Assessment_Project - House Loan Data Analysis

June 28, 2022

1 Home Loan Data Analysis

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# Import required library

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split
import tensorflow as tf
from keras.layers import Dense,Dropout
from keras.models import Sequential
from sklearn.impute import SimpleImputer
from sklearn.metrics import confusion_matrix,classification_report,roc_curve,auc
```

1.0.1 1. Load the dataset that is given to you

[3]:
```
# Load the given dataset

data = pd.read_csv('loan_data.csv')
data.head()
```

[3]:
```
   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]
```

[4]:
```
# Check the shape of the data

data.shape
```

[4]:
```
(307511, 122)
```

[5]:
```
# Check the size of the data

data.size
```

[5]:
```
37516342
```

[6]:
```
# Check the information of dataset
```

```
data.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
```

[7]:
```
data.columns
```

[7]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY',
       …
       'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',
       'FLAG_DOCUMENT_21', 'AMT_REQ_CREDIT_BUREAU_HOUR',
       'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
       'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object', length=122)
```

[8]:
```
data.describe()
```

[8]:
```
          SK_ID_CURR         TARGET   CNT_CHILDREN  AMT_INCOME_TOTAL  \
count  307511.000000  307511.000000  307511.000000      3.075110e+05
mean   278180.518577       0.080729       0.417052      1.687979e+05
std    102790.175348       0.272419       0.722121      2.371231e+05
min    100002.000000       0.000000       0.000000      2.565000e+04
25%    189145.500000       0.000000       0.000000      1.125000e+05
50%    278202.000000       0.000000       0.000000      1.471500e+05
75%    367142.500000       0.000000       1.000000      2.025000e+05
max    456255.000000       1.000000      19.000000      1.170000e+08

         AMT_CREDIT    AMT_ANNUITY  AMT_GOODS_PRICE  \
count  3.075110e+05  307499.000000     3.072330e+05
mean   5.990260e+05   27108.573909     5.383962e+05
std    4.024908e+05   14493.737315     3.694465e+05
min    4.500000e+04    1615.500000     4.050000e+04
25%    2.700000e+05   16524.000000     2.385000e+05
50%    5.135310e+05   24903.000000     4.500000e+05
75%    8.086500e+05   34596.000000     6.795000e+05
max    4.050000e+06  258025.500000     4.050000e+06

       REGION_POPULATION_RELATIVE     DAYS_BIRTH  DAYS_EMPLOYED  …  \
count               307511.000000  307511.000000  307511.000000  …
mean                     0.020868  -16036.995067   63815.045904  …
std                      0.013831    4363.988632  141275.766519  …
```

```
min                    0.000290  -25229.000000  -17912.000000  …
25%                    0.010006  -19682.000000   -2760.000000  …
50%                    0.018850  -15750.000000   -1213.000000  …
75%                    0.028663  -12413.000000    -289.000000  …
max                    0.072508   -7489.000000  365243.000000  …

       FLAG_DOCUMENT_18  FLAG_DOCUMENT_19  FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  \
count     307511.000000     307511.000000     307511.000000     307511.000000
mean           0.008130          0.000595          0.000507          0.000335
std            0.089798          0.024387          0.022518          0.018299
min            0.000000          0.000000          0.000000          0.000000
25%            0.000000          0.000000          0.000000          0.000000
50%            0.000000          0.000000          0.000000          0.000000
75%            0.000000          0.000000          0.000000          0.000000
max            1.000000          1.000000          1.000000          1.000000

       AMT_REQ_CREDIT_BUREAU_HOUR  AMT_REQ_CREDIT_BUREAU_DAY  \
count               265992.000000              265992.000000
mean                     0.006402                   0.007000
std                      0.083849                   0.110757
min                      0.000000                   0.000000
25%                      0.000000                   0.000000
50%                      0.000000                   0.000000
75%                      0.000000                   0.000000
max                      4.000000                   9.000000

       AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
count               265992.000000              265992.000000
mean                     0.034362                   0.267395
std                      0.204685                   0.916002
min                      0.000000                   0.000000
25%                      0.000000                   0.000000
50%                      0.000000                   0.000000
75%                      0.000000                   0.000000
max                      8.000000                  27.000000

       AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
count              265992.000000               265992.000000
mean                    0.265474                    1.899974
std                     0.794056                    1.869295
min                     0.000000                    0.000000
25%                     0.000000                    0.000000
50%                     0.000000                    1.000000
75%                     0.000000                    3.000000
max                   261.000000                   25.000000

[8 rows x 106 columns]
```

1.0.2 2. Check for null values in the dataset

[9]:
```
# Find out the Null or missing value in dataset

data.isnull().sum().any()
```

[9]:
```
True
```

[10]:
```
null_data = data.isna().sum()/data.shape[0]*100
null_data.sort_values(ascending=False,inplace=True)
null_data = null_data.reset_index()
null_data.head(40)
```

[10]:

	index	0
0	COMMONAREA_MEDI	69.872297
1	COMMONAREA_AVG	69.872297
2	COMMONAREA_MODE	69.872297
3	NONLIVINGAPARTMENTS_MODE	69.432963
4	NONLIVINGAPARTMENTS_AVG	69.432963
5	NONLIVINGAPARTMENTS_MEDI	69.432963
6	FONDKAPREMONT_MODE	68.386172
7	LIVINGAPARTMENTS_MODE	68.354953
8	LIVINGAPARTMENTS_AVG	68.354953
9	LIVINGAPARTMENTS_MEDI	68.354953
10	FLOORSMIN_AVG	67.848630
11	FLOORSMIN_MODE	67.848630
12	FLOORSMIN_MEDI	67.848630
13	YEARS_BUILD_MEDI	66.497784
14	YEARS_BUILD_MODE	66.497784
15	YEARS_BUILD_AVG	66.497784
16	OWN_CAR_AGE	65.990810
17	LANDAREA_MEDI	59.376738
18	LANDAREA_MODE	59.376738
19	LANDAREA_AVG	59.376738
20	BASEMENTAREA_MEDI	58.515956
21	BASEMENTAREA_AVG	58.515956
22	BASEMENTAREA_MODE	58.515956
23	EXT_SOURCE_1	56.381073
24	NONLIVINGAREA_MODE	55.179164
25	NONLIVINGAREA_AVG	55.179164
26	NONLIVINGAREA_MEDI	55.179164
27	ELEVATORS_MEDI	53.295980
28	ELEVATORS_AVG	53.295980
29	ELEVATORS_MODE	53.295980
30	WALLSMATERIAL_MODE	50.840783
31	APARTMENTS_MEDI	50.749729
32	APARTMENTS_AVG	50.749729
33	APARTMENTS_MODE	50.749729

```
34             ENTRANCES_MEDI  50.348768
35              ENTRANCES_AVG  50.348768
36             ENTRANCES_MODE  50.348768
37              LIVINGAREA_AVG  50.193326
38             LIVINGAREA_MODE  50.193326
39             LIVINGAREA_MEDI  50.193326
```

[11]:
```python
col = null_data['index'].head(40)
col
```

[11]:
```
0                COMMONAREA_MEDI
1                 COMMONAREA_AVG
2                COMMONAREA_MODE
3      NONLIVINGAPARTMENTS_MODE
4       NONLIVINGAPARTMENTS_AVG
5      NONLIVINGAPARTMENTS_MEDI
6            FONDKAPREMONT_MODE
7         LIVINGAPARTMENTS_MODE
8          LIVINGAPARTMENTS_AVG
9         LIVINGAPARTMENTS_MEDI
10                FLOORSMIN_AVG
11               FLOORSMIN_MODE
12               FLOORSMIN_MEDI
13             YEARS_BUILD_MEDI
14             YEARS_BUILD_MODE
15              YEARS_BUILD_AVG
16                  OWN_CAR_AGE
17                LANDAREA_MEDI
18                LANDAREA_MODE
19                 LANDAREA_AVG
20             BASEMENTAREA_MEDI
21              BASEMENTAREA_AVG
22             BASEMENTAREA_MODE
23                 EXT_SOURCE_1
24           NONLIVINGAREA_MODE
25            NONLIVINGAREA_AVG
26           NONLIVINGAREA_MEDI
27               ELEVATORS_MEDI
28                ELEVATORS_AVG
29               ELEVATORS_MODE
30           WALLSMATERIAL_MODE
31              APARTMENTS_MEDI
32               APARTMENTS_AVG
33              APARTMENTS_MODE
34               ENTRANCES_MEDI
35                ENTRANCES_AVG
36               ENTRANCES_MODE
```

```
37                LIVINGAREA_AVG
38               LIVINGAREA_MODE
39               LIVINGAREA_MEDI
Name: index, dtype: object
```

[12]:
```
data = data.drop(col,axis=1)
```

[13]:
```
data.shape
```

[13]:
```
(307511, 82)
```

1.0.3 3. Print percentage of default to payer of the dataset for the TARGET column

[14]:
```
data.TARGET.value_counts()
```

[14]:
```
0    282686
1     24825
Name: TARGET, dtype: int64
```

[15]:
```
defaulters = (data['TARGET'] == 1).sum()
payers = (data['TARGET'] == 0).sum()
print('Defaulters : ',defaulters)
print('Payers : ',payers)
```

```
Defaulters :  24825
Payers :  282686
```

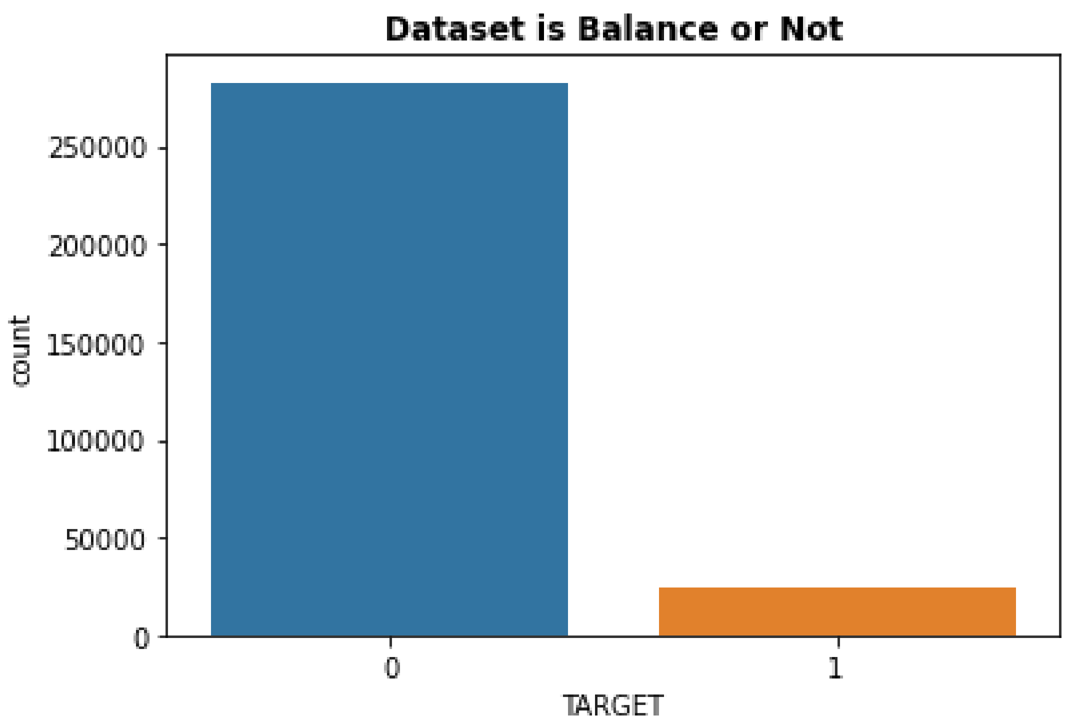
[16]:
```
default_perc = (defaulters/payers)*100
print("Percentage of default to payer is {:.2f} %".format(default_perc))
```

```
Percentage of default to payer is 8.78 %
```

1.0.4 4. Balance the dataset if the data is imbalanced

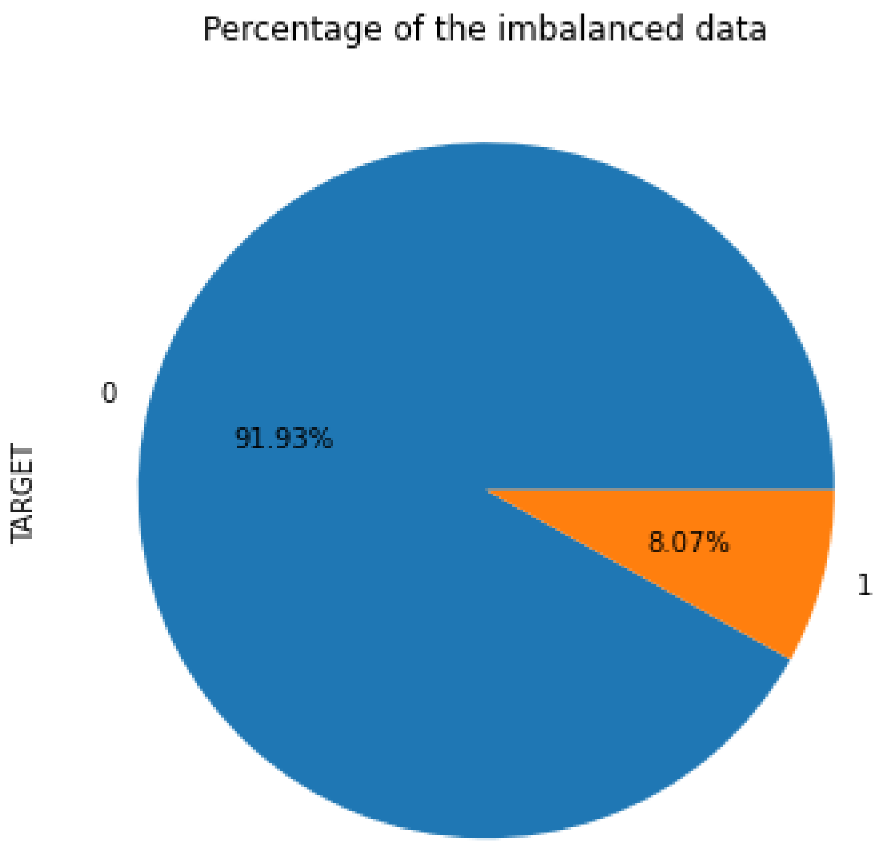
[17]:
```
sns.countplot(data['TARGET'])
plt.title('Dataset is Balance or Not',weight='bold',fontsize=12)
```

[17]:
```
Text(0.5, 1.0, 'Dataset is Balance or Not')
```



[18]:
```
data.TARGET.value_counts().plot(kind='pie',autopct='%.2f%%', title='Percentage␣
↪of the imbalanced data',figsize=(6,6))
```

[18]:
```
<AxesSubplot:title={'center':'Percentage of the imbalanced data'},
ylabel='TARGET'>
```

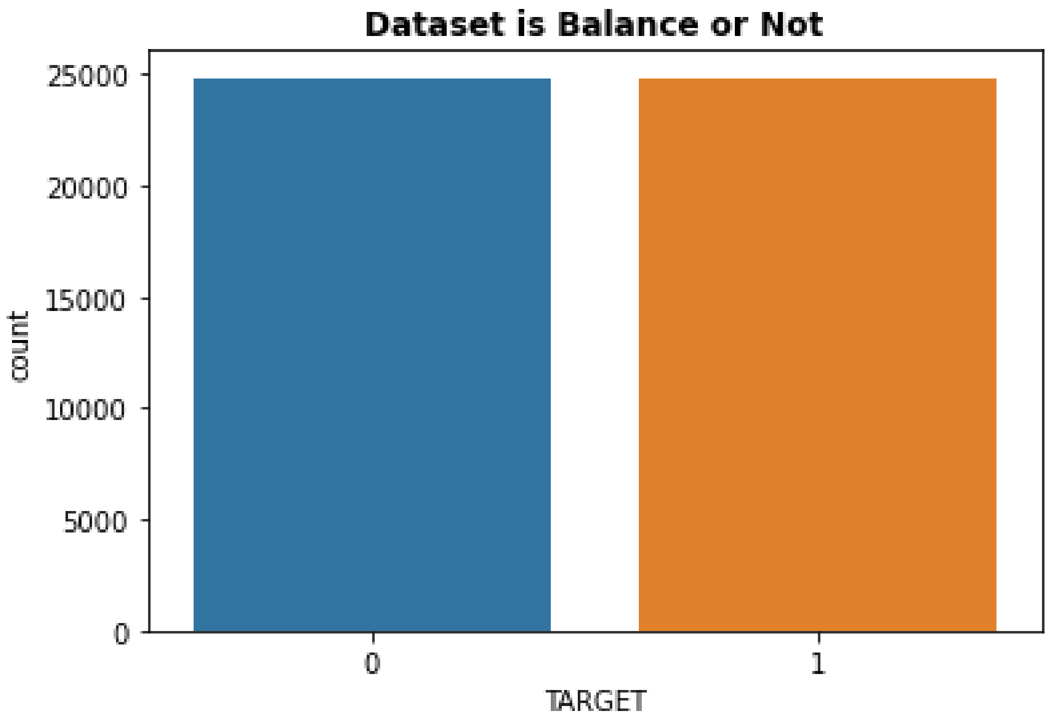


[19]:
```
shuffled_data = data.sample(frac=1,random_state=20)
fraud_data = shuffled_data[shuffled_data['TARGET'] == 1]
nonfraud_data = shuffled_data[shuffled_data['TARGET'] == 0].
 ↪sample(n=24825,random_state=20)
balance_data = pd.concat([fraud_data,nonfraud_data])
```

1.0.5 5. Plot the balanced data or imbalanced data

[20]:
```
sns.countplot(balance_data['TARGET'])
plt.title('Dataset is Balance or Not',weight='bold',fontsize=12)
```

[20]:
```
Text(0.5, 1.0, 'Dataset is Balance or Not')
```



[21]:
```
balance_data.describe(include='object')
```

[21]:
```
       NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  \
count               49650       49650        49650           49650
unique                  2           2            2               2
top            Cash loans           F            N               Y
freq                45615       30662        33566           34310

       NAME_TYPE_SUITE NAME_INCOME_TYPE            NAME_EDUCATION_TYPE  \
count            49471            49650                          49650
unique               7                7                              5
top      Unaccompanied          Working  Secondary / secondary special
freq             40445            27817                          37082

       NAME_FAMILY_STATUS  NAME_HOUSING_TYPE OCCUPATION_TYPE  \
count               49650              49650           35423
unique                  5                  6              18
top               Married  House / apartment        Laborers
freq                30663              43484           10217

       WEEKDAY_APPR_PROCESS_START       ORGANIZATION_TYPE  HOUSETYPE_MODE  \
count                       49650                   49650           23321
unique                          7                      58               3
```

```
top                          TUESDAY  Business Entity Type 3  block of flats
freq                            8828                   11642           22871

       EMERGENCYSTATE_MODE
count                24616
unique                   2
top                     No
freq                 24196
```

1.0.6 6. Encode the columns that is required for the model

[22]:
```python
# Find the categorical values in the dataset and and encode them

dictionary = {}
dictionary['categorical'] = balance_data.dtypes[balance_data.dtypes ==␣
 ↪'object'].index
dictionary
```

[22]:
```
{'categorical': 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', 'HOUSETYPE_MODE',
        'EMERGENCYSTATE_MODE'],
       dtype='object')}
```

[23]:
```python
for column in balance_data:
    if balance_data[column].dtypes == 'object':
        print(f'{column} : {balance_data[column].unique()}')
```

```
NAME_CONTRACT_TYPE : ['Cash loans' 'Revolving loans']
CODE_GENDER : ['M' 'F']
FLAG_OWN_CAR : ['N' 'Y']
FLAG_OWN_REALTY : ['N' 'Y']
NAME_TYPE_SUITE : ['Unaccompanied' 'Family' 'Other_A' 'Spouse, partner'
'Other_B' 'Children'
 nan 'Group of people']
NAME_INCOME_TYPE : ['Working' 'Commercial associate' 'Pensioner' 'State servant'
'Unemployed'
 'Maternity leave' 'Student']
NAME_EDUCATION_TYPE : ['Secondary / secondary special' 'Higher education'
'Incomplete higher'
 'Lower secondary' 'Academic degree']
NAME_FAMILY_STATUS : ['Single / not married' 'Married' 'Civil marriage'
'Separated' 'Widow']
NAME_HOUSING_TYPE : ['House / apartment' 'Rented apartment' 'Municipal
apartment'
```

```
 'With parents' 'Office apartment' 'Co-op apartment']
OCCUPATION_TYPE : ['Laborers' 'Sales staff' nan 'Managers' 'Secretaries'
 'High skill tech staff' 'Medicine staff' 'Core staff'
 'Low-skill Laborers' 'Drivers' 'Security staff' 'Cleaning staff'
 'Cooking staff' 'Accountants' 'Waiters/barmen staff' 'HR staff'
 'Private service staff' 'Realty agents' 'IT staff']
WEEKDAY_APPR_PROCESS_START : ['WEDNESDAY' 'MONDAY' 'TUESDAY' 'FRIDAY' 'SATURDAY'
'THURSDAY' 'SUNDAY']
ORGANIZATION_TYPE : ['Construction' 'Business Entity Type 3' 'Self-employed'
 'Business Entity Type 1' 'Industry: type 3' 'XNA' 'Kindergarten' 'Other'
 'Telecom' 'Security Ministries' 'Business Entity Type 2' 'Bank'
 'Medicine' 'School' 'Government' 'University' 'Postal' 'Trade: type 7'
 'Security' 'Trade: type 2' 'Industry: type 1' 'Transport: type 3'
 'Trade: type 3' 'Transport: type 4' 'Industry: type 11' 'Military'
 'Restaurant' 'Agriculture' 'Hotel' 'Police' 'Housing' 'Religion'
 'Services' 'Electricity' 'Industry: type 2' 'Transport: type 2'
 'Industry: type 4' 'Realtor' 'Insurance' 'Industry: type 9' 'Advertising'
 'Transport: type 1' 'Industry: type 5' 'Trade: type 1' 'Industry: type 7'
 'Legal Services' 'Culture' 'Trade: type 6' 'Mobile' 'Cleaning'
 'Emergency' 'Industry: type 12' 'Industry: type 10' 'Industry: type 6'
 'Industry: type 13' 'Industry: type 8' 'Trade: type 4' 'Trade: type 5']
HOUSETYPE_MODE : ['block of flats' nan 'terraced house' 'specific housing']
EMERGENCYSTATE_MODE : ['No' nan 'Yes']
```

[24]:

```
encoder = LabelEncoder()
balance_data['NAME_CONTRACT_TYPE'] = encoder.
 ↪fit_transform(balance_data['NAME_CONTRACT_TYPE'])
balance_data['CODE_GENDER'] = encoder.fit_transform(balance_data['CODE_GENDER'])
balance_data['FLAG_OWN_CAR'] = encoder.
 ↪fit_transform(balance_data['FLAG_OWN_CAR'])
balance_data['NAME_INCOME_TYPE'] = encoder.
 ↪fit_transform(balance_data['NAME_INCOME_TYPE'])
balance_data['NAME_EDUCATION_TYPE'] = encoder.
 ↪fit_transform(balance_data['NAME_EDUCATION_TYPE'])
balance_data['NAME_FAMILY_STATUS'] = encoder.
 ↪fit_transform(balance_data['NAME_FAMILY_STATUS'])
balance_data['NAME_HOUSING_TYPE'] = encoder.
 ↪fit_transform(balance_data['NAME_HOUSING_TYPE'])
balance_data['OCCUPATION_TYPE'] = encoder.
 ↪fit_transform(balance_data['OCCUPATION_TYPE'])
balance_data['ORGANIZATION_TYPE'] = encoder.
 ↪fit_transform(balance_data['ORGANIZATION_TYPE'])
balance_data['HOUSETYPE_MODE'] = encoder.
 ↪fit_transform(balance_data['HOUSETYPE_MODE'])
balance_data["NAME_CONTRACT_TYPE"]=encoder.
 ↪fit_transform(balance_data["NAME_CONTRACT_TYPE"])
```

```
balance_data['FLAG_OWN_REALTY']=encoder.
 ↪fit_transform(balance_data['FLAG_OWN_REALTY'])
balance_data['NAME_TYPE_SUITE']=encoder.
 ↪fit_transform(balance_data['NAME_TYPE_SUITE'])
balance_data['WEEKDAY_APPR_PROCESS_START']=encoder.
 ↪fit_transform(balance_data['WEEKDAY_APPR_PROCESS_START'])
balance_data['EMERGENCYSTATE_MODE']=encoder.
 ↪fit_transform(balance_data['EMERGENCYSTATE_MODE'])
```

[25]:
```
dictionary = {}
dictionary['categorical'] = balance_data.dtypes[balance_data.dtypes ==␣
 ↪'object'].index
dictionary
```

[25]:
```
{'categorical': Index([], dtype='object')}
```

[26]:
```
balance_data.head()
```

[26]:
```
        SK_ID_CURR  TARGET  NAME_CONTRACT_TYPE  CODE_GENDER  FLAG_OWN_CAR  \
247205      386051       1                   0            1             0
212658      346441       1                   0            0             0
111346      229187       1                   0            0             0
89315       203696       1                   0            0             0
259738      400581       1                   0            0             0

        FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  \
247205                0             1          180000.0   1473871.5
212658                1             1          180000.0    647046.0
111346                1             0          225000.0    481176.0
89315                 0             0          135000.0    646920.0
259738                1             0          180000.0    545040.0

        AMT_ANNUITY  …  FLAG_DOCUMENT_18  FLAG_DOCUMENT_19  \
247205      43222.5  …                 0                 0
212658      19048.5  …                 0                 0
111346      26100.0  …                 0                 0
89315       25195.5  …                 0                 0
259738      26640.0  …                 0                 0

        FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  AMT_REQ_CREDIT_BUREAU_HOUR  \
247205                 0                 0                         0.0
212658                 0                 0                         0.0
111346                 0                 0                         0.0
89315                  0                 0                         0.0
259738                 0                 0                         0.0

        AMT_REQ_CREDIT_BUREAU_DAY  AMT_REQ_CREDIT_BUREAU_WEEK  \
```

```
247205                    0.0                        0.0
212658                    0.0                        0.0
111346                    0.0                        0.0
89315                     0.0                        0.0
259738                    0.0                        0.0

        AMT_REQ_CREDIT_BUREAU_MON  AMT_REQ_CREDIT_BUREAU_QRT  \
247205                        0.0                        1.0
212658                        0.0                        0.0
111346                        0.0                        3.0
89315                         0.0                        0.0
259738                        0.0                        0.0

        AMT_REQ_CREDIT_BUREAU_YEAR
247205                         4.0
212658                         3.0
111346                         6.0
89315                          5.0
259738                         0.0

[5 rows x 82 columns]
```

[27]:
```
balance_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49650 entries, 247205 to 87599
Data columns (total 82 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   SK_ID_CURR                    49650 non-null  int64
 1   TARGET                        49650 non-null  int64
 2   NAME_CONTRACT_TYPE            49650 non-null  int64
 3   CODE_GENDER                   49650 non-null  int32
 4   FLAG_OWN_CAR                  49650 non-null  int32
 5   FLAG_OWN_REALTY               49650 non-null  int32
 6   CNT_CHILDREN                  49650 non-null  int64
 7   AMT_INCOME_TOTAL              49650 non-null  float64
 8   AMT_CREDIT                    49650 non-null  float64
 9   AMT_ANNUITY                   49649 non-null  float64
 10  AMT_GOODS_PRICE               49613 non-null  float64
 11  NAME_TYPE_SUITE               49650 non-null  int32
 12  NAME_INCOME_TYPE              49650 non-null  int32
 13  NAME_EDUCATION_TYPE           49650 non-null  int32
 14  NAME_FAMILY_STATUS            49650 non-null  int32
 15  NAME_HOUSING_TYPE             49650 non-null  int32
 16  REGION_POPULATION_RELATIVE    49650 non-null  float64
 17  DAYS_BIRTH                    49650 non-null  int64
```

```
18  DAYS_EMPLOYED                 49650 non-null  int64
19  DAYS_REGISTRATION             49650 non-null  float64
20  DAYS_ID_PUBLISH               49650 non-null  int64
21  FLAG_MOBIL                    49650 non-null  int64
22  FLAG_EMP_PHONE                49650 non-null  int64
23  FLAG_WORK_PHONE               49650 non-null  int64
24  FLAG_CONT_MOBILE              49650 non-null  int64
25  FLAG_PHONE                    49650 non-null  int64
26  FLAG_EMAIL                    49650 non-null  int64
27  OCCUPATION_TYPE               49650 non-null  int32
28  CNT_FAM_MEMBERS               49650 non-null  float64
29  REGION_RATING_CLIENT          49650 non-null  int64
30  REGION_RATING_CLIENT_W_CITY   49650 non-null  int64
31  WEEKDAY_APPR_PROCESS_START    49650 non-null  int32
32  HOUR_APPR_PROCESS_START       49650 non-null  int64
33  REG_REGION_NOT_LIVE_REGION    49650 non-null  int64
34  REG_REGION_NOT_WORK_REGION    49650 non-null  int64
35  LIVE_REGION_NOT_WORK_REGION   49650 non-null  int64
36  REG_CITY_NOT_LIVE_CITY        49650 non-null  int64
37  REG_CITY_NOT_WORK_CITY        49650 non-null  int64
38  LIVE_CITY_NOT_WORK_CITY       49650 non-null  int64
39  ORGANIZATION_TYPE             49650 non-null  int32
40  EXT_SOURCE_2                  49530 non-null  float64
41  EXT_SOURCE_3                  38995 non-null  float64
42  YEARS_BEGINEXPLUATATION_AVG   23980 non-null  float64
43  FLOORSMAX_AVG                 23470 non-null  float64
44  YEARS_BEGINEXPLUATATION_MODE  23980 non-null  float64
45  FLOORSMAX_MODE                23470 non-null  float64
46  YEARS_BEGINEXPLUATATION_MEDI  23980 non-null  float64
47  FLOORSMAX_MEDI                23470 non-null  float64
48  HOUSETYPE_MODE                49650 non-null  int32
49  TOTALAREA_MODE                24190 non-null  float64
50  EMERGENCYSTATE_MODE           49650 non-null  int32
51  OBS_30_CNT_SOCIAL_CIRCLE      49531 non-null  float64
52  DEF_30_CNT_SOCIAL_CIRCLE      49531 non-null  float64
53  OBS_60_CNT_SOCIAL_CIRCLE      49531 non-null  float64
54  DEF_60_CNT_SOCIAL_CIRCLE      49531 non-null  float64
55  DAYS_LAST_PHONE_CHANGE        49649 non-null  float64
56  FLAG_DOCUMENT_2               49650 non-null  int64
57  FLAG_DOCUMENT_3               49650 non-null  int64
58  FLAG_DOCUMENT_4               49650 non-null  int64
59  FLAG_DOCUMENT_5               49650 non-null  int64
60  FLAG_DOCUMENT_6               49650 non-null  int64
61  FLAG_DOCUMENT_7               49650 non-null  int64
62  FLAG_DOCUMENT_8               49650 non-null  int64
63  FLAG_DOCUMENT_9               49650 non-null  int64
64  FLAG_DOCUMENT_10              49650 non-null  int64
65  FLAG_DOCUMENT_11              49650 non-null  int64
```

```
 66  FLAG_DOCUMENT_12              49650 non-null  int64
 67  FLAG_DOCUMENT_13              49650 non-null  int64
 68  FLAG_DOCUMENT_14              49650 non-null  int64
 69  FLAG_DOCUMENT_15              49650 non-null  int64
 70  FLAG_DOCUMENT_16              49650 non-null  int64
 71  FLAG_DOCUMENT_17              49650 non-null  int64
 72  FLAG_DOCUMENT_18              49650 non-null  int64
 73  FLAG_DOCUMENT_19              49650 non-null  int64
 74  FLAG_DOCUMENT_20              49650 non-null  int64
 75  FLAG_DOCUMENT_21              49650 non-null  int64
 76  AMT_REQ_CREDIT_BUREAU_HOUR    42006 non-null  float64
 77  AMT_REQ_CREDIT_BUREAU_DAY     42006 non-null  float64
 78  AMT_REQ_CREDIT_BUREAU_WEEK    42006 non-null  float64
 79  AMT_REQ_CREDIT_BUREAU_MON     42006 non-null  float64
 80  AMT_REQ_CREDIT_BUREAU_QRT     42006 non-null  float64
 81  AMT_REQ_CREDIT_BUREAU_YEAR    42006 non-null  float64
dtypes: float64(27), int32(13), int64(42)
memory usage: 30.0 MB
```

[28]:
```
float_col = balance_data.select_dtypes('float').columns
int_col = balance_data.select_dtypes('int64').columns
column_process = int_col.append(float_col)
```

[29]:
```
len(column_process)
```

[29]:
```
69
```

[30]:
```
column_process
```

[30]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CNT_CHILDREN',
       'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE',
       'FLAG_EMAIL', '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', '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_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
       'REGION_POPULATION_RELATIVE', 'DAYS_REGISTRATION', 'CNT_FAM_MEMBERS',
       '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', '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')
```

[31]:
```
# Find the Nan or Missing value and treatment on it by using simple imputer.

imputer = SimpleImputer(missing_values=np.nan,strategy='mean')
balance_data[column_process] = imputer.
↪fit_transform(balance_data[column_process])
```

[32]:
```
# after treatment of find the missing value is available or not

balance_data.isnull().sum().sum()
```

[32]:
```
0
```

[33]:
```
# after treatment of find the missing value is available or not

null_counts = balance_data.isna().sum()/balance_data.shape[0]*100
null_counts.sort_values(ascending=False,inplace=True)
null_counts.head(40)
```

[33]:
```
SK_ID_CURR                      0.0
FLAG_DOCUMENT_7                 0.0
FLAG_DOCUMENT_5                 0.0
FLAG_DOCUMENT_4                 0.0
FLAG_DOCUMENT_3                 0.0
FLAG_DOCUMENT_2                 0.0
DAYS_LAST_PHONE_CHANGE          0.0
DEF_60_CNT_SOCIAL_CIRCLE        0.0
OBS_60_CNT_SOCIAL_CIRCLE        0.0
DEF_30_CNT_SOCIAL_CIRCLE        0.0
OBS_30_CNT_SOCIAL_CIRCLE        0.0
EMERGENCYSTATE_MODE             0.0
TOTALAREA_MODE                  0.0
HOUSETYPE_MODE                  0.0
FLOORSMAX_MEDI                  0.0
YEARS_BEGINEXPLUATATION_MEDI    0.0
FLOORSMAX_MODE                  0.0
YEARS_BEGINEXPLUATATION_MODE    0.0
FLOORSMAX_AVG                   0.0
FLAG_DOCUMENT_6                 0.0
```

```
FLAG_DOCUMENT_8                  0.0
TARGET                           0.0
FLAG_DOCUMENT_9                  0.0
AMT_REQ_CREDIT_BUREAU_QRT        0.0
AMT_REQ_CREDIT_BUREAU_MON        0.0
AMT_REQ_CREDIT_BUREAU_WEEK       0.0
AMT_REQ_CREDIT_BUREAU_DAY        0.0
AMT_REQ_CREDIT_BUREAU_HOUR       0.0
FLAG_DOCUMENT_21                 0.0
FLAG_DOCUMENT_20                 0.0
FLAG_DOCUMENT_19                 0.0
FLAG_DOCUMENT_18                 0.0
FLAG_DOCUMENT_17                 0.0
FLAG_DOCUMENT_16                 0.0
FLAG_DOCUMENT_15                 0.0
FLAG_DOCUMENT_14                 0.0
FLAG_DOCUMENT_13                 0.0
FLAG_DOCUMENT_12                 0.0
FLAG_DOCUMENT_11                 0.0
FLAG_DOCUMENT_10                 0.0
dtype: float64
```

[34]:
```
# Split the dataset for model building

X = balance_data.drop(['TARGET'],axis=1)
y = balance_data[['TARGET']]
```

[35]:
```
# all variable values convert in 0 to 1 form by using standard scaler.

scaler = StandardScaler()
X = scaler.fit_transform(X)
```

[36]:
```
xtrain,xtest,ytrain,ytest = train_test_split(X,y,random_state=25)
```

[37]:
```
print(xtrain.shape)
print(xtest.shape)
print(ytrain.shape)
print(ytest.shape)
```

```
(37237, 81)
(12413, 81)
(37237, 1)
(12413, 1)
```

[38]:
```
# Build the model

model = Sequential()
```

```
model.add(Dense(256,activation='relu',input_shape=(xtrain.shape[1],)))
model.add(Dropout(0.5))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid'))
```

[39]:
```
# Find the summary of model

model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               20992

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 64)                8256

 dropout_2 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 62,209
Trainable params: 62,209
Non-trainable params: 0
_________________________________________________________________
```

[40]:
```
# Compile the model

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

[41]:
```
# Fit the model

model.fit(xtrain,ytrain,epochs=100,batch_size=256,validation_data=(xtest,ytest))
```

```
Epoch 1/100
```

```
146/146 [==============================] - 8s 22ms/step - loss: 0.6813 -
accuracy: 0.5931 - val_loss: 0.6266 - val_accuracy: 0.6750
Epoch 2/100
146/146 [==============================] - 2s 16ms/step - loss: 0.6347 -
accuracy: 0.6492 - val_loss: 0.6196 - val_accuracy: 0.6770
Epoch 3/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6240 -
accuracy: 0.6631 - val_loss: 0.6121 - val_accuracy: 0.6785
Epoch 4/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6197 -
accuracy: 0.6641 - val_loss: 0.6082 - val_accuracy: 0.6799
Epoch 5/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6161 -
accuracy: 0.6706 - val_loss: 0.6133 - val_accuracy: 0.6812
Epoch 6/100
146/146 [==============================] - 2s 13ms/step - loss: 0.6126 -
accuracy: 0.6735 - val_loss: 0.6101 - val_accuracy: 0.6793
Epoch 7/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6113 -
accuracy: 0.6720 - val_loss: 0.6059 - val_accuracy: 0.6832
Epoch 8/100
146/146 [==============================] - 2s 15ms/step - loss: 0.6093 -
accuracy: 0.6774 - val_loss: 0.6045 - val_accuracy: 0.6821
Epoch 9/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6090 -
accuracy: 0.6769 - val_loss: 0.6097 - val_accuracy: 0.6823
Epoch 10/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6085 -
accuracy: 0.6756 - val_loss: 0.6064 - val_accuracy: 0.6838
Epoch 11/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6067 -
accuracy: 0.6784 - val_loss: 0.6034 - val_accuracy: 0.6819
Epoch 12/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6037 -
accuracy: 0.6786 - val_loss: 0.6030 - val_accuracy: 0.6807
Epoch 13/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6043 -
accuracy: 0.6813 - val_loss: 0.6032 - val_accuracy: 0.6822
Epoch 14/100
146/146 [==============================] - 2s 13ms/step - loss: 0.6040 -
accuracy: 0.6797 - val_loss: 0.6022 - val_accuracy: 0.6835
Epoch 15/100
146/146 [==============================] - 2s 14ms/step - loss: 0.6032 -
accuracy: 0.6802 - val_loss: 0.6047 - val_accuracy: 0.6813
Epoch 16/100
146/146 [==============================] - 2s 15ms/step - loss: 0.6016 -
accuracy: 0.6835 - val_loss: 0.6048 - val_accuracy: 0.6844
Epoch 17/100
```

```
146/146 [==============================] - 2s 14ms/step - loss: 0.6011 -
accuracy: 0.6820 - val_loss: 0.5998 - val_accuracy: 0.6832
Epoch 18/100
146/146 [==============================] - 2s 15ms/step - loss: 0.6013 -
accuracy: 0.6831 - val_loss: 0.6002 - val_accuracy: 0.6832
Epoch 19/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5995 -
accuracy: 0.6827 - val_loss: 0.6035 - val_accuracy: 0.6845
Epoch 20/100
146/146 [==============================] - 2s 15ms/step - loss: 0.5992 -
accuracy: 0.6832 - val_loss: 0.6025 - val_accuracy: 0.6803
Epoch 21/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5970 -
accuracy: 0.6841 - val_loss: 0.6006 - val_accuracy: 0.6832
Epoch 22/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5987 -
accuracy: 0.6849 - val_loss: 0.6041 - val_accuracy: 0.6831
Epoch 23/100
146/146 [==============================] - 2s 15ms/step - loss: 0.5967 -
accuracy: 0.6852 - val_loss: 0.6014 - val_accuracy: 0.6828
Epoch 24/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5972 -
accuracy: 0.6829 - val_loss: 0.6019 - val_accuracy: 0.6815
Epoch 25/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5939 -
accuracy: 0.6845 - val_loss: 0.6014 - val_accuracy: 0.6807
Epoch 26/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5947 -
accuracy: 0.6868 - val_loss: 0.6015 - val_accuracy: 0.6820
Epoch 27/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5917 -
accuracy: 0.6870 - val_loss: 0.6006 - val_accuracy: 0.6821
Epoch 28/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5916 -
accuracy: 0.6870 - val_loss: 0.6034 - val_accuracy: 0.6843
Epoch 29/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5916 -
accuracy: 0.6881 - val_loss: 0.6034 - val_accuracy: 0.6807
Epoch 30/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5903 -
accuracy: 0.6876 - val_loss: 0.5976 - val_accuracy: 0.6814
Epoch 31/100
146/146 [==============================] - 2s 15ms/step - loss: 0.5912 -
accuracy: 0.6860 - val_loss: 0.5991 - val_accuracy: 0.6826
Epoch 32/100
146/146 [==============================] - 2s 15ms/step - loss: 0.5897 -
accuracy: 0.6877 - val_loss: 0.5993 - val_accuracy: 0.6820
Epoch 33/100
```

```
146/146 [==============================] - 2s 15ms/step - loss: 0.5884 -
accuracy: 0.6896 - val_loss: 0.5985 - val_accuracy: 0.6828
Epoch 34/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5863 -
accuracy: 0.6900 - val_loss: 0.6016 - val_accuracy: 0.6813
Epoch 35/100
146/146 [==============================] - 2s 16ms/step - loss: 0.5875 -
accuracy: 0.6914 - val_loss: 0.6026 - val_accuracy: 0.6797
Epoch 36/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5869 -
accuracy: 0.6878 - val_loss: 0.5993 - val_accuracy: 0.6803
Epoch 37/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5852 -
accuracy: 0.6912 - val_loss: 0.6002 - val_accuracy: 0.6786
Epoch 38/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5858 -
accuracy: 0.6885 - val_loss: 0.6006 - val_accuracy: 0.6782
Epoch 39/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5854 -
accuracy: 0.6918 - val_loss: 0.6019 - val_accuracy: 0.6767
Epoch 40/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5857 -
accuracy: 0.6904 - val_loss: 0.6019 - val_accuracy: 0.6789
Epoch 41/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5823 -
accuracy: 0.6909 - val_loss: 0.6007 - val_accuracy: 0.6809
Epoch 42/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5833 -
accuracy: 0.6898 - val_loss: 0.5999 - val_accuracy: 0.6805
Epoch 43/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5825 -
accuracy: 0.6889 - val_loss: 0.6001 - val_accuracy: 0.6795
Epoch 44/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5827 -
accuracy: 0.6919 - val_loss: 0.6007 - val_accuracy: 0.6791
Epoch 45/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5817 -
accuracy: 0.6910 - val_loss: 0.6015 - val_accuracy: 0.6813
Epoch 46/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5817 -
accuracy: 0.6926 - val_loss: 0.6017 - val_accuracy: 0.6763
Epoch 47/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5814 -
accuracy: 0.6934 - val_loss: 0.5998 - val_accuracy: 0.6810
Epoch 48/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5787 -
accuracy: 0.6937 - val_loss: 0.6014 - val_accuracy: 0.6791
Epoch 49/100
```

```
146/146 [==============================] - 2s 15ms/step - loss: 0.5777 -
accuracy: 0.6951 - val_loss: 0.6018 - val_accuracy: 0.6794
Epoch 50/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5795 -
accuracy: 0.6925 - val_loss: 0.6031 - val_accuracy: 0.6808
Epoch 51/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5772 -
accuracy: 0.6922 - val_loss: 0.6017 - val_accuracy: 0.6811
Epoch 52/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5767 -
accuracy: 0.6957 - val_loss: 0.6030 - val_accuracy: 0.6790
Epoch 53/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5754 -
accuracy: 0.6945 - val_loss: 0.6024 - val_accuracy: 0.6771
Epoch 54/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5776 -
accuracy: 0.6941 - val_loss: 0.6051 - val_accuracy: 0.6768
Epoch 55/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5756 -
accuracy: 0.6962 - val_loss: 0.6038 - val_accuracy: 0.6769
Epoch 56/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5751 -
accuracy: 0.6950 - val_loss: 0.6052 - val_accuracy: 0.6780
Epoch 57/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5739 -
accuracy: 0.6956 - val_loss: 0.6045 - val_accuracy: 0.6782
Epoch 58/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5755 -
accuracy: 0.6965 - val_loss: 0.6027 - val_accuracy: 0.6807
Epoch 59/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5734 -
accuracy: 0.6944 - val_loss: 0.6049 - val_accuracy: 0.6794
Epoch 60/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5725 -
accuracy: 0.6973 - val_loss: 0.6036 - val_accuracy: 0.6785
Epoch 61/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5728 -
accuracy: 0.6966 - val_loss: 0.6055 - val_accuracy: 0.6792
Epoch 62/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5715 -
accuracy: 0.6971 - val_loss: 0.6072 - val_accuracy: 0.6782
Epoch 63/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5724 -
accuracy: 0.6984 - val_loss: 0.6049 - val_accuracy: 0.6790
Epoch 64/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5705 -
accuracy: 0.6979 - val_loss: 0.6078 - val_accuracy: 0.6741
Epoch 65/100
```

```
146/146 [==============================] - 2s 14ms/step - loss: 0.5698 -
accuracy: 0.6960 - val_loss: 0.6048 - val_accuracy: 0.6790
Epoch 66/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5719 -
accuracy: 0.6965 - val_loss: 0.6076 - val_accuracy: 0.6758
Epoch 67/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5719 -
accuracy: 0.6976 - val_loss: 0.6071 - val_accuracy: 0.6760
Epoch 68/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5697 -
accuracy: 0.6974 - val_loss: 0.6072 - val_accuracy: 0.6772
Epoch 69/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5692 -
accuracy: 0.6990 - val_loss: 0.6060 - val_accuracy: 0.6770
Epoch 70/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5687 -
accuracy: 0.7000 - val_loss: 0.6057 - val_accuracy: 0.6745
Epoch 71/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5674 -
accuracy: 0.6991 - val_loss: 0.6068 - val_accuracy: 0.6749
Epoch 72/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5673 -
accuracy: 0.6991 - val_loss: 0.6075 - val_accuracy: 0.6750
Epoch 73/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5675 -
accuracy: 0.7019 - val_loss: 0.6073 - val_accuracy: 0.6745
Epoch 74/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5673 -
accuracy: 0.6993 - val_loss: 0.6075 - val_accuracy: 0.6752
Epoch 75/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5654 -
accuracy: 0.7024 - val_loss: 0.6068 - val_accuracy: 0.6770
Epoch 76/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5653 -
accuracy: 0.7029 - val_loss: 0.6076 - val_accuracy: 0.6751
Epoch 77/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5638 -
accuracy: 0.7035 - val_loss: 0.6082 - val_accuracy: 0.6749
Epoch 78/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5642 -
accuracy: 0.7011 - val_loss: 0.6082 - val_accuracy: 0.6777
Epoch 79/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5640 -
accuracy: 0.6996 - val_loss: 0.6071 - val_accuracy: 0.6772
Epoch 80/100
146/146 [==============================] - 2s 15ms/step - loss: 0.5643 -
accuracy: 0.6989 - val_loss: 0.6083 - val_accuracy: 0.6771
Epoch 81/100
```

```
146/146 [==============================] - 2s 14ms/step - loss: 0.5632 -
accuracy: 0.7024 - val_loss: 0.6077 - val_accuracy: 0.6764
Epoch 82/100
146/146 [==============================] - 2s 15ms/step - loss: 0.5625 -
accuracy: 0.7017 - val_loss: 0.6090 - val_accuracy: 0.6750
Epoch 83/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5609 -
accuracy: 0.7025 - val_loss: 0.6105 - val_accuracy: 0.6719
Epoch 84/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5637 -
accuracy: 0.7023 - val_loss: 0.6108 - val_accuracy: 0.6746
Epoch 85/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5599 -
accuracy: 0.7014 - val_loss: 0.6106 - val_accuracy: 0.6744
Epoch 86/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5610 -
accuracy: 0.7035 - val_loss: 0.6107 - val_accuracy: 0.6708
Epoch 87/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5616 -
accuracy: 0.7022 - val_loss: 0.6087 - val_accuracy: 0.6729
Epoch 88/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5613 -
accuracy: 0.7033 - val_loss: 0.6093 - val_accuracy: 0.6740
Epoch 89/100
146/146 [==============================] - 2s 15ms/step - loss: 0.5592 -
accuracy: 0.7038 - val_loss: 0.6104 - val_accuracy: 0.6732
Epoch 90/100
146/146 [==============================] - 3s 17ms/step - loss: 0.5588 -
accuracy: 0.7049 - val_loss: 0.6103 - val_accuracy: 0.6719
Epoch 91/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5586 -
accuracy: 0.7060 - val_loss: 0.6106 - val_accuracy: 0.6721
Epoch 92/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5582 -
accuracy: 0.7051 - val_loss: 0.6101 - val_accuracy: 0.6761
Epoch 93/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5595 -
accuracy: 0.7016 - val_loss: 0.6108 - val_accuracy: 0.6736
Epoch 94/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5568 -
accuracy: 0.7058 - val_loss: 0.6106 - val_accuracy: 0.6758
Epoch 95/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5559 -
accuracy: 0.7063 - val_loss: 0.6115 - val_accuracy: 0.6758
Epoch 96/100
146/146 [==============================] - 2s 15ms/step - loss: 0.5547 -
accuracy: 0.7080 - val_loss: 0.6113 - val_accuracy: 0.6727
Epoch 97/100
```

```
146/146 [==============================] - 2s 14ms/step - loss: 0.5548 -
accuracy: 0.7062 - val_loss: 0.6124 - val_accuracy: 0.6728
Epoch 98/100
146/146 [==============================] - 2s 14ms/step - loss: 0.5571 -
accuracy: 0.7055 - val_loss: 0.6104 - val_accuracy: 0.6748
Epoch 99/100
146/146 [==============================] - 2s 13ms/step - loss: 0.5581 -
accuracy: 0.7020 - val_loss: 0.6122 - val_accuracy: 0.6718
Epoch 100/100
146/146 [==============================] - 2s 17ms/step - loss: 0.5551 -
accuracy: 0.7046 - val_loss: 0.6108 - val_accuracy: 0.6722
```

[41]:
```
<keras.callbacks.History at 0x1f70f472b80>
```

[42]:
```python
prediction = (model.predict(xtest)>0.5)*1.0
prediction
```

```
388/388 [==============================] - 1s 3ms/step
```

[42]:
```
array([[1.],
       [0.],
       [0.],
       …,
       [0.],
       [1.],
       [1.]])
```

[43]:
```python
print('Confusion Matrix : ')
print(confusion_matrix(prediction,ytest))
```

```
Confusion Matrix :
[[3898 1780]
 [2289 4446]]
```

[44]:
```python
print('Classification Report : ')
print(classification_report(prediction,ytest))
```

```
Classification Report :
              precision    recall  f1-score   support

         0.0       0.63      0.69      0.66      5678
         1.0       0.71      0.66      0.69      6735

    accuracy                           0.67     12413
   macro avg       0.67      0.67      0.67     12413
weighted avg       0.68      0.67      0.67     12413
```

[45]:
```
test_loss,test_accuracy = model.evaluate(xtest,ytest)
print('Test Loss: ',test_loss)
print('Test Accuracy: ',test_accuracy)
```

```
388/388 [==============================] - 1s 4ms/step - loss: 0.6108 -
accuracy: 0.6722
Test Loss:  0.610781192779541
Test Accuracy:  0.672198474407196
```

1.0.7 7. Calculate Sensitivity as a metrice

Sensitivity=TP/(FN+TP)

Specificity=TN/(FP+TN)

[48]:
```
print('Sensitivity of the dataset is: ',3898/(1780+3898))
```

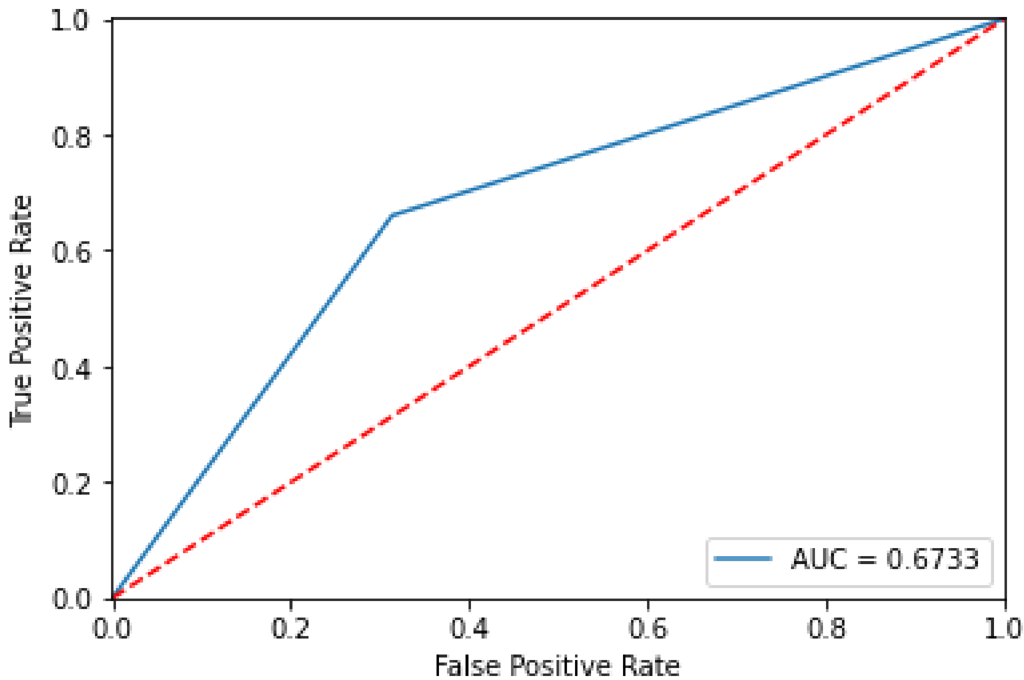
```
Sensitivity of the dataset is:  0.6865093342726312
```

1.0.8 8. Calculate area under receiver operating characteristics curve

[47]:
```
fpr,tpr,threshold = roc_curve(prediction,ytest)
roc_auc = auc(fpr,tpr)
print('False Postive Rate :',fpr)
print('True Positive Rate :',tpr)
print('Threshold values :',threshold)

plt.plot(fpr,tpr,label='AUC = %0.4f'% roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.001, 1])
plt.ylim([0, 1.001])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc='lower right')
plt.show()
```

```
False Postive Rate : [0.         0.31349067 1.        ]
True Positive Rate : [0.         0.66013363 1.        ]
Threshold values : [2. 1. 0.]
```


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
AUC = 0.6733
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

[ ]: