Project1

August 5, 2022

[1]:
```python
# Importing the libraries
import pandas as pd
import numpy as np
import tensorflow as tf
```

[2]:
```python
# Importing the data
data = pd.read_csv('loan_data (1).csv')
```

[3]:
```python
data.shape
```

[3]:
```
(307511, 122)
```

[4]:
```python
data.head()
```

[4]:
```
   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]
```

[5]:
```
#Check for null values in the dataset
data.isnull().sum().sort_values(ascending=False)
```

[5]:
```
COMMONAREA_MEDI              214865
COMMONAREA_AVG               214865
COMMONAREA_MODE              214865
NONLIVINGAPARTMENTS_MODE     213514
NONLIVINGAPARTMENTS_MEDI     213514
                              …
REG_CITY_NOT_LIVE_CITY            0
LIVE_REGION_NOT_WORK_REGION       0
REG_REGION_NOT_WORK_REGION        0
HOUR_APPR_PROCESS_START           0
SK_ID_CURR                        0
Length: 122, dtype: int64
```

[6]:
```
# WE can remove the columns which have more than 50% of missing values
perc = 50.0
min_count =  int(((100-perc)/100)*data.shape[0] + 1)
mod_df = data.dropna( axis=1,
                thresh=min_count)
```

[7]:
```
mod_df.shape
```

[7]:
```
(307511, 81)
```

[8]:
```
# Imputing the missing values
```

[9]:
```
mod_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 81 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  FLAG_MOBIL                    307511 non-null  int64
 22  FLAG_EMP_PHONE                307511 non-null  int64
 23  FLAG_WORK_PHONE               307511 non-null  int64
 24  FLAG_CONT_MOBILE              307511 non-null  int64
 25  FLAG_PHONE                    307511 non-null  int64
 26  FLAG_EMAIL                    307511 non-null  int64
 27  OCCUPATION_TYPE               211120 non-null  object
 28  CNT_FAM_MEMBERS               307509 non-null  float64
 29  REGION_RATING_CLIENT          307511 non-null  int64
 30  REGION_RATING_CLIENT_W_CITY   307511 non-null  int64
 31  WEEKDAY_APPR_PROCESS_START    307511 non-null  object
 32  HOUR_APPR_PROCESS_START       307511 non-null  int64
 33  REG_REGION_NOT_LIVE_REGION    307511 non-null  int64
 34  REG_REGION_NOT_WORK_REGION    307511 non-null  int64
 35  LIVE_REGION_NOT_WORK_REGION   307511 non-null  int64
 36  REG_CITY_NOT_LIVE_CITY        307511 non-null  int64
 37  REG_CITY_NOT_WORK_CITY        307511 non-null  int64
```

```
 38  LIVE_CITY_NOT_WORK_CITY       307511 non-null  int64
 39  ORGANIZATION_TYPE             307511 non-null  object
 40  EXT_SOURCE_2                  306851 non-null  float64
 41  EXT_SOURCE_3                  246546 non-null  float64
 42  YEARS_BEGINEXPLUATATION_AVG   157504 non-null  float64
 43  FLOORSMAX_AVG                 154491 non-null  float64
 44  YEARS_BEGINEXPLUATATION_MODE  157504 non-null  float64
 45  FLOORSMAX_MODE                154491 non-null  float64
 46  YEARS_BEGINEXPLUATATION_MEDI  157504 non-null  float64
 47  FLOORSMAX_MEDI                154491 non-null  float64
 48  TOTALAREA_MODE                159080 non-null  float64
 49  EMERGENCYSTATE_MODE           161756 non-null  object
 50  OBS_30_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 51  DEF_30_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 52  OBS_60_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 53  DEF_60_CNT_SOCIAL_CIRCLE      306490 non-null  float64
 54  DAYS_LAST_PHONE_CHANGE        307510 non-null  float64
 55  FLAG_DOCUMENT_2               307511 non-null  int64
 56  FLAG_DOCUMENT_3               307511 non-null  int64
 57  FLAG_DOCUMENT_4               307511 non-null  int64
 58  FLAG_DOCUMENT_5               307511 non-null  int64
 59  FLAG_DOCUMENT_6               307511 non-null  int64
 60  FLAG_DOCUMENT_7               307511 non-null  int64
 61  FLAG_DOCUMENT_8               307511 non-null  int64
 62  FLAG_DOCUMENT_9               307511 non-null  int64
 63  FLAG_DOCUMENT_10              307511 non-null  int64
 64  FLAG_DOCUMENT_11              307511 non-null  int64
 65  FLAG_DOCUMENT_12              307511 non-null  int64
 66  FLAG_DOCUMENT_13              307511 non-null  int64
 67  FLAG_DOCUMENT_14              307511 non-null  int64
 68  FLAG_DOCUMENT_15              307511 non-null  int64
 69  FLAG_DOCUMENT_16              307511 non-null  int64
 70  FLAG_DOCUMENT_17              307511 non-null  int64
 71  FLAG_DOCUMENT_18              307511 non-null  int64
 72  FLAG_DOCUMENT_19              307511 non-null  int64
 73  FLAG_DOCUMENT_20              307511 non-null  int64
 74  FLAG_DOCUMENT_21              307511 non-null  int64
 75  AMT_REQ_CREDIT_BUREAU_HOUR    265992 non-null  float64
 76  AMT_REQ_CREDIT_BUREAU_DAY     265992 non-null  float64
 77  AMT_REQ_CREDIT_BUREAU_WEEK    265992 non-null  float64
 78  AMT_REQ_CREDIT_BUREAU_MON     265992 non-null  float64
 79  AMT_REQ_CREDIT_BUREAU_QRT     265992 non-null  float64
 80  AMT_REQ_CREDIT_BUREAU_YEAR    265992 non-null  float64
dtypes: float64(27), int64(41), object(13)
memory usage: 190.0+ MB
```

[10]:

```
mod_df.OCCUPATION_TYPE.value_counts()
```

[10]:
```
Laborers                 55186
Sales staff              32102
Core staff               27570
Managers                 21371
Drivers                  18603
High skill tech staff    11380
Accountants               9813
Medicine staff            8537
Security staff            6721
Cooking staff             5946
Cleaning staff            4653
Private service staff     2652
Low-skill Laborers        2093
Waiters/barmen staff      1348
Secretaries               1305
Realty agents              751
HR staff                   563
IT staff                   526
Name: OCCUPATION_TYPE, dtype: int64
```

[11]:
```python
mod_df.OCCUPATION_TYPE.isnull().sum()
```

[11]:
```
96391
```

[12]:
```python
mod_df.OCCUPATION_TYPE = mod_df.OCCUPATION_TYPE.fillna('Missing')
```

```
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5170:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
```

[13]:
```python
pd.set_option('display.float_format',lambda x: '%2f' %x)
```

[14]:
```python
mod_df.EXT_SOURCE_3.describe()
```

[14]:
```
count   246546.000000
mean         0.510853
std          0.194844
min          0.000527
25%          0.370650
50%          0.535276
75%          0.669057
max          0.896010
```

```
Name: EXT_SOURCE_3, dtype: float64
```

[15]:
```
mod_df.EXT_SOURCE_2 = mod_df.EXT_SOURCE_2.fillna(mod_df.EXT_SOURCE_2.mean())
```

[16]:
```
mod_df.YEARS_BEGINEXPLUATATION_AVG.describe()
```

[16]:
```
count    157504.000000
mean          0.977735
std           0.059223
min           0.000000
25%           0.976700
50%           0.981600
75%           0.986600
max           1.000000
Name: YEARS_BEGINEXPLUATATION_AVG, dtype: float64
```

[17]:
```
mod_df.YEARS_BEGINEXPLUATATION_AVG = mod_df.YEARS_BEGINEXPLUATATION_AVG.
 ↪fillna(mod_df.YEARS_BEGINEXPLUATATION_AVG.mean())
```

[18]:
```
mod_df.FLOORSMAX_AVG.describe()
```

[18]:
```
count    154491.000000
mean          0.226282
std           0.144641
min           0.000000
25%           0.166700
50%           0.166700
75%           0.333300
max           1.000000
Name: FLOORSMAX_AVG, dtype: float64
```

[19]:
```
mod_df.FLOORSMAX_AVG = mod_df.FLOORSMAX_AVG.fillna(mod_df.FLOORSMAX_AVG.mean())
```

[20]:
```
mod_df.YEARS_BEGINEXPLUATATION_MODE.describe()
```

[20]:
```
count    157504.000000
mean          0.977065
std           0.064575
min           0.000000
25%           0.976700
50%           0.981600
75%           0.986600
max           1.000000
Name: YEARS_BEGINEXPLUATATION_MODE, dtype: float64
```

[21]:
```
mod_df.YEARS_BEGINEXPLUATATION_MODE = mod_df.YEARS_BEGINEXPLUATATION_MODE.
 ↪fillna(mod_df.YEARS_BEGINEXPLUATATION_MODE.mean())
```

[22]:
```
mod_df.FLOORSMAX_MODE.describe()
```

[22]:
```
count    154491.000000
mean          0.222315
std           0.143709
min           0.000000
25%           0.166700
50%           0.166700
75%           0.333300
max           1.000000
Name: FLOORSMAX_MODE, dtype: float64
```

[23]:
```
mod_df.FLOORSMAX_MODE =mod_df.FLOORSMAX_MODE.fillna(mod_df.FLOORSMAX_MODE.
 ↪mean())
```

[24]:
```
mod_df.YEARS_BEGINEXPLUATATION_MEDI.describe()
```

[24]:
```
count    157504.000000
mean          0.977752
std           0.059897
min           0.000000
25%           0.976700
50%           0.981600
75%           0.986600
max           1.000000
Name: YEARS_BEGINEXPLUATATION_MEDI, dtype: float64
```

[25]:
```
mod_df.YEARS_BEGINEXPLUATATION_MEDI = mod_df.YEARS_BEGINEXPLUATATION_MEDI.
 ↪fillna(mod_df.YEARS_BEGINEXPLUATATION_MEDI.median())
```

[26]:
```
mod_df.FLOORSMAX_MEDI.describe()
```

[26]:
```
count    154491.000000
mean          0.225897
std           0.145067
min           0.000000
25%           0.166700
50%           0.166700
75%           0.333300
max           1.000000
Name: FLOORSMAX_MEDI, dtype: float64
```

[27]:
```
mod_df.FLOORSMAX_MEDI = mod_df.FLOORSMAX_MEDI.fillna(mod_df.FLOORSMAX_MEDI.
 ↪median())
```

[28]:
```
mod_df.TOTALAREA_MODE.describe()
```

[28]:
```
count    159080.000000
mean          0.102547
std           0.107462
min           0.000000
25%           0.041200
50%           0.068800
75%           0.127600
max           1.000000
Name: TOTALAREA_MODE, dtype: float64
```

[29]:
```
mod_df.TOTALAREA_MODE = mod_df.TOTALAREA_MODE.fillna(mod_df.TOTALAREA_MODE.
 ↪median())
```

[30]:
```
mod_df.EMERGENCYSTATE_MODE.value_counts(dropna=False)
```

[30]:
```
No     159428
NaN    145755
Yes      2328
Name: EMERGENCYSTATE_MODE, dtype: int64
```

[31]:
```
mod_df.EMERGENCYSTATE_MODE = mod_df.EMERGENCYSTATE_MODE.fillna('Not known')
```

[32]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_HOUR.describe()
```

[32]:
```
count    265992.000000
mean          0.006402
std           0.083849
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           4.000000
Name: AMT_REQ_CREDIT_BUREAU_HOUR, dtype: float64
```

[33]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_HOUR.value_counts(dropna=False)
```

[33]:
```
0.000000    264366
nan          41519
1.000000      1560
2.000000        56
3.000000         9
4.000000         1
Name: AMT_REQ_CREDIT_BUREAU_HOUR, dtype: int64
```

[34]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_HOUR = mod_df.AMT_REQ_CREDIT_BUREAU_HOUR.fillna(0.
 ↪0)
```

[35]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_DAY.value_counts(dropna=False)
```

[35]:
```
0.000000    264503
nan          41519
1.000000      1292
2.000000       106
3.000000        45
4.000000        26
5.000000         9
6.000000         8
9.000000         2
8.000000         1
Name: AMT_REQ_CREDIT_BUREAU_DAY, dtype: int64
```

[36]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_DAY = mod_df.AMT_REQ_CREDIT_BUREAU_DAY.fillna(0.0)
```

[37]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_WEEK.value_counts(dropna=False)
```

[37]:
```
0.000000    257456
nan          41519
1.000000      8208
2.000000       199
3.000000        58
4.000000        34
6.000000        20
5.000000        10
8.000000         5
7.000000         2
Name: AMT_REQ_CREDIT_BUREAU_WEEK, dtype: int64
```

[38]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_WEEK = mod_df.AMT_REQ_CREDIT_BUREAU_WEEK.fillna(0.
 ↪0)
```

[39]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_MON.value_counts(dropna=False)
```

[39]:
```
0.000000     222233
nan           41519
1.000000      33147
2.000000       5386
3.000000       1991
4.000000       1076
5.000000        602
6.000000        343
7.000000        298
9.000000        206
8.000000        185
10.000000       132
```

```
11.000000       119
12.000000        77
13.000000        72
14.000000        40
15.000000        35
16.000000        23
17.000000        14
18.000000         6
19.000000         3
23.000000         1
27.000000         1
22.000000         1
24.000000         1
Name: AMT_REQ_CREDIT_BUREAU_MON, dtype: int64
```

[40]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_MON = mod_df.AMT_REQ_CREDIT_BUREAU_MON.fillna(0.0)
```

[41]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_QRT.value_counts(dropna=False)
```

[41]:
```
0.000000      215417
nan            41519
1.000000       33862
2.000000       14412
3.000000        1717
4.000000         476
5.000000          64
6.000000          28
7.000000           7
8.000000           7
19.000000          1
261.000000         1
Name: AMT_REQ_CREDIT_BUREAU_QRT, dtype: int64
```

[42]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_QRT = mod_df.AMT_REQ_CREDIT_BUREAU_QRT.fillna(0.0)
```

[43]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_YEAR.value_counts(dropna=False)
```

[43]:
```
0.000000     71801
1.000000     63405
2.000000     50192
nan          41519
3.000000     33628
4.000000     20714
5.000000     12052
6.000000      6967
7.000000      3869
8.000000      2127
```

```
9.000000      1096
11.000000       31
12.000000       30
10.000000       22
13.000000       19
14.000000       10
17.000000        7
15.000000        6
19.000000        4
18.000000        4
16.000000        3
21.000000        1
23.000000        1
25.000000        1
20.000000        1
22.000000        1
Name: AMT_REQ_CREDIT_BUREAU_YEAR, dtype: int64
```

[44]:
```
mod_df.AMT_REQ_CREDIT_BUREAU_YEAR = mod_df.AMT_REQ_CREDIT_BUREAU_YEAR.fillna(0.
 ↪0)
```

[45]:
```
mod_df.isnull().sum().sort_values(ascending=False)
```

[45]:
```
EXT_SOURCE_3                  60965
NAME_TYPE_SUITE                1292
DEF_60_CNT_SOCIAL_CIRCLE       1021
OBS_60_CNT_SOCIAL_CIRCLE       1021
DEF_30_CNT_SOCIAL_CIRCLE       1021
                              …
FLAG_DOCUMENT_3                   0
FLAG_DOCUMENT_4                   0
FLAG_DOCUMENT_5                   0
FLAG_DOCUMENT_6                   0
SK_ID_CURR                        0
Length: 81, dtype: int64
```

[46]:
```
mod_df = mod_df.dropna(axis=0)
```

[47]:
```
mod_df.shape
```

[47]:
```
(244708, 81)
```

[48]:
```
# we still have enough data to train our model
```

[49]:
```
#Print percentage of default to payer of the dataset for the TARGET column
```

[50]:
```
default_to_payer = (mod_df.TARGET.value_counts()[1]) / (mod_df.TARGET.
↪value_counts()[0]) * 100
default_to_payer
```

[50]: 8.450148687516897

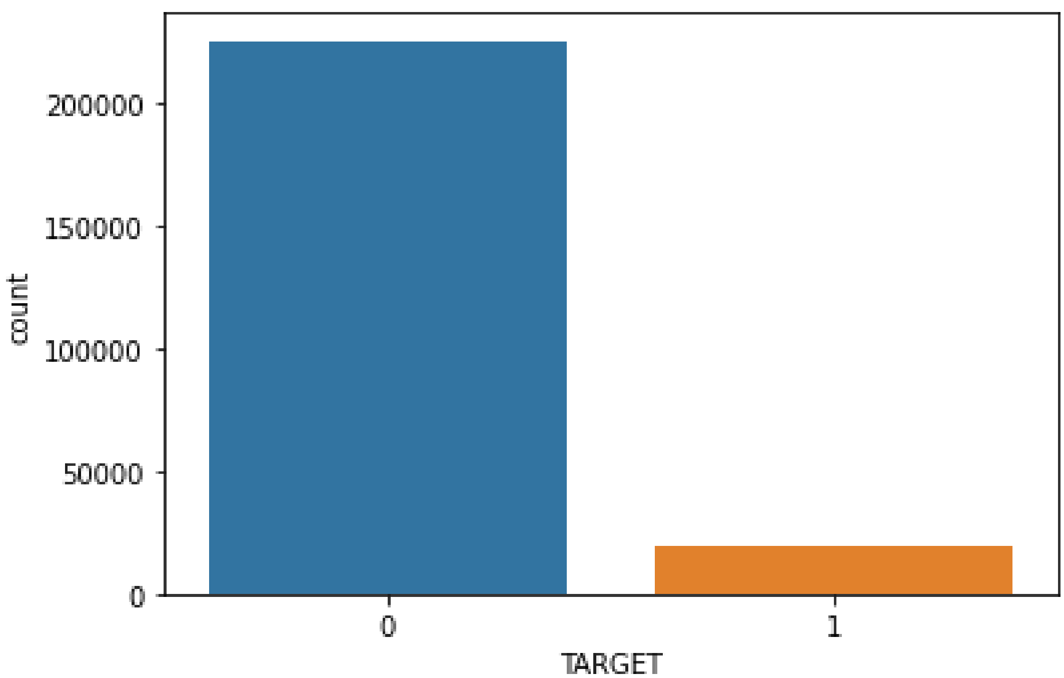
[51]:
```
#default to payer percentage is 8.45%
# here we can see that the data is highly imbalanced.
```

[52]:
```
import seaborn as sns
```

[53]:
```
sns.countplot(mod_df['TARGET'])
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

[53]: <AxesSubplot:xlabel='TARGET', ylabel='count'>



[54]:
```
# Before we treat the imbalance in our data we need to split the data into␣
↪train and test set so that the originality of testing set is not compromised
# Before we split the data we need to lable encode the categorical columns so␣
↪that we will have the same columns in train and test dataset.
```

[55]:
```
X = mod_df.drop('TARGET',axis =1)
```

[56]:
```
y = mod_df.TARGET
```

[57]:
```
X.dtypes.value_counts()
```

[57]:
```
int64      40
float64    27
object     13
dtype: int64
```

[58]:
```
# Label encode the categorical columns
```

[59]:
```
cat_cols = X.select_dtypes(include='object')
```

[60]:
```
cat_cols_encoded = pd.get_dummies(cat_cols,prefix_sep='_')
```

[61]:
```
cat_cols_encoded.shape
```

[61]:
```
(244708, 126)
```

[62]:
```
num_cols = X.select_dtypes(exclude='object')
```

[63]:
```
X_encoded = pd.concat([cat_cols_encoded,num_cols],axis=1)
```

[64]:
```
X_encoded.dtypes.value_counts()
```

[64]:
```
uint8      126
int64       40
float64     27
dtype: int64
```

[65]:
```
# Splitting the model
from sklearn.model_selection import train_test_split
```

[66]:
```
X_train,X_test,y_train,y_test = train_test_split(X_encoded,y,test_size=0.
 ↪2,random_state=22)
```

[67]:
```
# Now we will combine X train and y train and over sample the data to handlw␣
 ↪the imbalance
```

[68]:
```
imbalanced_data = pd.concat([X_train,y_train],axis=1)
```

[69]:
```
# Split into majoirty and minority data
df_majority = imbalanced_data[imbalanced_data.TARGET==0]
df_minority = imbalanced_data[imbalanced_data.TARGET==1]
```

[70]:
```
# Upsample minority class
from sklearn.utils import resample
```

[71]:
```
df_upsampled_minority =␣
↪resample(df_minority,replace=True,n_samples=180596,random_state=123)
```

[72]:
```
df_upsampled_minority.shape
```

[72]: (180596, 194)

[73]:
```
df_majority.shape
```

[73]: (180596, 194)

[74]:
```
df_upsampled = pd.concat([df_majority, df_upsampled_minority])
```

[75]:
```
df_upsampled.shape
```

[75]: (361192, 194)

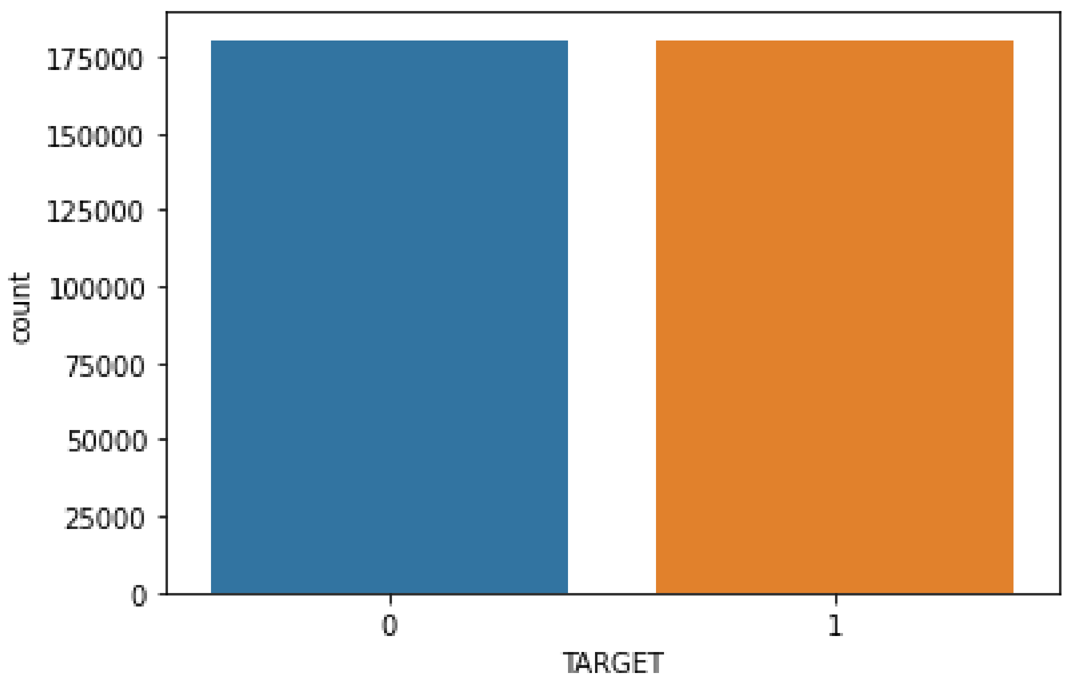
[76]:
```
df_upsampled.TARGET.value_counts()
```

[76]:
```
1    180596
0    180596
Name: TARGET, dtype: int64
```

[77]:
```
sns.countplot(df_upsampled['TARGET'])
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

[77]: <AxesSubplot:xlabel='TARGET', ylabel='count'>



[78]:
```
# Now our data is balanced
```

[79]:
```
# Getting the data ready to be fit in model
```

[80]:
```
df_upsampled_X = df_upsampled.drop(['TARGET','SK_ID_CURR'],axis=1)
```

[81]:
```
df_upsampled_y = df_upsampled.TARGET
```

[82]:
```
X_test = X_test.drop('SK_ID_CURR',axis=1)
```

[83]:
```
df_upsampled_X.shape
```

[83]: (361192, 192)

[84]:
```
#Create model
```

[85]:
```
model = tf.keras.models.Sequential()
```

[86]:
```
model.add(tf.keras.layers.Reshape((192,),input_shape=(192,)))
```

[87]:
```
model.add(tf.keras.layers.BatchNormalization())
```

[88]:
```
model.add(tf.keras.layers.Dense(200, activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
```

[89]:
```
model.add(tf.keras.layers.Dense(100, activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
```

[90]:
```
model.add(tf.keras.layers.Dense(60, activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
```

[91]:
```
model.add(tf.keras.layers.Dense(30, activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
```

[92]:
```
#Output layer
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

[93]:
```
sgd_optimizer = tf.keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=sgd_optimizer, loss='binary_crossentropy',␣
↪metrics=['accuracy'])
```

[94]:
```
model.
↪fit(df_upsampled_X,df_upsampled_y,validation_data=(X_test,y_test),epochs=100,batch_size=32)
```

```
Epoch 1/100
11288/11288 [==============================] - 29s 2ms/step - loss: 0.6000 -
accuracy: 0.6794 - val_loss: 0.5933 - val_accuracy: 0.6813
Epoch 2/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.5657 -
accuracy: 0.7079 - val_loss: 0.5886 - val_accuracy: 0.6881
Epoch 3/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.5386 -
accuracy: 0.7285 - val_loss: 0.5921 - val_accuracy: 0.6893
Epoch 4/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.5148 -
accuracy: 0.7458 - val_loss: 0.5850 - val_accuracy: 0.6948
Epoch 5/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.4944 -
accuracy: 0.7596 - val_loss: 0.5787 - val_accuracy: 0.7035
Epoch 6/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.4783 -
accuracy: 0.7707 - val_loss: 0.6032 - val_accuracy: 0.6944
Epoch 7/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.4620 -
accuracy: 0.7820 - val_loss: 0.5817 - val_accuracy: 0.7075
Epoch 8/100
11288/11288 [==============================] - 29s 3ms/step - loss: 0.4508 -
accuracy: 0.7887 - val_loss: 0.5777 - val_accuracy: 0.7197
Epoch 9/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.4401 -
accuracy: 0.7953 - val_loss: 0.5729 - val_accuracy: 0.7273
Epoch 10/100
```

```
11288/11288 [==============================] - 28s 2ms/step - loss: 0.4301 -
accuracy: 0.8019 - val_loss: 0.5973 - val_accuracy: 0.7191
Epoch 11/100
11288/11288 [==============================] - 30s 3ms/step - loss: 0.4212 -
accuracy: 0.8069 - val_loss: 0.5805 - val_accuracy: 0.7319
Epoch 12/100
11288/11288 [==============================] - 29s 3ms/step - loss: 0.4139 -
accuracy: 0.8113 - val_loss: 0.5801 - val_accuracy: 0.7318
Epoch 13/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.4070 -
accuracy: 0.8155 - val_loss: 0.5795 - val_accuracy: 0.7345
Epoch 14/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.3996 -
accuracy: 0.8193 - val_loss: 0.5828 - val_accuracy: 0.7341
Epoch 15/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.3927 -
accuracy: 0.8229 - val_loss: 0.5742 - val_accuracy: 0.7404
Epoch 16/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.3877 -
accuracy: 0.8260 - val_loss: 0.5847 - val_accuracy: 0.7386
Epoch 17/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.3832 -
accuracy: 0.8296 - val_loss: 0.5750 - val_accuracy: 0.7496
Epoch 18/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3776 -
accuracy: 0.8317 - val_loss: 0.5845 - val_accuracy: 0.7438
Epoch 19/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3747 -
accuracy: 0.8335 - val_loss: 0.5807 - val_accuracy: 0.7459
Epoch 20/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3688 -
accuracy: 0.8369 - val_loss: 0.5765 - val_accuracy: 0.7520
Epoch 21/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3655 -
accuracy: 0.8381 - val_loss: 0.5843 - val_accuracy: 0.7567
Epoch 22/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3622 -
accuracy: 0.8397 - val_loss: 0.5862 - val_accuracy: 0.7516
Epoch 23/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3592 -
accuracy: 0.8424 - val_loss: 0.5825 - val_accuracy: 0.7557
Epoch 24/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3552 -
accuracy: 0.8444 - val_loss: 0.5889 - val_accuracy: 0.7539
Epoch 25/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.3520 -
accuracy: 0.8458 - val_loss: 0.5891 - val_accuracy: 0.7596
Epoch 26/100
```

```
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3498 -
accuracy: 0.8468 - val_loss: 0.5870 - val_accuracy: 0.7548
Epoch 27/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3476 -
accuracy: 0.8476 - val_loss: 0.5966 - val_accuracy: 0.7581
Epoch 28/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3442 -
accuracy: 0.8500 - val_loss: 0.5915 - val_accuracy: 0.7605
Epoch 29/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.3423 -
accuracy: 0.8511 - val_loss: 0.5907 - val_accuracy: 0.7596
Epoch 30/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.3392 -
accuracy: 0.8530 - val_loss: 0.5852 - val_accuracy: 0.7594
Epoch 31/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3385 -
accuracy: 0.8533 - val_loss: 0.5996 - val_accuracy: 0.7578
Epoch 32/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3356 -
accuracy: 0.8547 - val_loss: 0.5933 - val_accuracy: 0.7611
Epoch 33/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3277 -
accuracy: 0.8590 - val_loss: 0.5918 - val_accuracy: 0.7670
Epoch 38/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3249 -
accuracy: 0.8602 - val_loss: 0.5962 - val_accuracy: 0.7704
Epoch 39/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3226 -
accuracy: 0.8615 - val_loss: 0.5748 - val_accuracy: 0.7739
Epoch 40/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3212 -
accuracy: 0.8621 - val_loss: 0.5971 - val_accuracy: 0.7693
Epoch 41/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3192 -
accuracy: 0.8633 - val_loss: 0.5939 - val_accuracy: 0.7676
Epoch 42/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3195 -
accuracy: 0.8631 - val_loss: 0.6066 - val_accuracy: 0.7614
Epoch 43/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3177 -
accuracy: 0.8635 - val_loss: 0.6094 - val_accuracy: 0.7631
Epoch 44/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3153 -
accuracy: 0.8649 - val_loss: 0.5927 - val_accuracy: 0.7728
Epoch 45/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3157 -
accuracy: 0.8647 - val_loss: 0.5890 - val_accuracy: 0.7718
Epoch 46/100
```

```
11288/11288 [==============================] - 28s 3ms/step - loss: 0.3151 -
accuracy: 0.8648 - val_loss: 0.6027 - val_accuracy: 0.7698
Epoch 47/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3132 -
accuracy: 0.8659 - val_loss: 0.6095 - val_accuracy: 0.7666
Epoch 48/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3108 -
accuracy: 0.8674 - val_loss: 0.5981 - val_accuracy: 0.7691
Epoch 49/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3102 -
accuracy: 0.8679 - val_loss: 0.5814 - val_accuracy: 0.7753
Epoch 50/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3095 -
accuracy: 0.8684 - val_loss: 0.6002 - val_accuracy: 0.7680
Epoch 51/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.3093 -
accuracy: 0.8679 - val_loss: 0.5952 - val_accuracy: 0.7763
Epoch 52/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.3070 -
accuracy: 0.8688 - val_loss: 0.5963 - val_accuracy: 0.7752
Epoch 53/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.3057 -
accuracy: 0.8699 - val_loss: 0.5998 - val_accuracy: 0.7778
Epoch 54/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2886 -
accuracy: 0.8785 - val_loss: 0.6027 - val_accuracy: 0.7831
Epoch 78/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2878 -
accuracy: 0.8789 - val_loss: 0.6154 - val_accuracy: 0.7811
Epoch 79/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2874 -
accuracy: 0.8790 - val_loss: 0.6008 - val_accuracy: 0.7826
Epoch 80/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2876 -
accuracy: 0.8783 - val_loss: 0.6065 - val_accuracy: 0.7776
Epoch 81/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2873 -
accuracy: 0.8787 - val_loss: 0.6258 - val_accuracy: 0.7722
Epoch 82/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.2863 -
accuracy: 0.8797 - val_loss: 0.6061 - val_accuracy: 0.7823
Epoch 83/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.2845 -
accuracy: 0.8804 - val_loss: 0.6021 - val_accuracy: 0.7837
Epoch 84/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.2850 -
accuracy: 0.8797 - val_loss: 0.6147 - val_accuracy: 0.7819
Epoch 85/100
```

```
11288/11288 [==============================] - 28s 2ms/step - loss: 0.2843 -
accuracy: 0.8801 - val_loss: 0.6031 - val_accuracy: 0.7809
Epoch 86/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2829 -
accuracy: 0.8811 - val_loss: 0.5967 - val_accuracy: 0.7857
Epoch 87/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2835 -
accuracy: 0.8808 - val_loss: 0.5904 - val_accuracy: 0.7892
Epoch 88/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2834 -
accuracy: 0.8808 - val_loss: 0.5915 - val_accuracy: 0.7885
Epoch 89/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2824 -
accuracy: 0.8812 - val_loss: 0.5991 - val_accuracy: 0.7811
Epoch 90/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2808 -
accuracy: 0.8821 - val_loss: 0.5964 - val_accuracy: 0.7867
Epoch 91/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2805 -
accuracy: 0.8823 - val_loss: 0.6179 - val_accuracy: 0.7784
Epoch 92/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2816 -
accuracy: 0.8815 - val_loss: 0.5890 - val_accuracy: 0.7879
Epoch 93/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.2810 -
accuracy: 0.8819 - val_loss: 0.5924 - val_accuracy: 0.7869
Epoch 94/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2802 -
accuracy: 0.8823 - val_loss: 0.6075 - val_accuracy: 0.7849
Epoch 95/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2797 -
accuracy: 0.8822 - val_loss: 0.6003 - val_accuracy: 0.7840
Epoch 96/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2799 -
accuracy: 0.8827 - val_loss: 0.5950 - val_accuracy: 0.7892
Epoch 97/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2789 -
accuracy: 0.8837 - val_loss: 0.5875 - val_accuracy: 0.7894
Epoch 98/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2794 -
accuracy: 0.8823 - val_loss: 0.5968 - val_accuracy: 0.7847
Epoch 99/100
11288/11288 [==============================] - 28s 2ms/step - loss: 0.2779 -
accuracy: 0.8834 - val_loss: 0.5967 - val_accuracy: 0.7919
Epoch 100/100
11288/11288 [==============================] - 27s 2ms/step - loss: 0.2784 -
accuracy: 0.8838 - val_loss: 0.6052 - val_accuracy: 0.7861
```

[94]: <keras.callbacks.History at 0x7fcb17193810>

[95]:
```
y_pred = model.predict(X_test)
```

[96]:
```
# Calculate Sensitivity as a metrice
```

[97]:
```
m = tf.keras.metrics.Recall()
```

[98]:
```
m.update_state(y_test,y_pred)
```

[99]:
```
m.result().numpy()
```

[99]: 0.36438286

[100]:
```
# Sensitivty of the model is 0.36
```

[101]:
```
#Calculate area under receiver operating characteristics curve
```

[103]:
```
m1 = tf.keras.metrics.AUC(num_thresholds=200,curve="ROC")
```

[104]:
```
m1.update_state(y_test,y_pred)
```

[105]:
```
m1.result().numpy()
```

[105]: 0.6381495

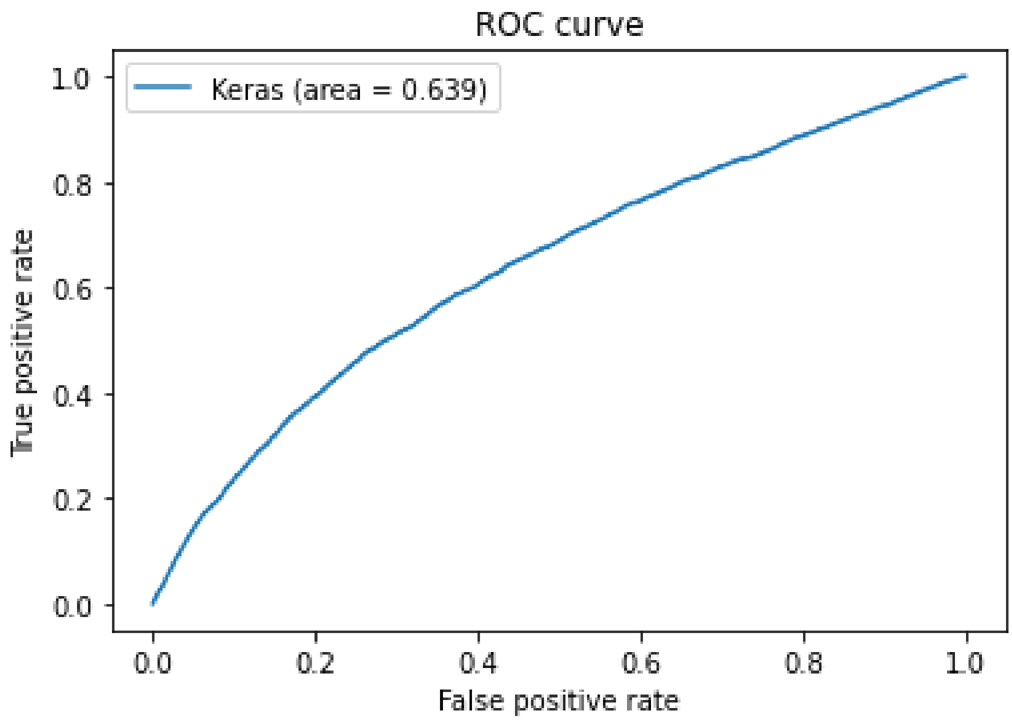
[106]:
```
# The AUC-ROC is 0.63
```

[107]:
```
from sklearn.metrics import roc_curve
y_pred_keras = y_pred.ravel()
fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_test, y_pred_keras)
```

[108]:
```
from sklearn.metrics import auc
auc_keras = auc(fpr_keras, tpr_keras)
```

[109]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

[110]:
```
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```


ROC curve
Keras (area = 0.639)
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]:
```