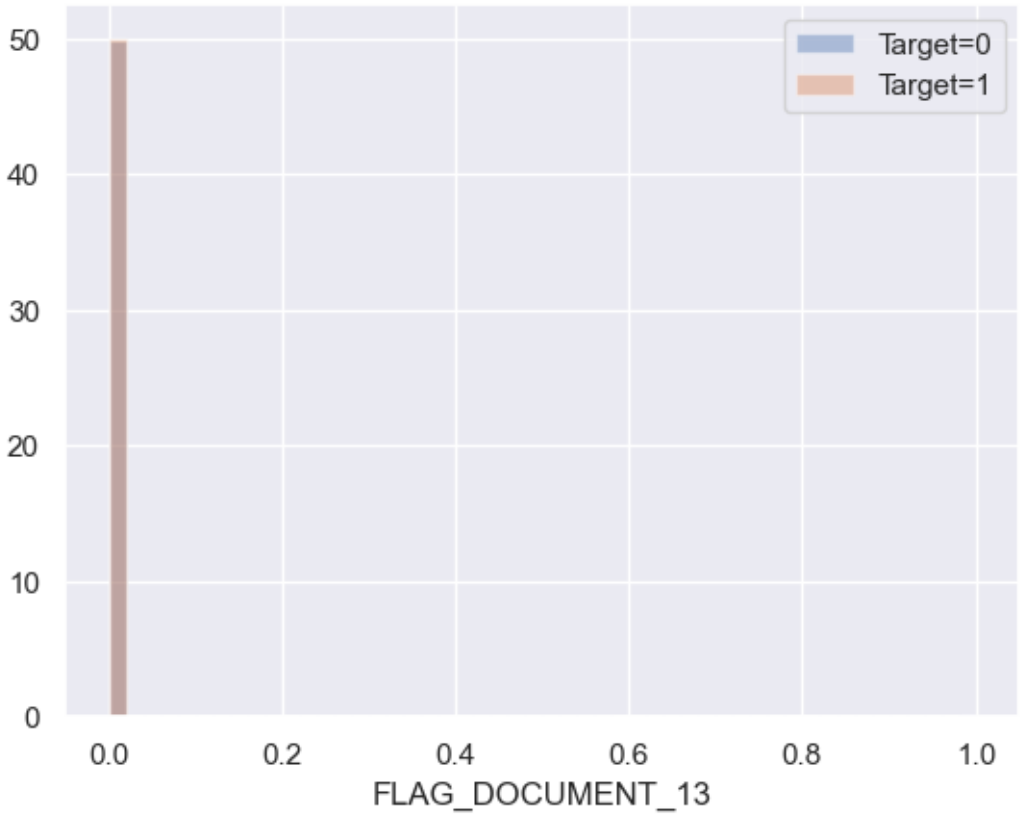
Plot of FLAG_DOCUMENT_12


Target=0
Target=1
250000
200000
150000
100000
50000
0
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

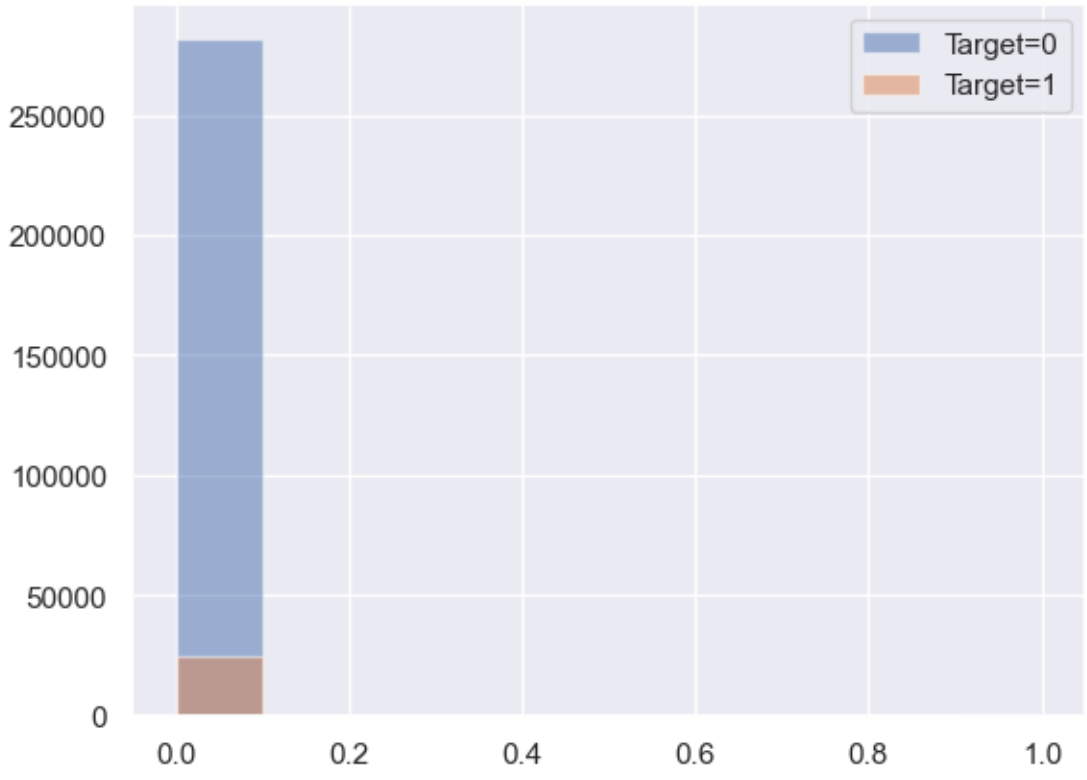


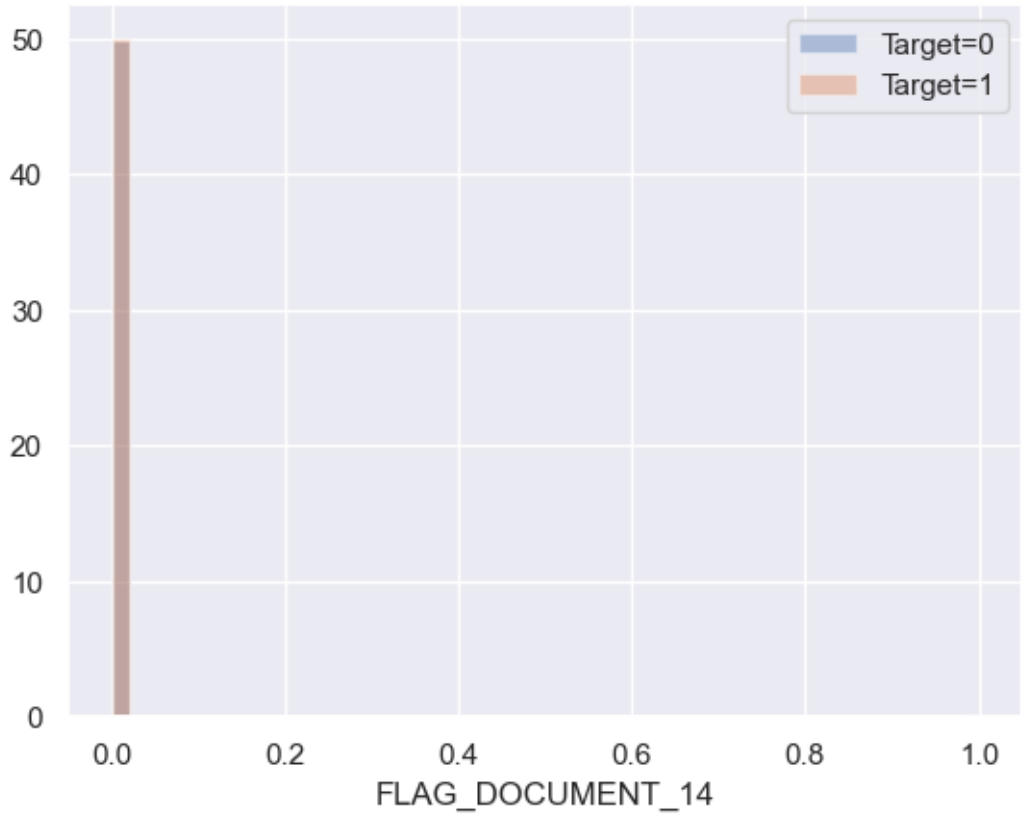
Plot of FLAG_DOCUMENT_13


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

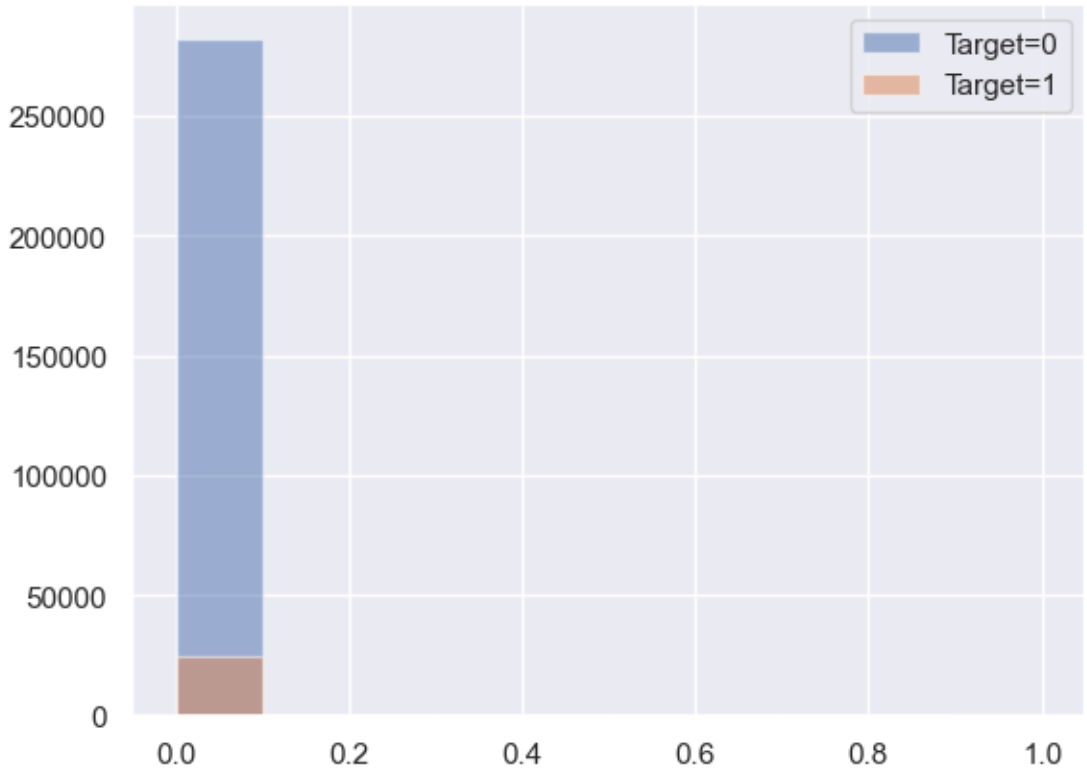


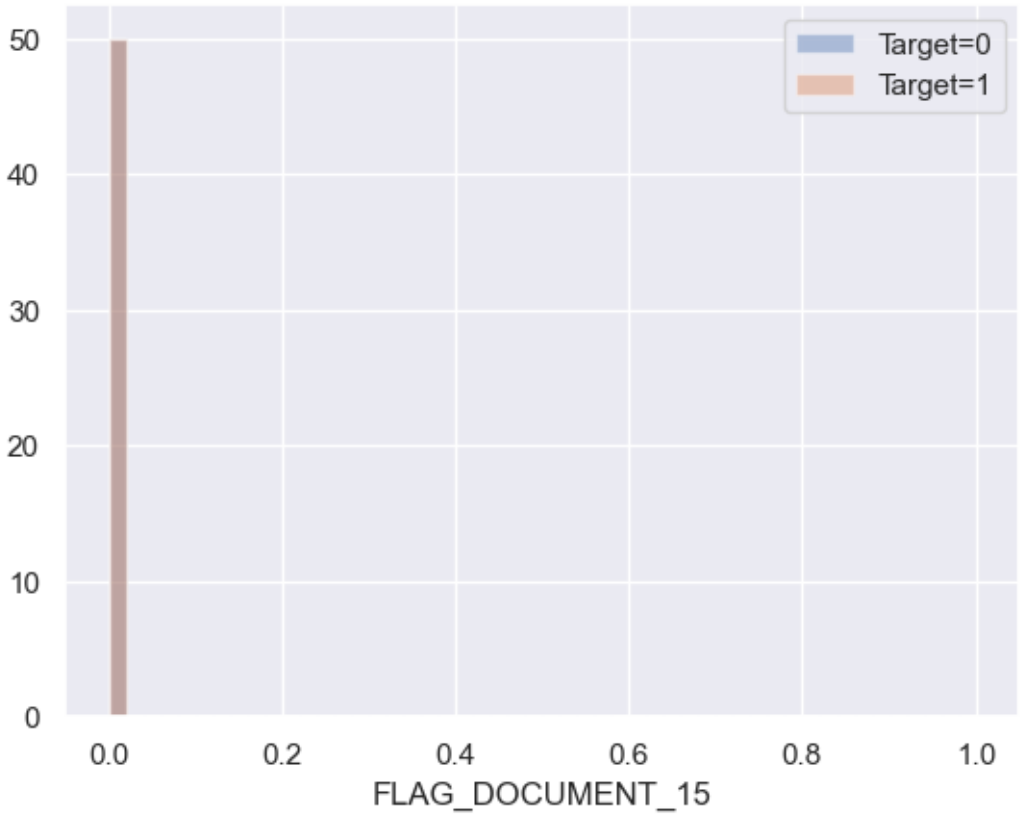
Plot of FLAG_DOCUMENT_14


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

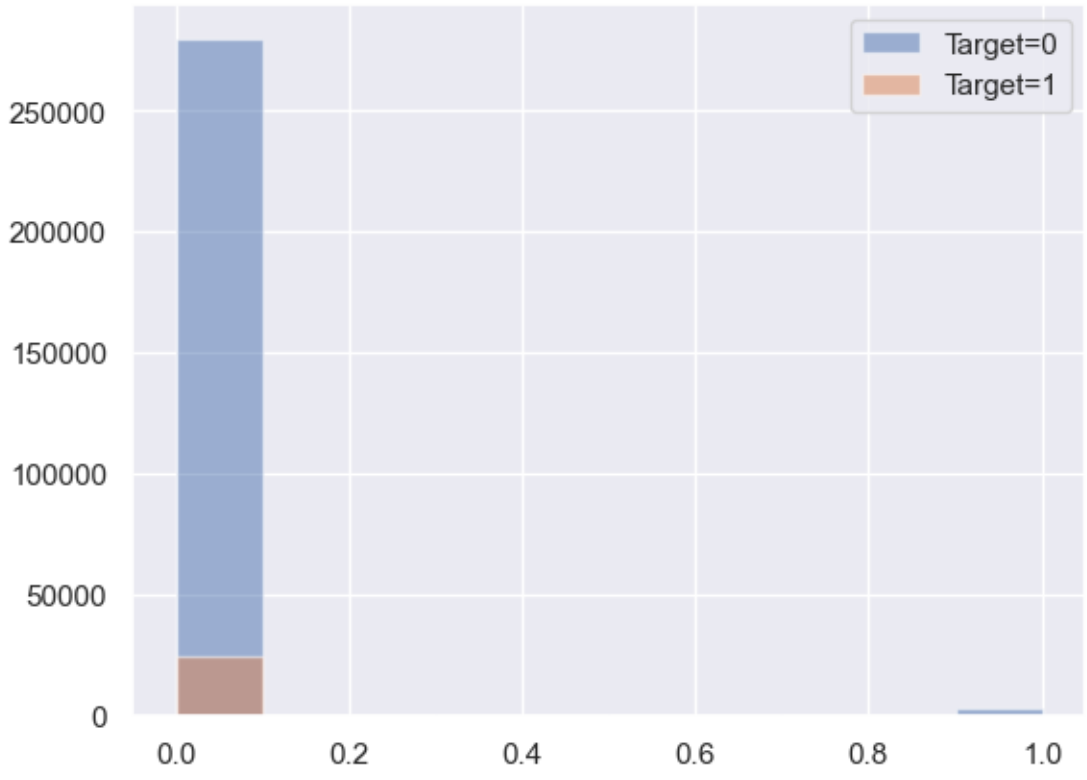


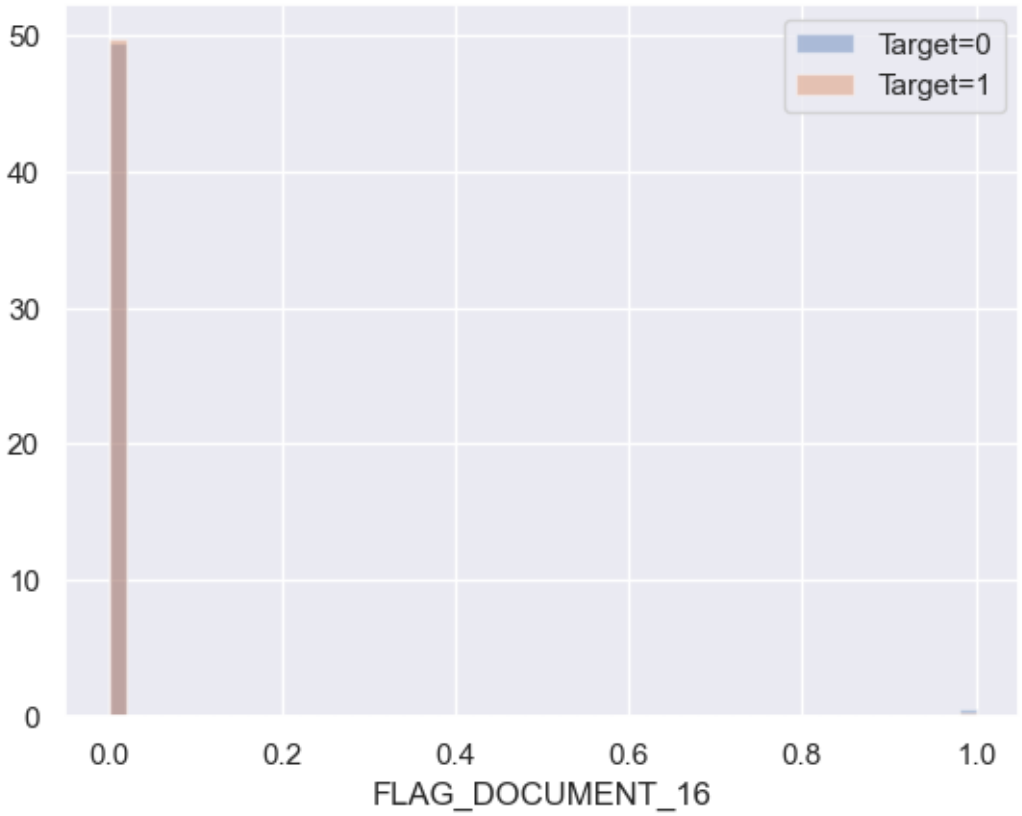
Plot of FLAG_DOCUMENT_15


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

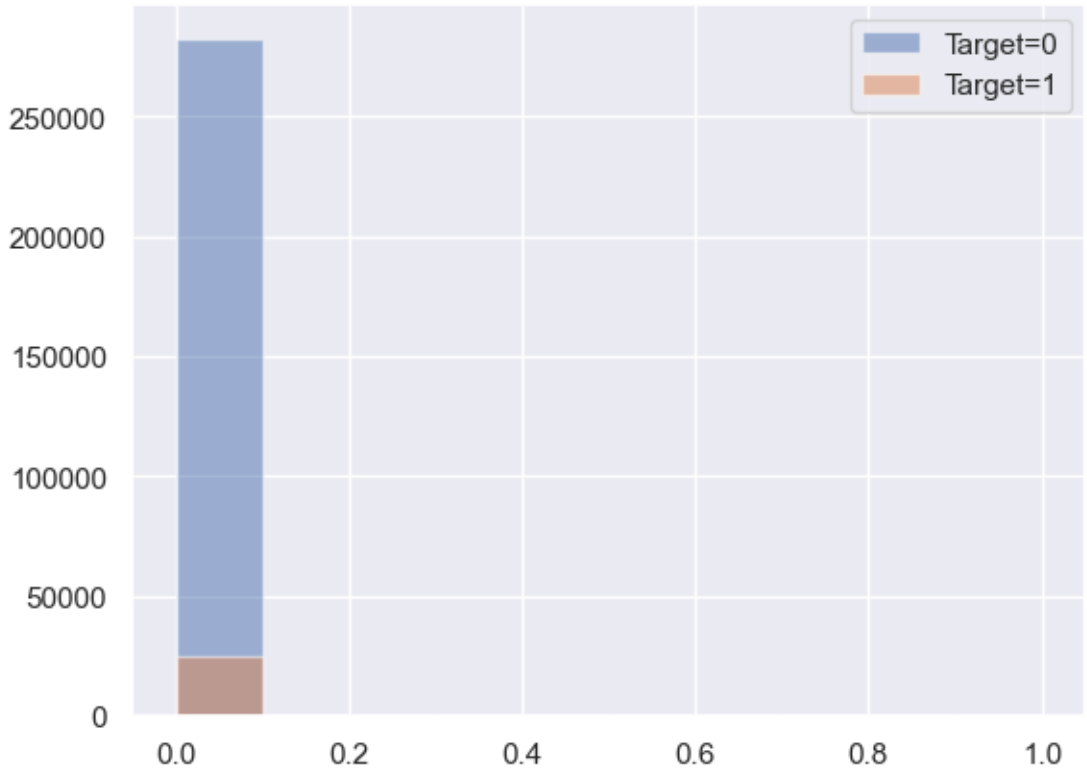


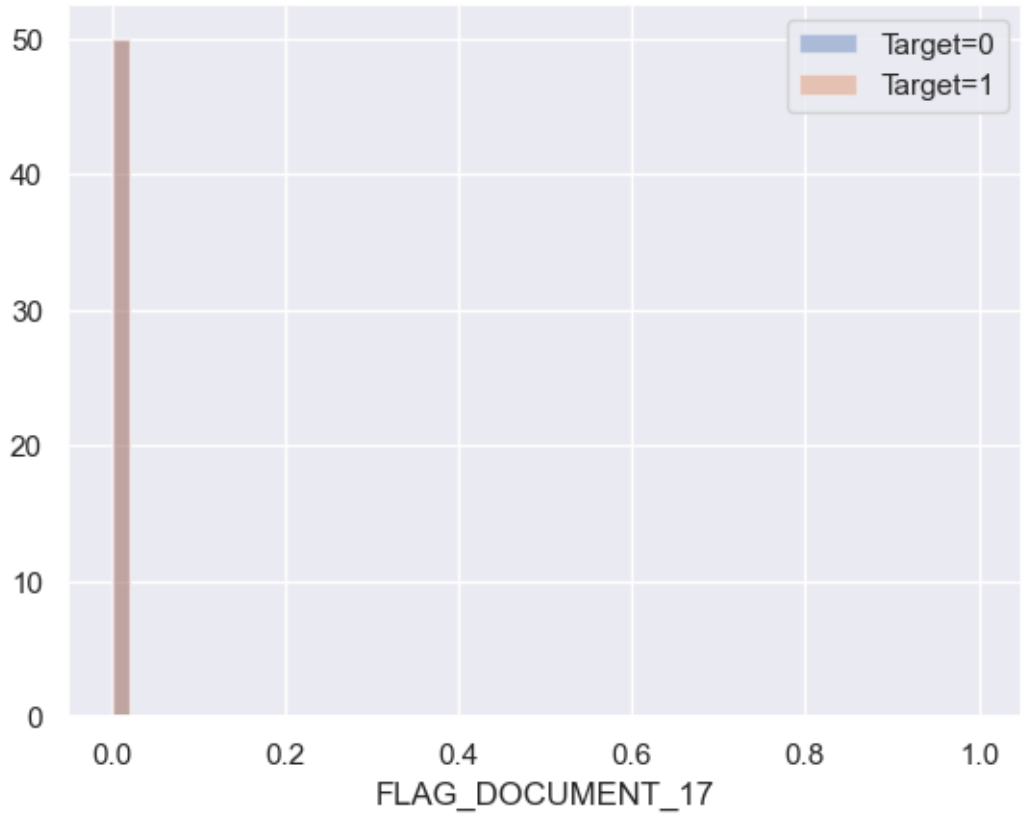
Plot of FLAG_DOCUMENT_16


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

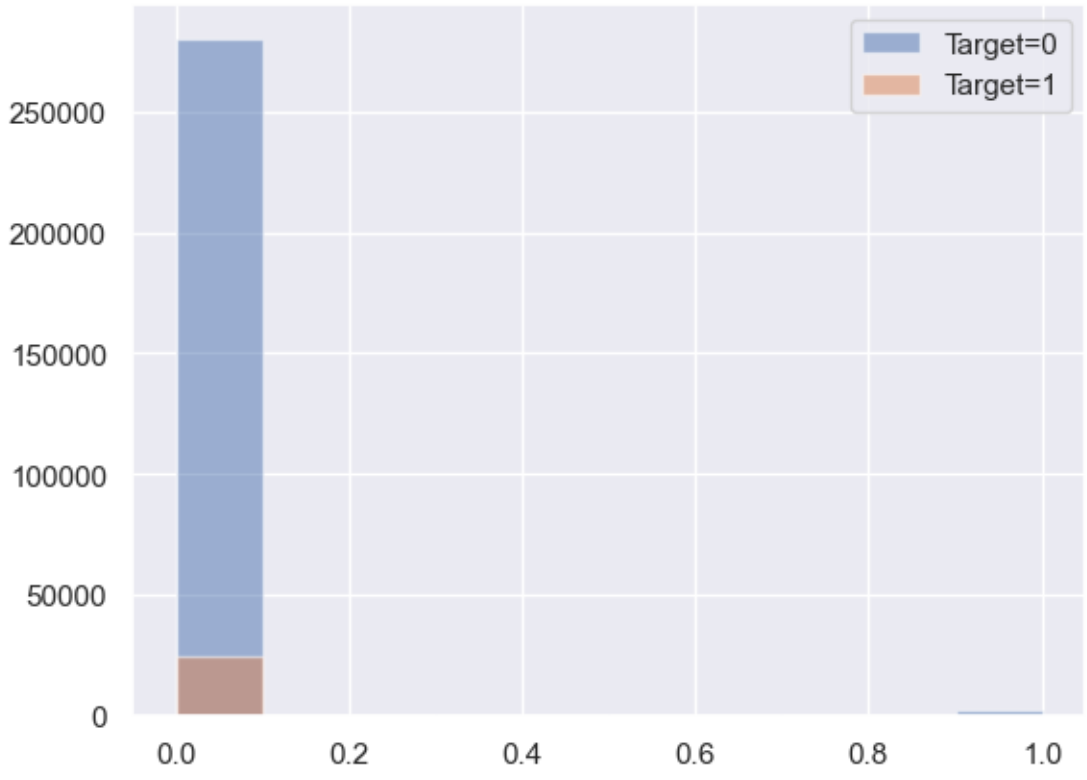


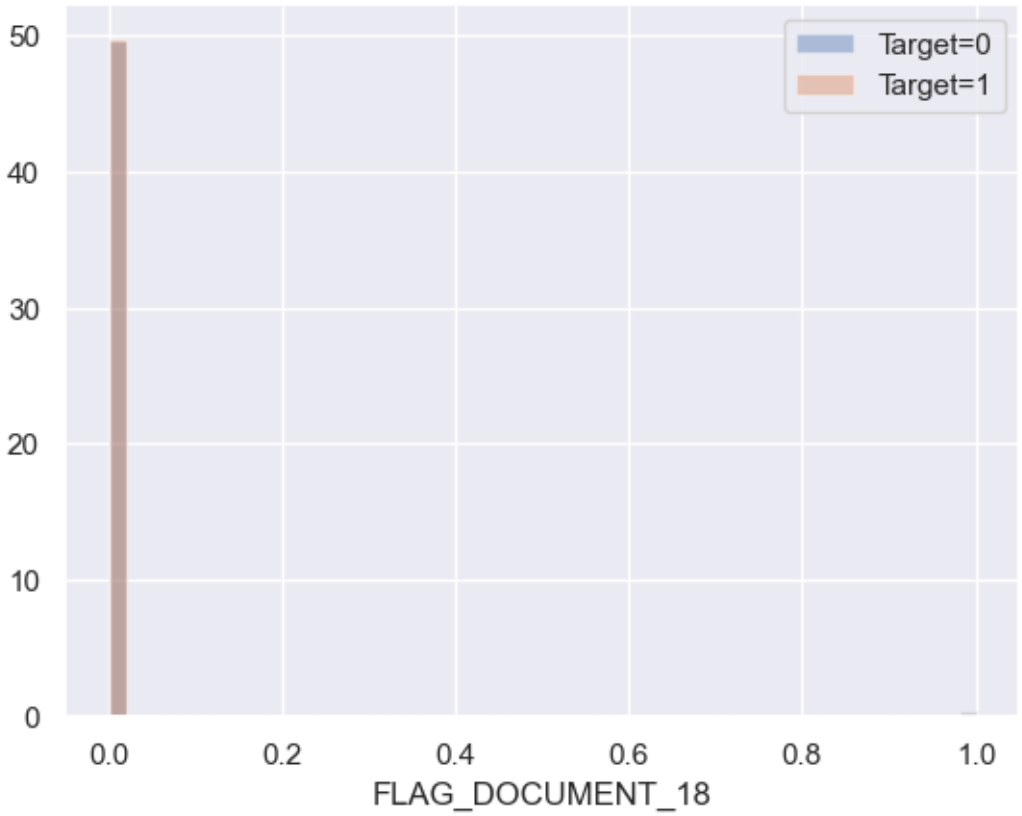
Plot of FLAG_DOCUMENT_17


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

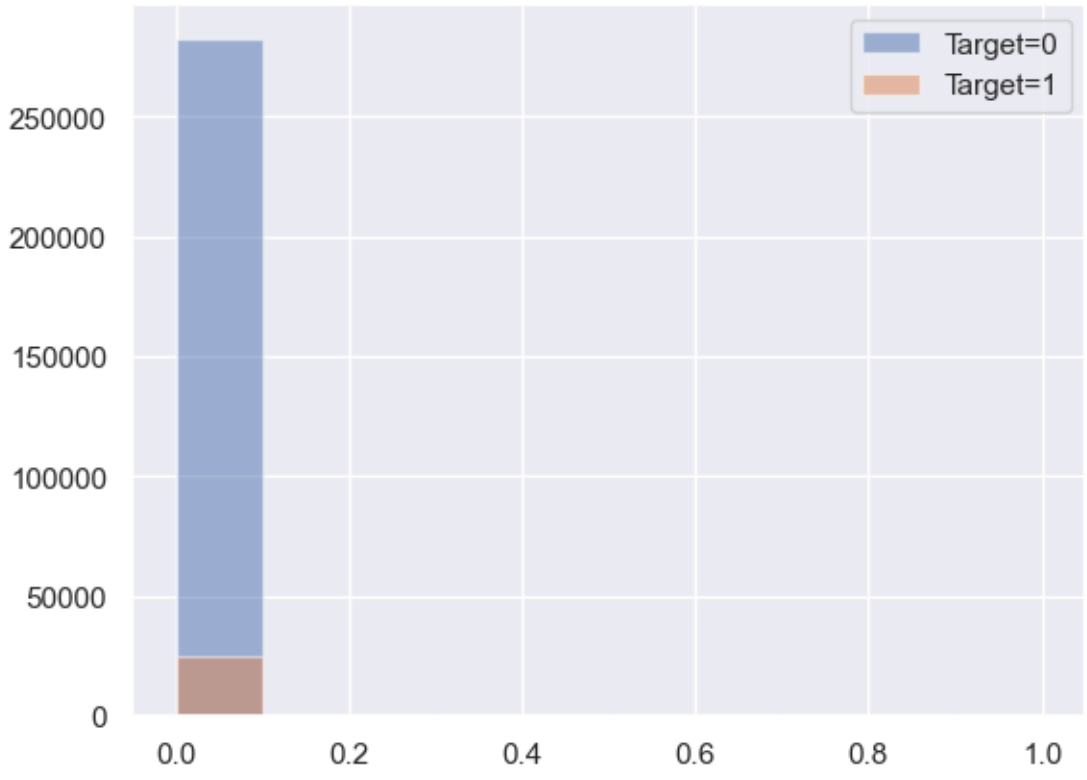


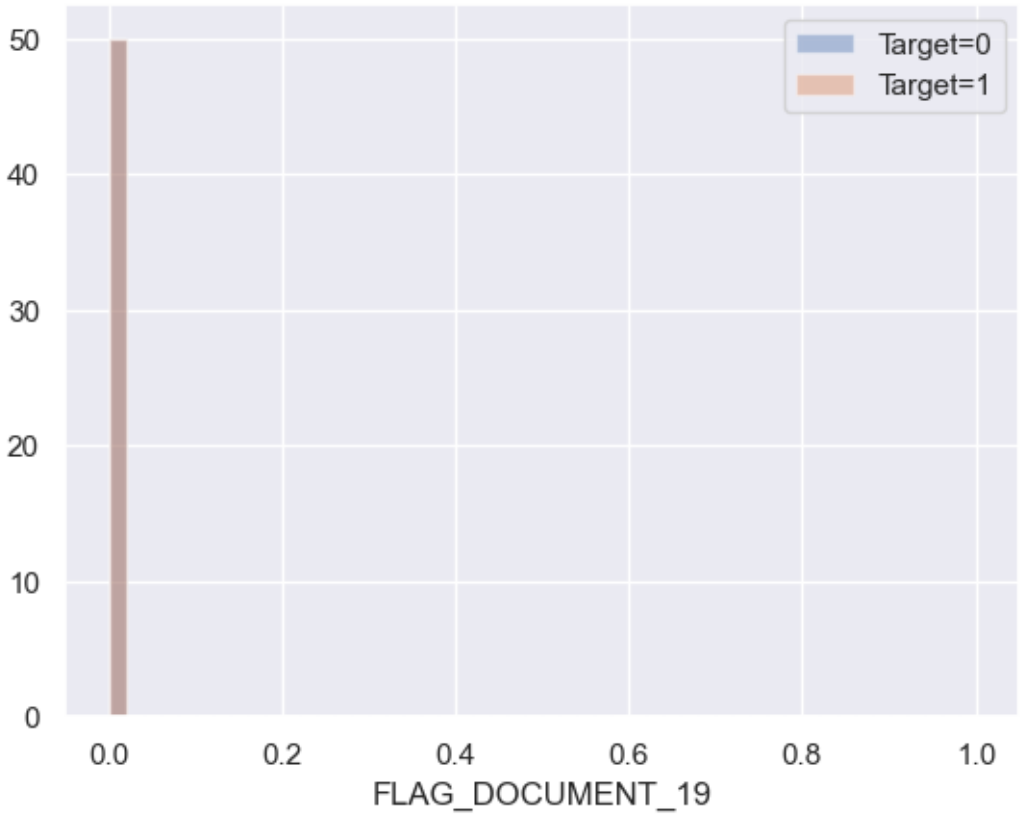
Plot of FLAG_DOCUMENT_18


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

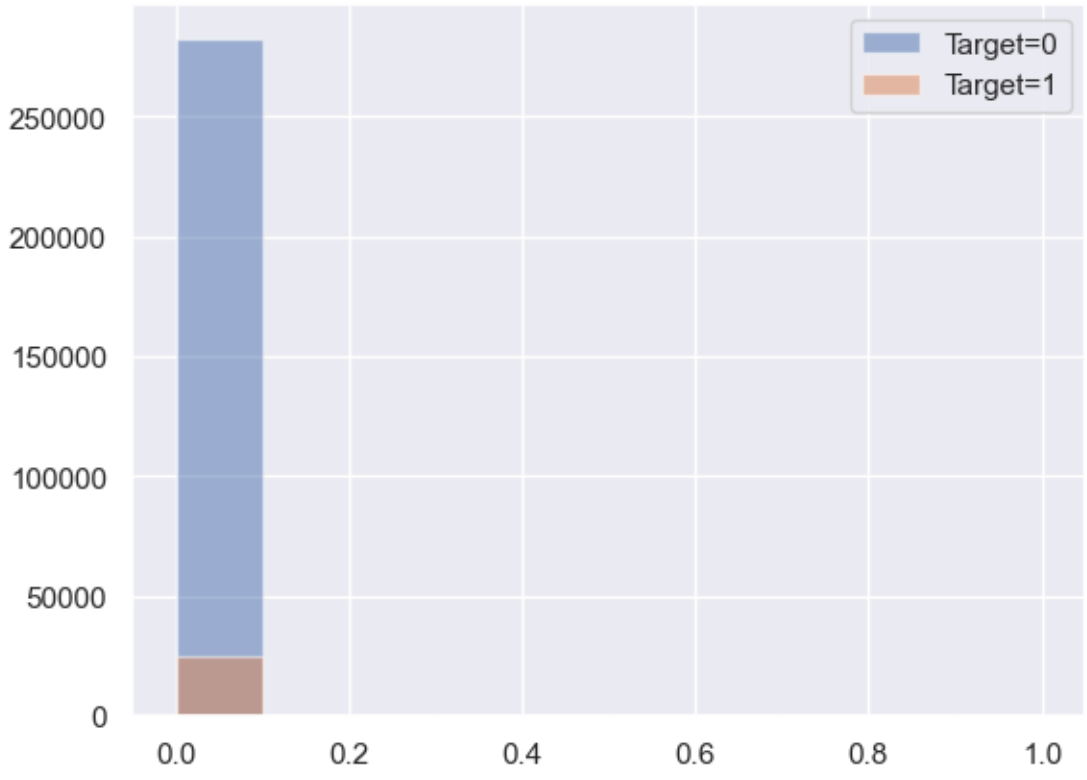


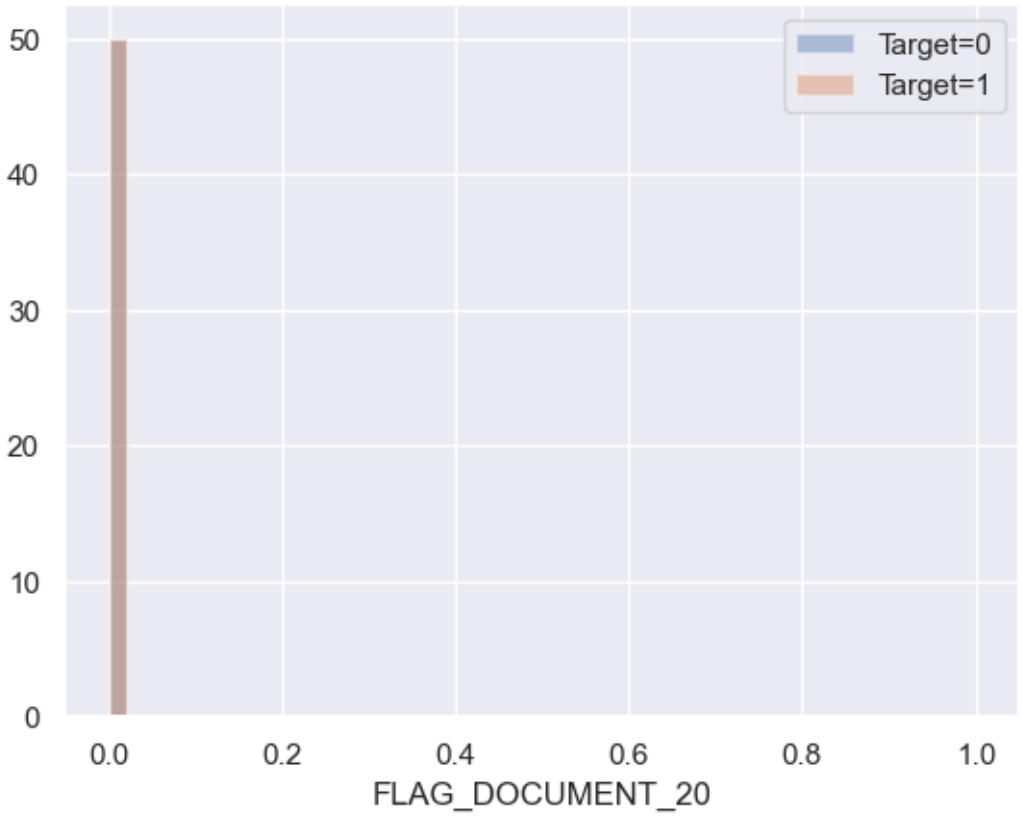
Plot of FLAG_DOCUMENT_19


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

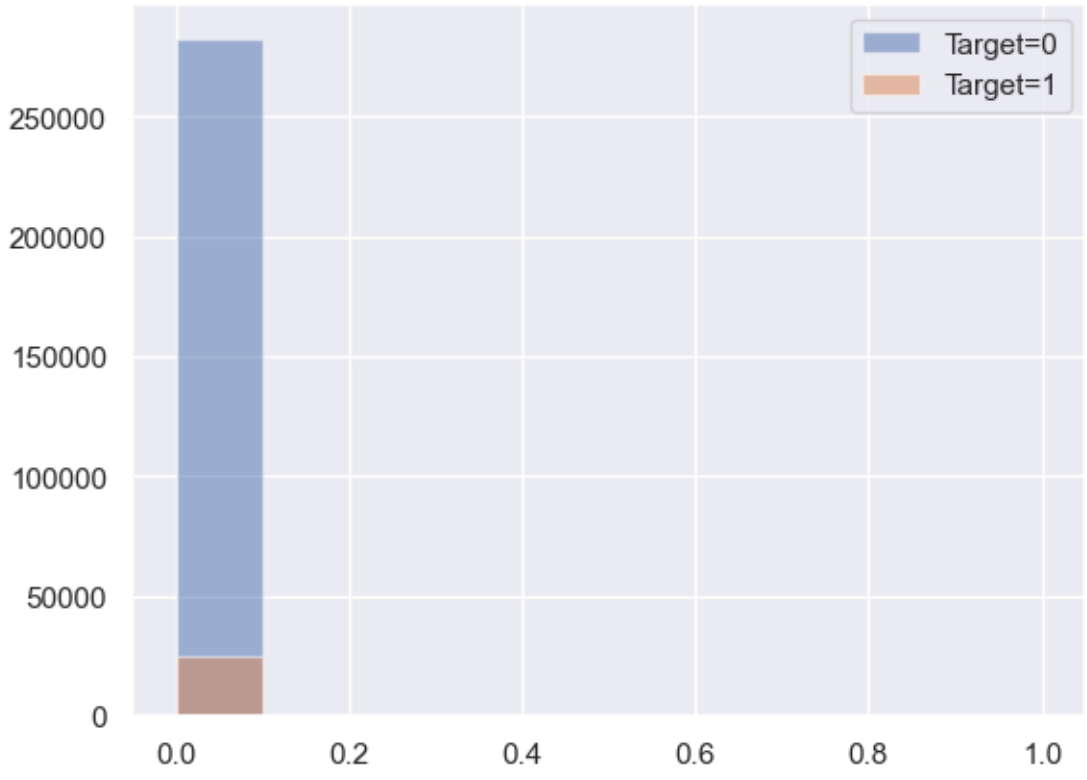


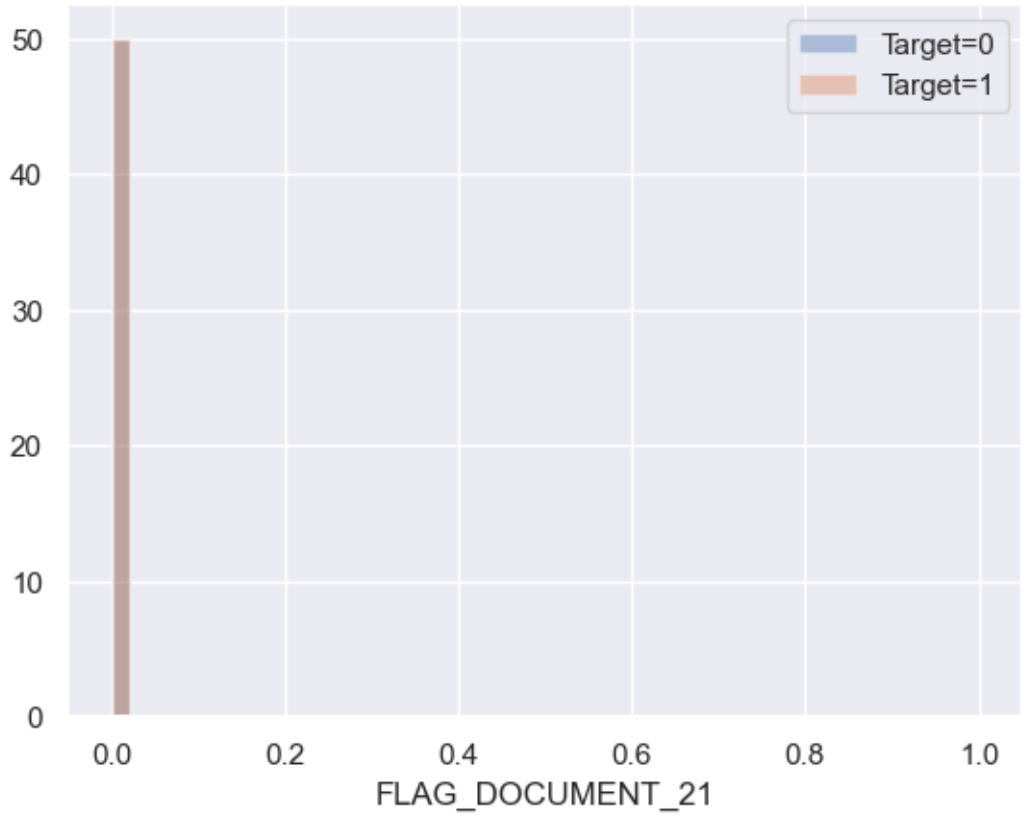
Plot of FLAG_DOCUMENT_20


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

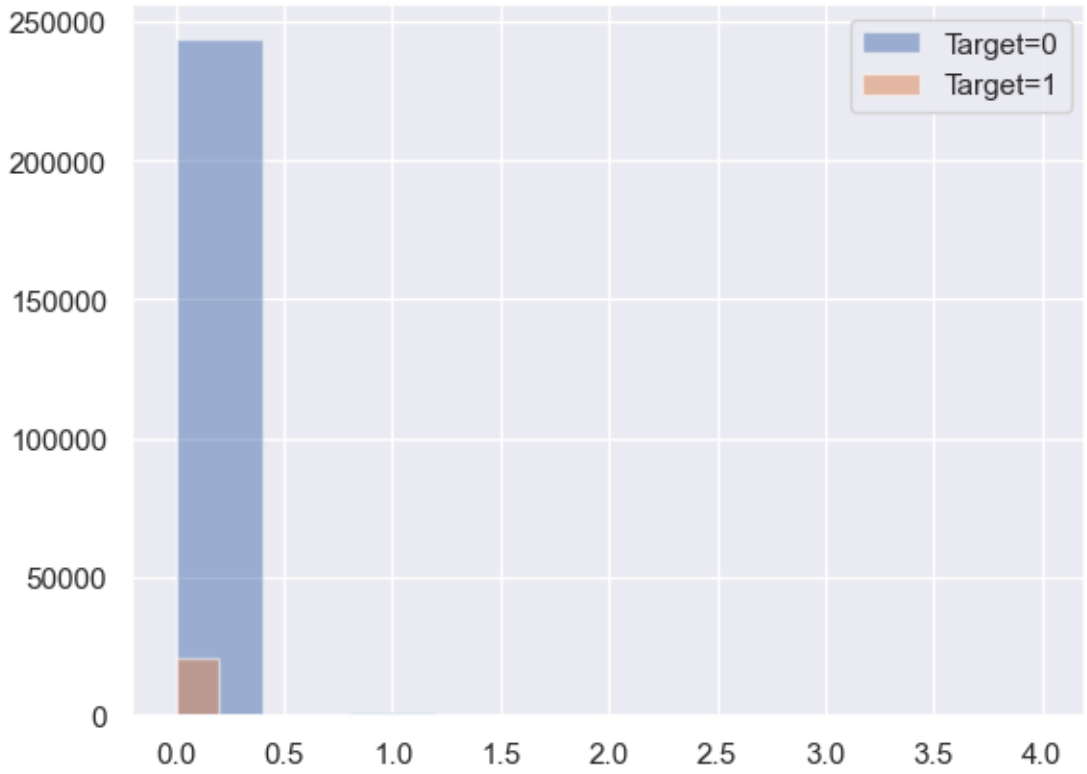


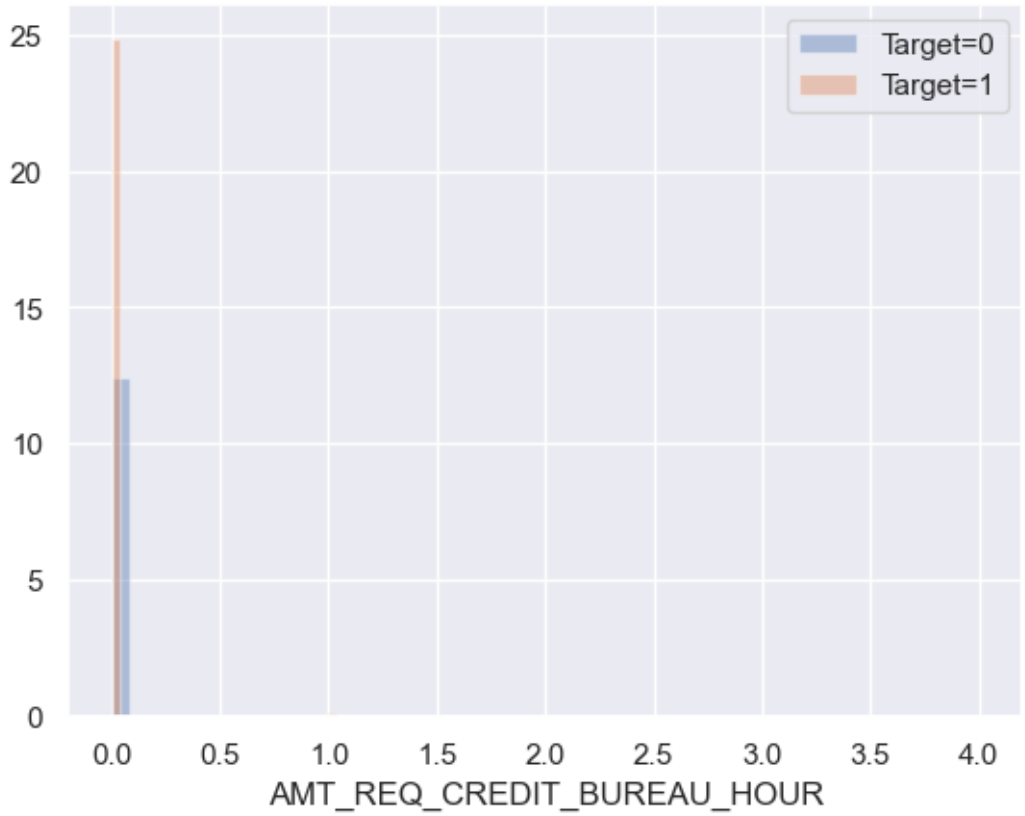
Plot of FLAG_DOCUMENT_21


Target=0
Target=1
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

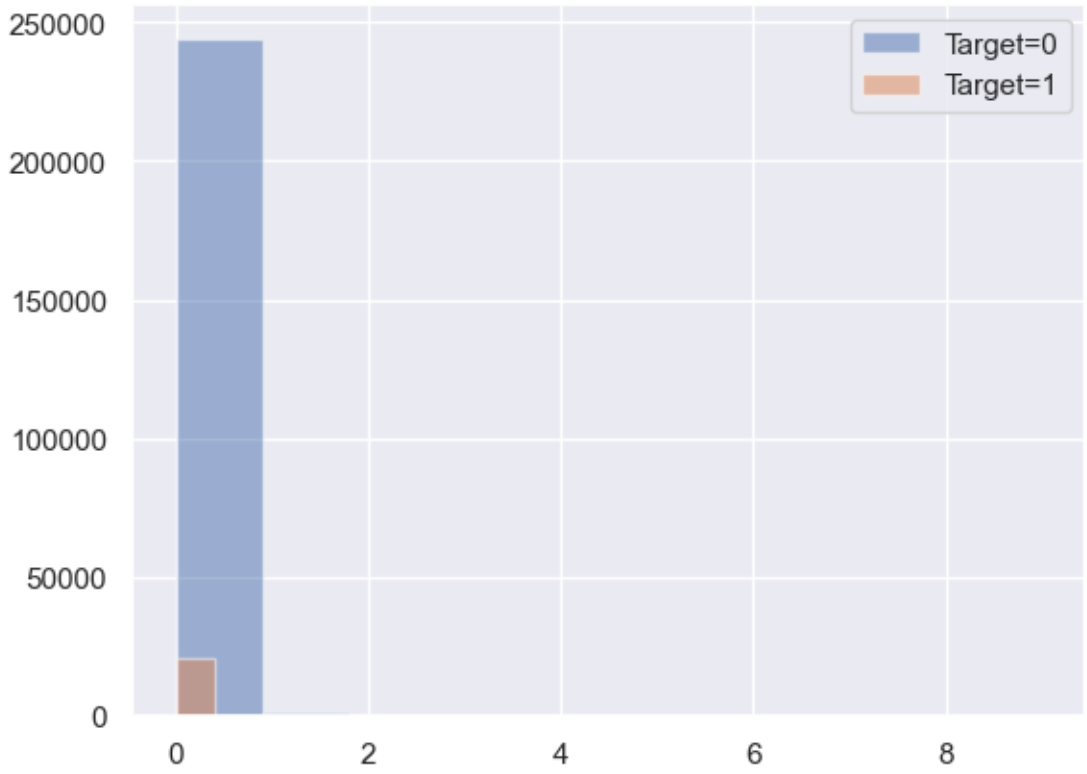


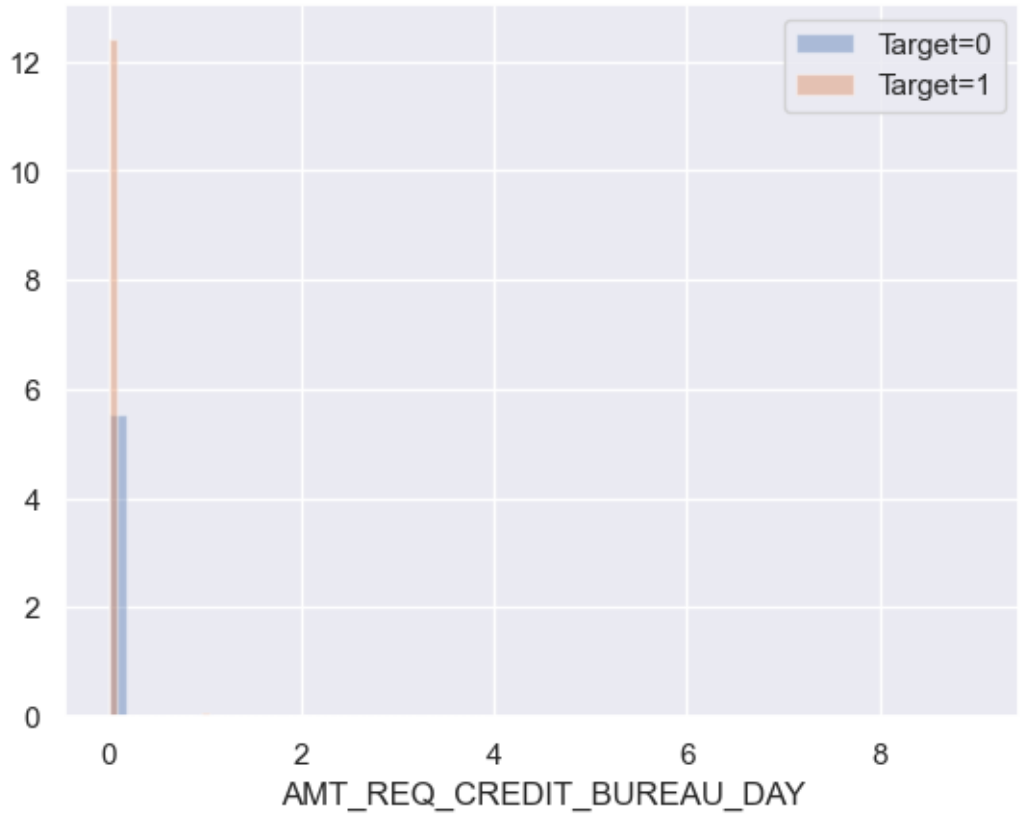
Plot of AMT_REQ_CREDIT_BUREAU_HOUR


250000
200000
150000
100000
50000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Target=0
Target=1

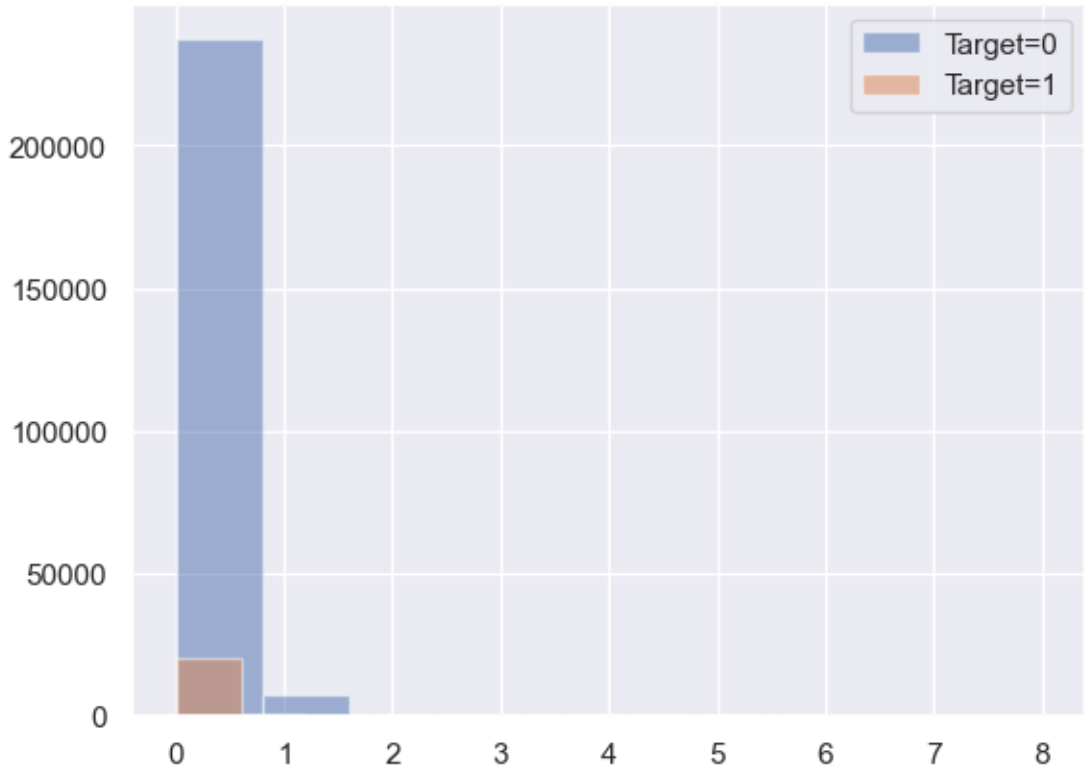


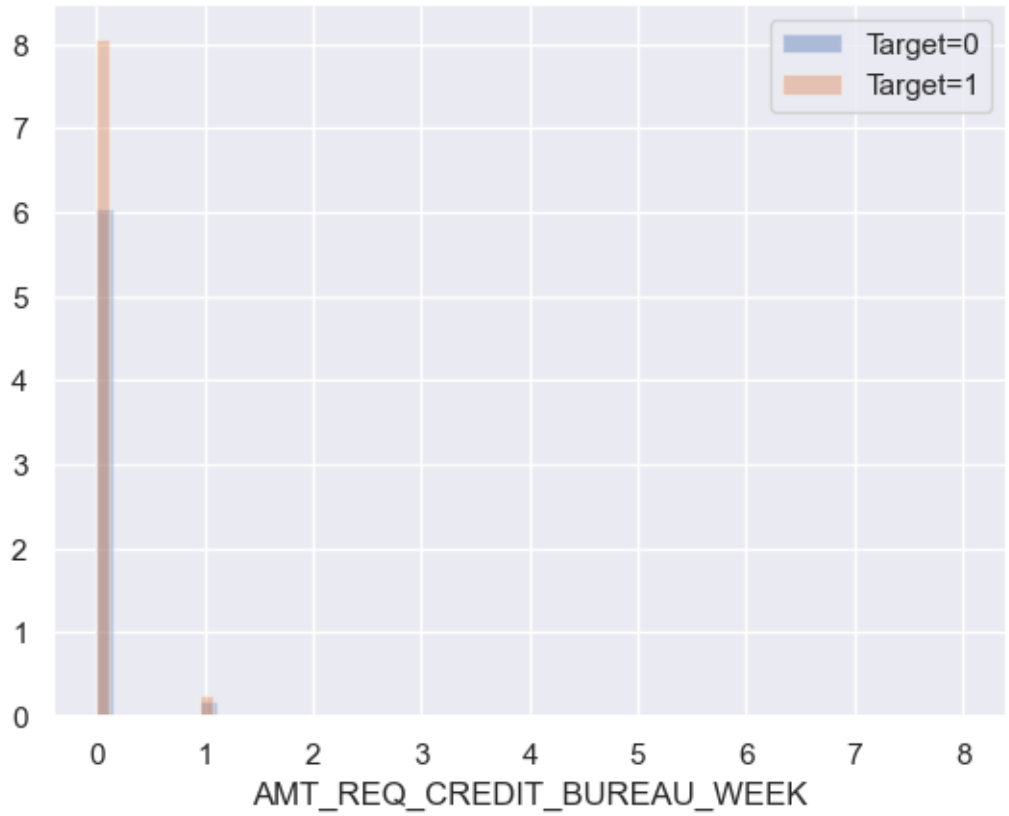
Plot of AMT_REQ_CREDIT_BUREAU_DAY


250000
200000
150000
100000
50000
0
0
2
4
6
8
Target=0
Target=1

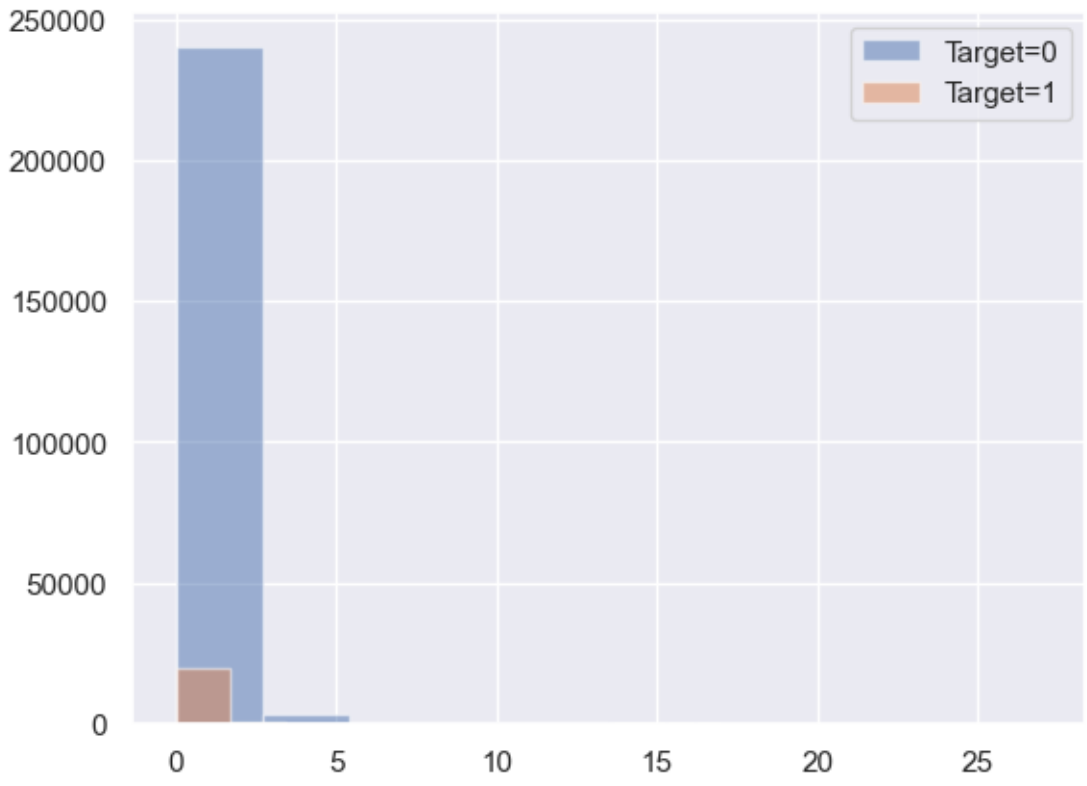


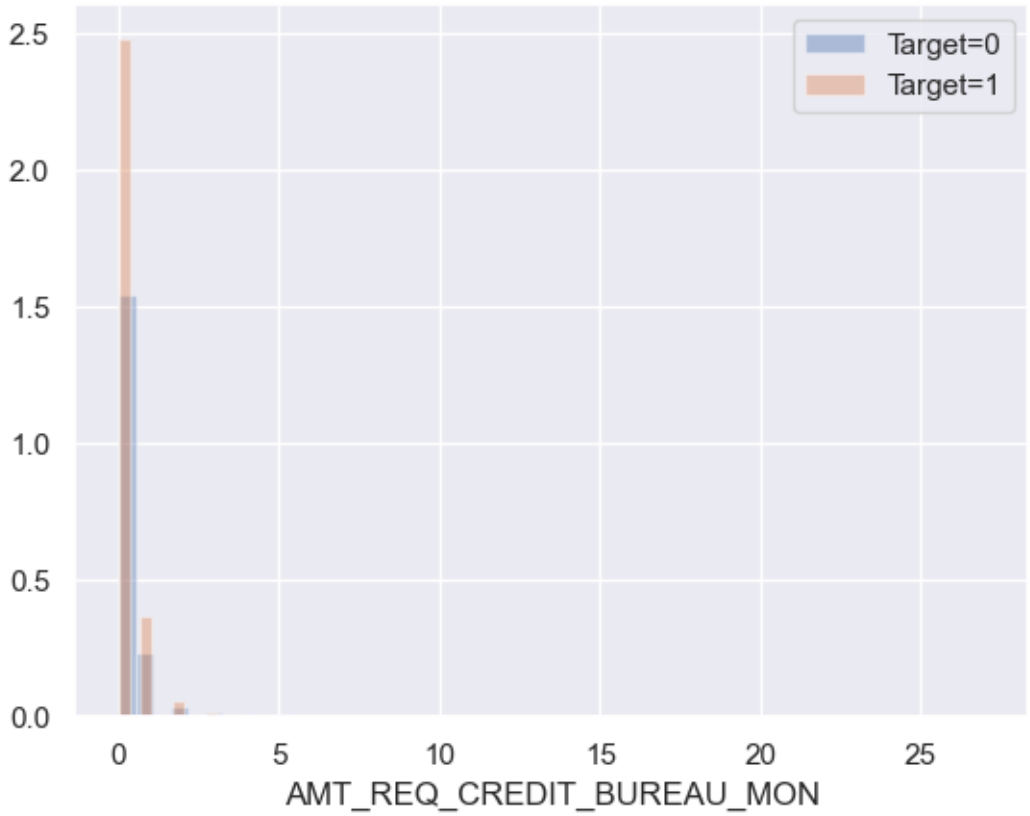
Plot of AMT_REQ_CREDIT_BUREAU_WEEK


Target=0
Target=1
200000
150000
100000
50000
0
0
1
2
3
4
5
6
7
8

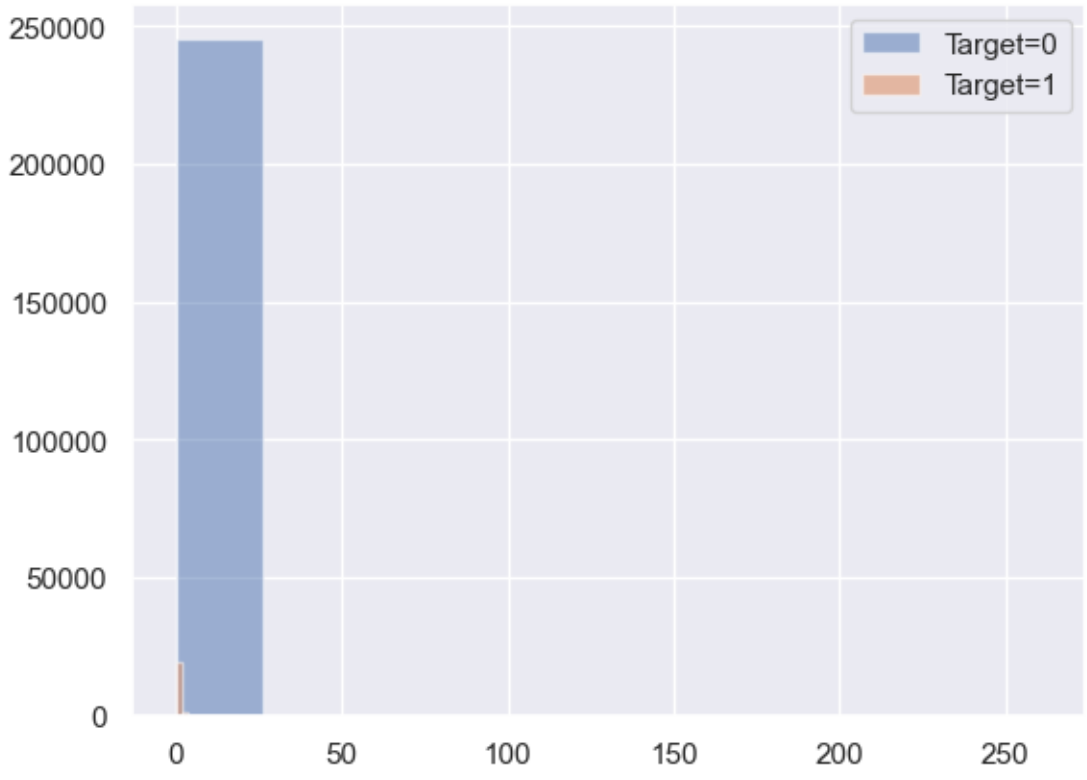


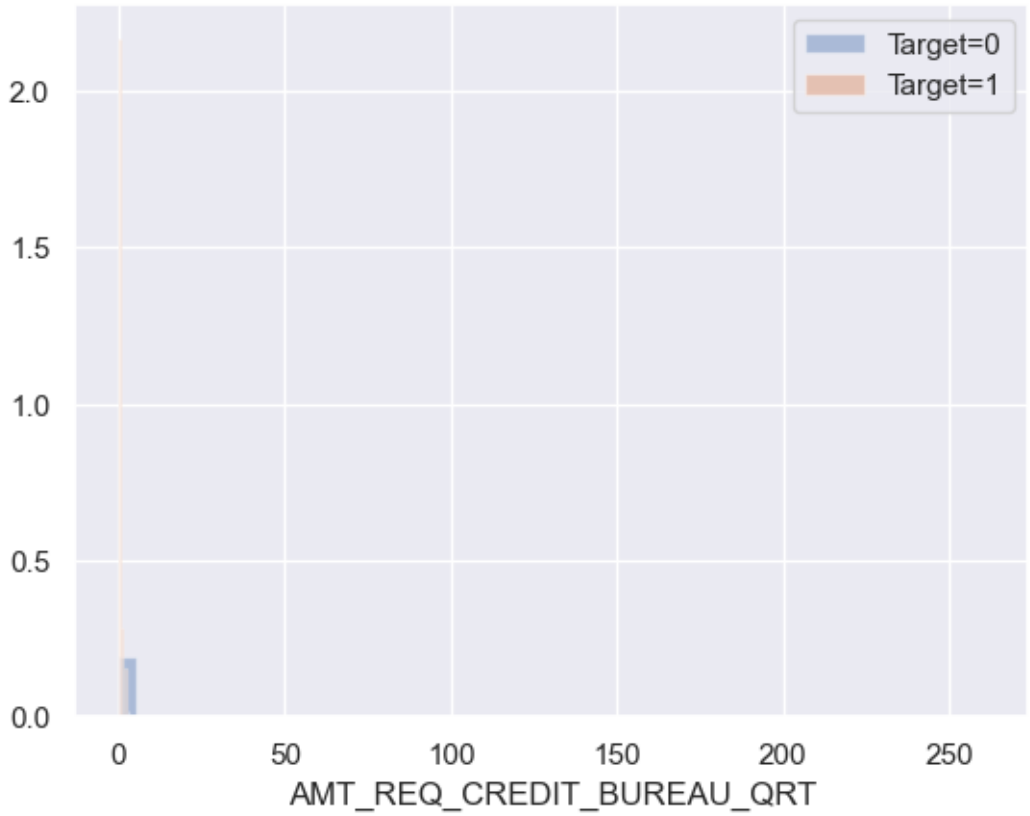

```
----------------------------------------------------------------------
Plot of AMT_REQ_CREDIT_BUREAU_MON
```


250000
200000
150000
100000
50000
0
0
5
10
15
20
25
Target=0
Target=1

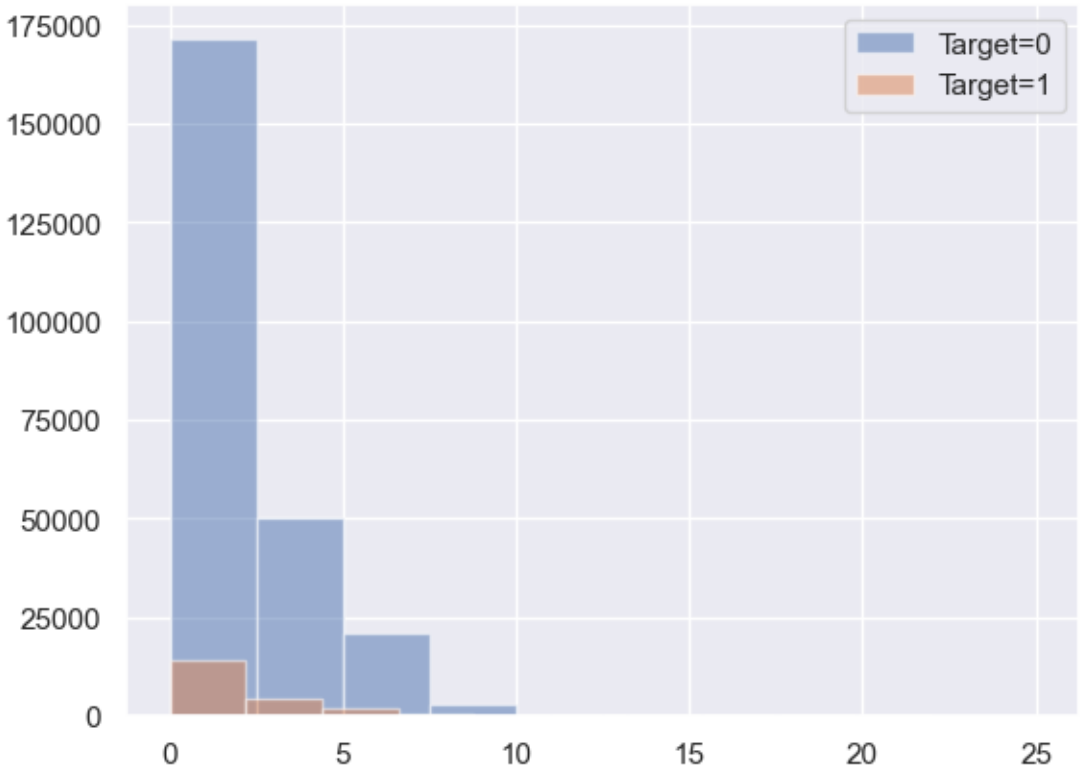


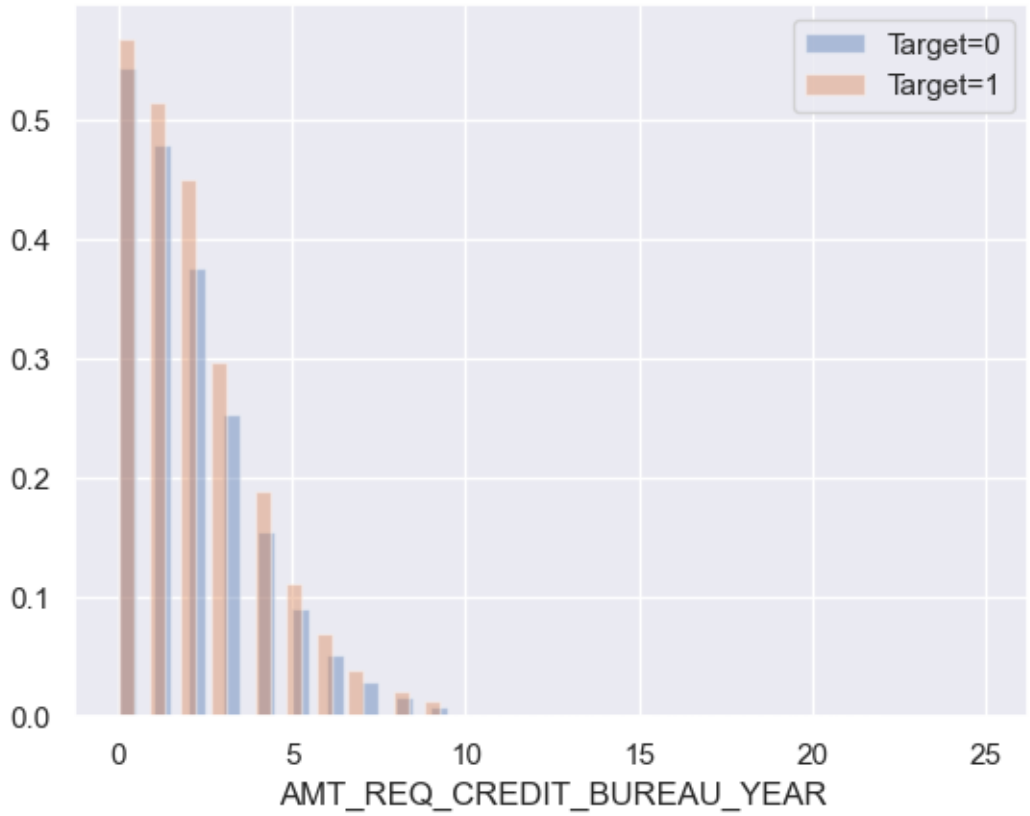
Plot of AMT_REQ_CREDIT_BUREAU_QRT


250000
200000
150000
100000
50000
0
0
50
100
150
200
250
Target=0
Target=1



Plot of AMT_REQ_CREDIT_BUREAU_YEAR


175000
150000
125000
100000
75000
50000
25000
0
0
5
10
15
20
25
Target=0
Target=1



---

[142]:
```
# The columns which seems different where both the histogram and distribution␣
 ↪plot are different for the target = 0 and target = 1 are:

# AMT_CREDIT:
# AMT_ANNUITY:
# AMT_GOODS_PRICE
# DAYS_BIRTH
# HOURS_APPR_PROCESS_START
# EXT_SOUCE_2
# EXT_SOURCE_3
# AMT_REQ_CREDIT_BUREAU_YEAR
```

[143]:
```
# Reading previous application

# Read the data from the file 'previous_application.csv' into a DataFrame
previous_application = pd.read_csv('previous_application.csv')

# Display the first few rows of the DataFrame 'previous_application'
previous_application.head()
```

[143]:
```
   SK_ID_PREV  SK_ID_CURR NAME_CONTRACT_TYPE  AMT_ANNUITY  AMT_APPLICATION  \
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

   AMT_CREDIT  AMT_DOWN_PAYMENT  AMT_GOODS_PRICE WEEKDAY_APPR_PROCESS_START  \
0     17145.0               0.0          17145.0                  SATURDAY
1    679671.0               NaN         607500.0                  THURSDAY
2    136444.5               NaN         112500.0                   TUESDAY
3    470790.0               NaN         450000.0                    MONDAY
4    404055.0               NaN         337500.0                  THURSDAY

   HOUR_APPR_PROCESS_START  … NAME_SELLER_INDUSTRY  CNT_PAYMENT  \
0                       15  …         Connectivity         12.0
1                       11  …                  XNA         36.0
2                       11  …                  XNA         12.0
3                        7  …                  XNA         12.0
4                        9  …                  XNA         24.0

   NAME_YIELD_GROUP       PRODUCT_COMBINATION  DAYS_FIRST_DRAWING  \
0            middle  POS mobile with interest            365243.0
1       low_action          Cash X-Sell: low            365243.0
2              high         Cash X-Sell: high            365243.0
3            middle       Cash X-Sell: middle            365243.0
4              high         Cash Street: high                 NaN

  DAYS_FIRST_DUE DAYS_LAST_DUE_1ST_VERSION  DAYS_LAST_DUE DAYS_TERMINATION  \
0          -42.0                    300.0         -42.0            -37.0
1         -134.0                    916.0      365243.0         365243.0
2         -271.0                     59.0      365243.0         365243.0
3         -482.0                   -152.0        -182.0           -177.0
4            NaN                      NaN           NaN              NaN

  NFLAG_INSURED_ON_APPROVAL
0                       0.0
1                       1.0
2                       1.0
3                       1.0
4                       NaN

[5 rows x 37 columns]
```

[144]:
```
# Shape of previous application
previous_application.shape
```

[144]: (1670214, 37)

[145]:
```
# There are duplicate 'SK_ID_CURR' as a person could have taken loan multiple␣
 ↪times

# Number of unique id in previous application
previous_application = previous_application.reset_index()
previous_application.SK_ID_PREV.value_counts()
```

[145]:
```
SK_ID_PREV
2030495    1
1035848    1
1526498    1
2148893    1
2437429    1
          ..
2811649    1
1221292    1
2780117    1
2194001    1
2418762    1
Name: count, Length: 1670214, dtype: int64
```

[146]:
```
# Number of unique id in previous application
previous_application.SK_ID_CURR.value_counts()
```

[146]:
```
SK_ID_CURR
187868    77
265681    73
173680    72
242412    68
206783    67
          ..
135285     1
311960     1
427136     1
241434     1
191629     1
Name: count, Length: 338857, dtype: int64
```

[147]:
```
# Task-6 - Merging Datasets
# Merging DataFrames: Train and Previous Application Based on SK_ID_PREV
```

[148]:
```
# Merge 'train' DataFrame with 'previous_application' DataFrame based on␣
 ↪'SK_ID_CURR'
# Using 'inner' join to retain only common rows between the two DataFrames
```

```
previous_train = train.merge(previous_application, left_on='SK_ID_CURR',␣
 ↪right_on='SK_ID_CURR', how='inner')
previous_train.head()
```

[148]:
```
   SK_ID_CURR TARGET NAME_CONTRACT_TYPE_x CODE_GENDER FLAG_OWN_CAR  \
0      100002      1           Cash loans           M            N
1      100003      0           Cash loans           F            N
2      100003      0           Cash loans           F            N
3      100003      0           Cash loans           F            N
4      100004      0      Revolving loans           M            Y

  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT_x  \
0               Y             0          202500.0      406597.5
1               N             0          270000.0     1293502.5
2               N             0          270000.0     1293502.5
3               N             0          270000.0     1293502.5
4               Y             0           67500.0      135000.0

   AMT_ANNUITY_x  …  NAME_SELLER_INDUSTRY CNT_PAYMENT NAME_YIELD_GROUP  \
0        24700.5  …       Auto technology        24.0       low_normal
1        35698.5  …                   XNA        12.0       low_normal
2        35698.5  …             Furniture         6.0           middle
3        35698.5  …  Consumer electronics        12.0           middle
4         6750.0  …          Connectivity         4.0           middle

          PRODUCT_COMBINATION DAYS_FIRST_DRAWING DAYS_FIRST_DUE  \
0      POS other with interest          365243.0         -565.0
1            Cash X-Sell: low          365243.0         -716.0
2   POS industry with interest          365243.0         -797.0
3  POS household with interest          365243.0        -2310.0
4  POS mobile without interest          365243.0         -784.0

   DAYS_LAST_DUE_1ST_VERSION  DAYS_LAST_DUE  DAYS_TERMINATION  \
0                      125.0          -25.0             -17.0
1                     -386.0         -536.0            -527.0
2                     -647.0         -647.0            -639.0
3                    -1980.0        -1980.0           -1976.0
4                     -694.0         -724.0            -714.0

   NFLAG_INSURED_ON_APPROVAL
0                        0.0
1                        1.0
2                        0.0
3                        1.0
4                        0.0

[5 rows x 119 columns]
```

[149]:
```
# Get the names of all columns in the DataFrame 'previous_application'
previous_application.columns.values
```

[149]:
```
array(['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)
```

[150]:
```
# The merged dataframe also has multiple values for SK_ID_CURR

# Count the occurrences of each unique value in the 'SK_ID_CURR' column of the␣
 ↪DataFrame 'previous_application' and display the top values
previous_application['SK_ID_CURR'].value_counts().head()
```

[150]:
```
SK_ID_CURR
187868    77
265681    73
173680    72
242412    68
206783    67
Name: count, dtype: int64
```

[151]:
```
# Segregating the dataset on Target = 0 and Target = 1
```

[152]:
```
# Create a subset of the DataFrame 'train' containing records where the␣
 ↪'TARGET' column is equal to '0'
train_0 = train.loc[train['TARGET'] == '0']

# Create a subset of the DataFrame 'train' containing records where the␣
 ↪'TARGET' column is equal to '1'
train_1 = train.loc[train['TARGET'] == '1']
```

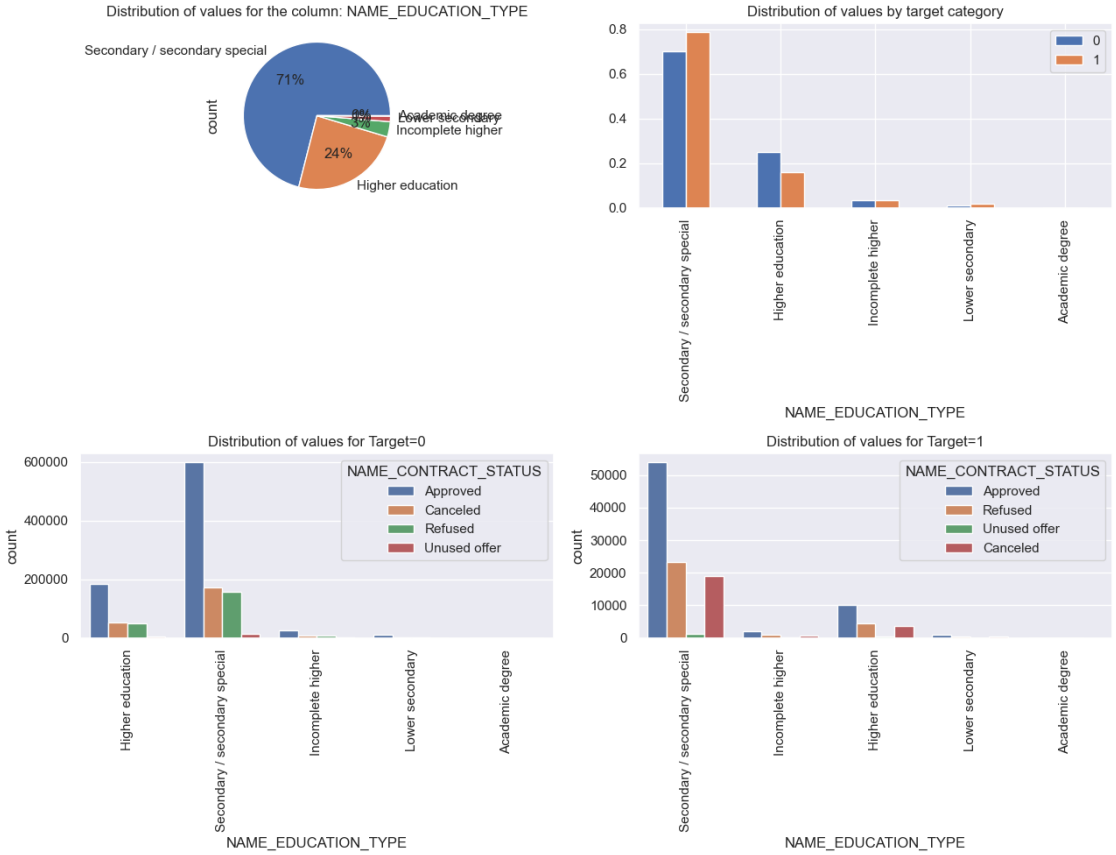
[153]:
```
# Create a subset of the DataFrame 'previous_train' containing records where␣
 ↪the 'TARGET' column is equal to '0'
ptrain_0 = previous_train.loc[previous_train['TARGET'] == '0']
```

```
# Create a subset of the DataFrame 'previous_train' containing records where␣
 ↪the 'TARGET' column is equal to '1'
ptrain_1 = previous_train.loc[previous_train['TARGET'] == '1']
```

[154]:
```
# Plotting data

def plotting(column, hue):
    # Assign column and hue parameters to local variables
    col = column
    hue = hue

    # Create a figure for the plots with a specific size
    fig = plt.figure(figsize=(13,10))

    # Subplot 1: Pie chart showing the distribution of values in the column
    ax1 = plt.subplot(221)
    train[col].value_counts().plot.pie(autopct="%1.0f%%", ax=ax1)
    plt.title('Distribution of values for the column: '+ column)

    # Subplot 2: Bar plot displaying the distribution of values by target␣
 ↪categories
    ax2 = plt.subplot(222)
    df = pd.DataFrame()
    df['0'] = ((train_0[col].value_counts()) / len(train_0))
    df['1'] = ((train_1[col].value_counts()) / len(train_1))
    df.plot.bar(ax=ax2)
    plt.title('Distribution of values by target category')

    # Subplot 3: Count plot showing the distribution of values for Target=0
    ax3 = plt.subplot(223)
    sns.countplot(x=col, hue=hue, data=ptrain_0, ax=ax3)
    plt.xticks(rotation=90)
    plt.title('Distribution of values for Target=0')

    # Subplot 4: Count plot showing the distribution of values for Target=1
    ax4 = plt.subplot(224)
    sns.countplot(x=col, hue=hue, data=ptrain_1, ax=ax4)
    plt.xticks(rotation=90)
    plt.title('Distribution of values for Target=1')

    # Adjust layout to prevent overlap
    fig.tight_layout()

    # Display the plots
    plt.show()
```

[155]:
```
# Task-7 - Bivariate Analysis
```

[156]:
```
# Plotting NAME_CONTRACT_STATUS

plotting('NAME_EDUCATION_TYPE','NAME_CONTRACT_STATUS')
```



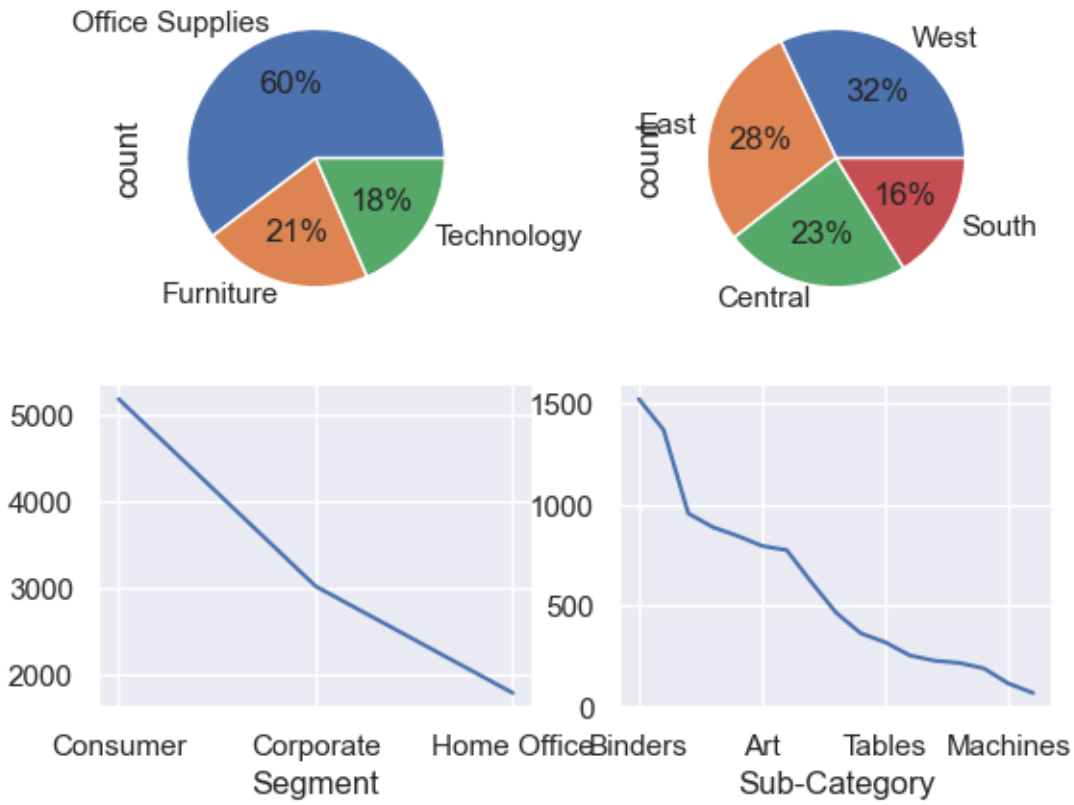
[157]:
```
import pandas as pd
# Conclusion
sales = pd.read_excel("Superstore.xls")
print(sales['Region'].value_counts().reset_index())
```

```
    Region  count
0     West   3203
1     East   2848
2  Central   2323
3    South   1620
```

[195]:
```
ax1 = plt.subplot(221)
sales['Category'].value_counts().plot.pie(autopct="%1.0f%%", ax = ax1)
ax2 = plt.subplot(222)
sales['Region'].value_counts().plot.pie(autopct="%1.0f%%", ax = ax2)
ax3 = plt.subplot(223)
```

```
sales['Segment'].value_counts().plot(ax=ax3)
ax4 = plt.subplot(224)
sales['Sub-Category'].value_counts().plot(ax=ax4)
```

[195]: <Axes: xlabel='Sub-Category'>



[159]:
```
sales.dtypes
```

[159]:
```
Row ID                     int64
Order ID                  object
Order Date        datetime64[ns]
Ship Date         datetime64[ns]
Ship Mode                 object
Customer ID               object
Customer Name             object
Segment                   object
Country                   object
City                      object
State                     object
Postal Code                int64
Region                    object
```

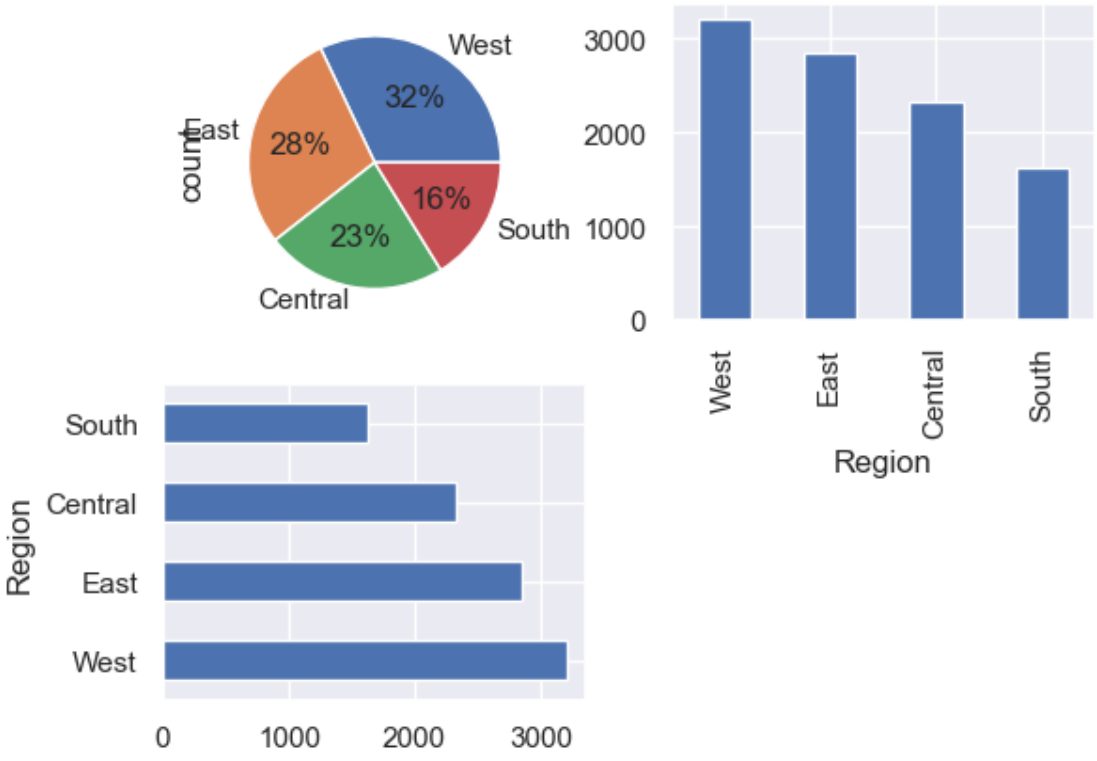
```
Product ID               object
Category                 object
Sub-Category             object
Product Name             object
Sales                   float64
Quantity                  int64
Discount                float64
Profit                  float64
dtype: object
```

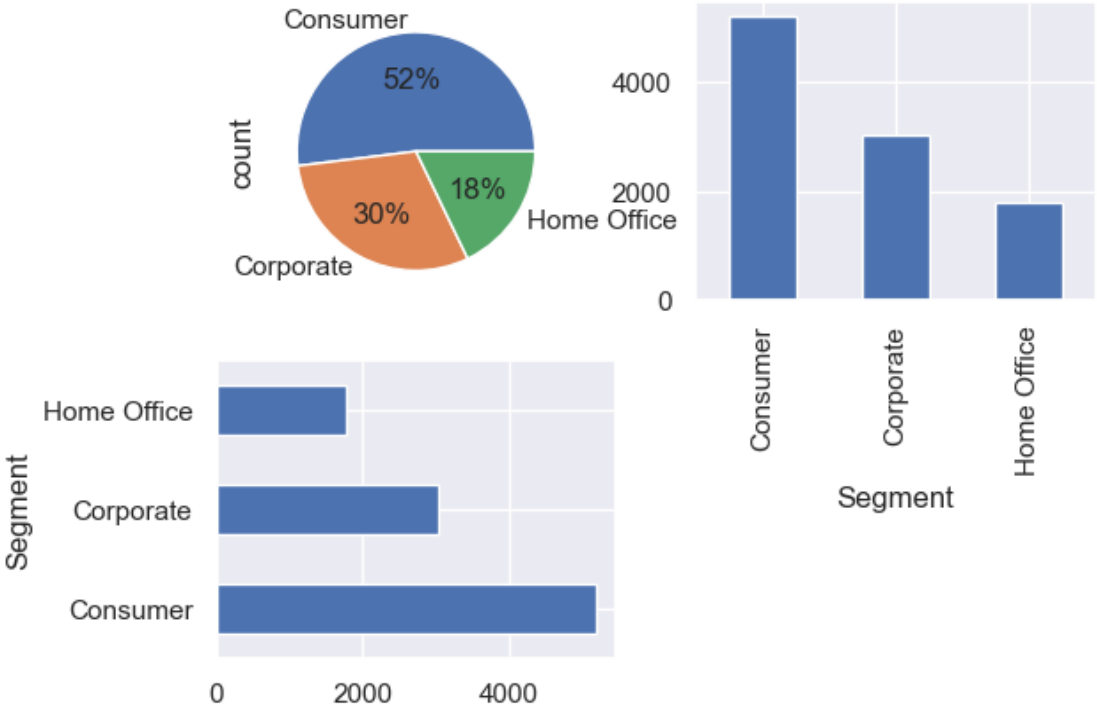
[160]:
```
sales.select_dtypes(include = 'int')
```

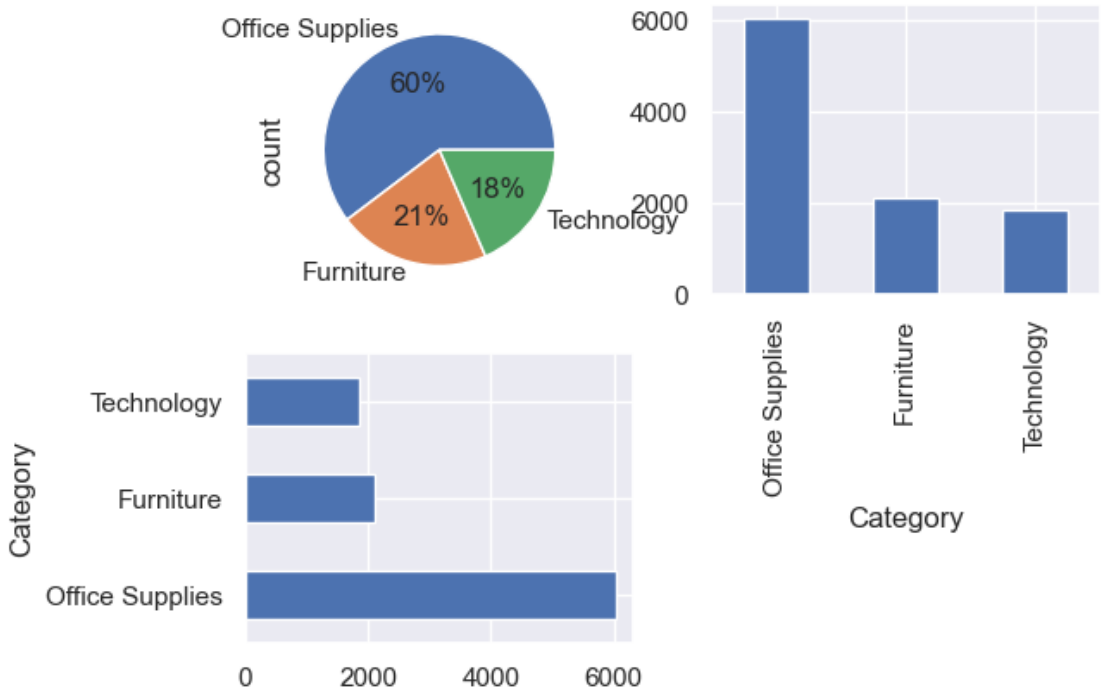
[160]:
```
      Row ID  Postal Code  Quantity
0          1        42420         2
1          2        42420         3
2          3        90036         2
3          4        33311         5
4          5        33311         2
…        …            …         …
9989    9990        33180         3
9990    9991        92627         2
9991    9992        92627         2
9992    9993        92627         4
9993    9994        92683         2

[9994 rows x 3 columns]
```

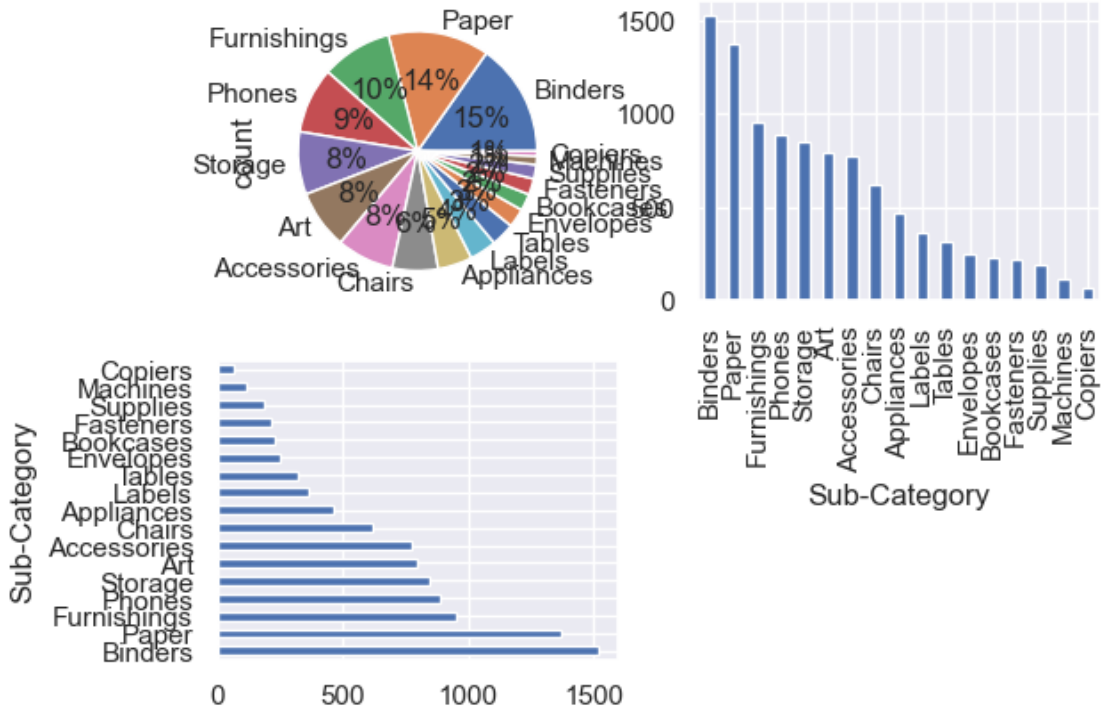
[161]:
```
cols = ["Region","Segment",'Category','Sub-Category']
```

[185]:
```
for col in cols:
    ax1 = plt.subplot(221)
    sales[col].value_counts().plot.pie(autopct="%1.0f%%", ax = ax1)
    ax2 = plt.subplot(222)
    sales[col].value_counts().plot(kind = 'bar',ax = ax2)
    ax3 = plt.subplot(223)
    sales[col].value_counts().plot(kind = 'barh',ax=ax3)
    plt.show()
```


West
32%
East
28%
16%
South
23%
Central
count
3000
2000
1000
0
West
East
Central
South
Region
South
Central
East
West
Region
0
1000
2000
3000


Consumer
52%
18%
30%
Corporate
Home Office
count
4000
2000
0
Consumer
Corporate
Home Office
Segment
Home Office
Corporate
Consumer
Segment
0
2000
4000


Office Supplies
60%
count
18%
21%
Technology
Furniture
6000
4000
2000
0
Office Supplies
Furniture
Technology
Category
Technology
Furniture
Office Supplies
Category
0
2000
4000
6000


Paper
Furnishings
14%
10%
Binders
Phones
9%
15%
count
Storage
8%
Art
8%
Accessories
Chairs
Appliances
Tables
Envelopes
Bookcases
Fasteners
Supplies
Machines
Copiers
1500
1000
0
Binders
Paper
Furnishings
Phones
Storage
Art
Accessories
Chairs
Appliances
Labels
Tables
Envelopes
Bookcases
Fasteners
Supplies
Machines
Copiers
Sub-Category
Sub-Category
Copiers
Machines
Supplies
Fasteners
Bookcases
Envelopes
Tables
Labels
Appliances
Chairs
Accessories
Art
Storage
Phones
Furnishings
Paper
Binders
0
500
1000
1500

[ ]: