House loan Project-01

March 26, 2022

1 Name: Sunil Pradhan

Project Name: House Loan Data Analysis

1.1 OBJECTIVE-

Create a model that predicts whether or not an applicant will be able to repay a loan using historical data.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
#Load the dataset
df=pd.read_csv("loan_data1.csv")
df.head()
```

[2]:
```
   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]
```

[3]:
```
df.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
```

[4]:
```
df.describe()
```

[4]:
```
          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]
```

[5]:

```
#Check for null values in the dataset
df.isna().sum()
```

[5]:

```
SK_ID_CURR                        0
TARGET                            0
NAME_CONTRACT_TYPE                0
CODE_GENDER                       0
FLAG_OWN_CAR                      0
                              …
AMT_REQ_CREDIT_BUREAU_DAY     41519
AMT_REQ_CREDIT_BUREAU_WEEK    41519
AMT_REQ_CREDIT_BUREAU_MON     41519
AMT_REQ_CREDIT_BUREAU_QRT     41519
AMT_REQ_CREDIT_BUREAU_YEAR    41519
Length: 122, dtype: int64
```

[6]:

```
df.head()
```

[6]:

```
   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]
```

1.1.1 Print percentage of default to payer of the dataset for the TARGET column

[7]:
```
defaulters=(df["TARGET"]==1).sum()
payers=(df["TARGET"]==0).sum()
defaulter_percent=(defaulters/payers)*100
print("Percentage of default to payer is ",defaulter_percent)
```

```
Percentage of default to payer is  8.781828601345662
```

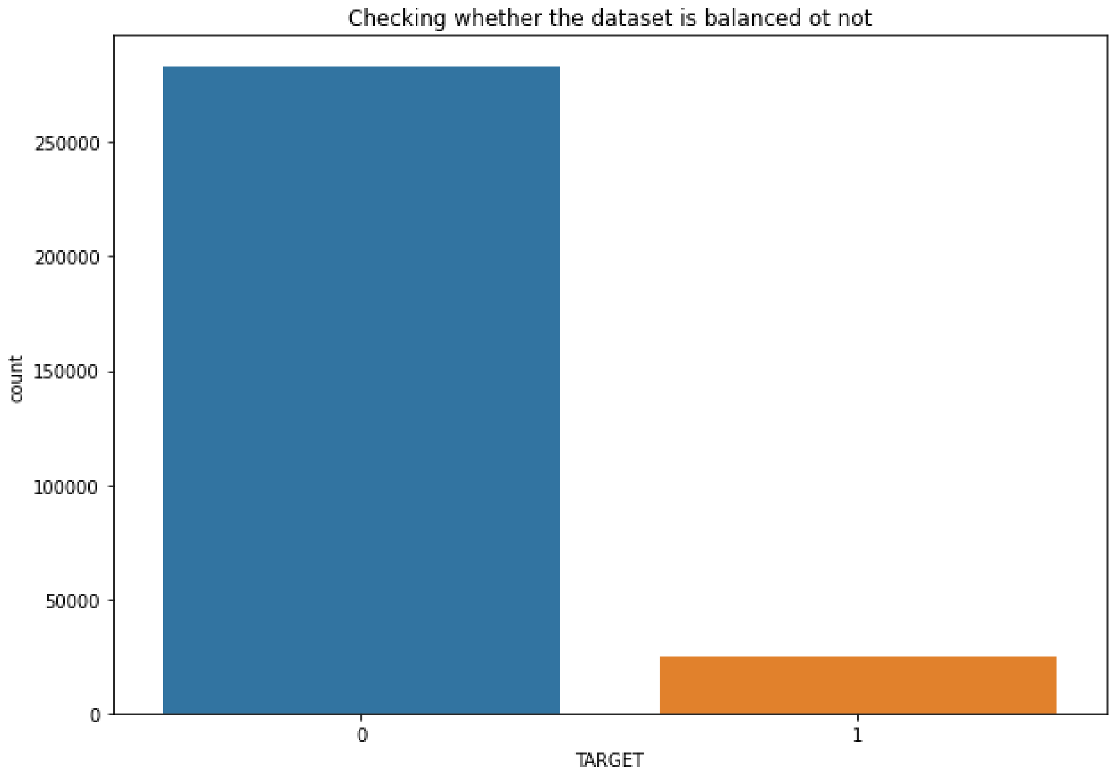
1.2 Balance the dataset if the data is imbalanced

[8]:
```
df["TARGET"].value_counts()
```

[8]:
```
0    282686
1     24825
Name: TARGET, dtype: int64
```

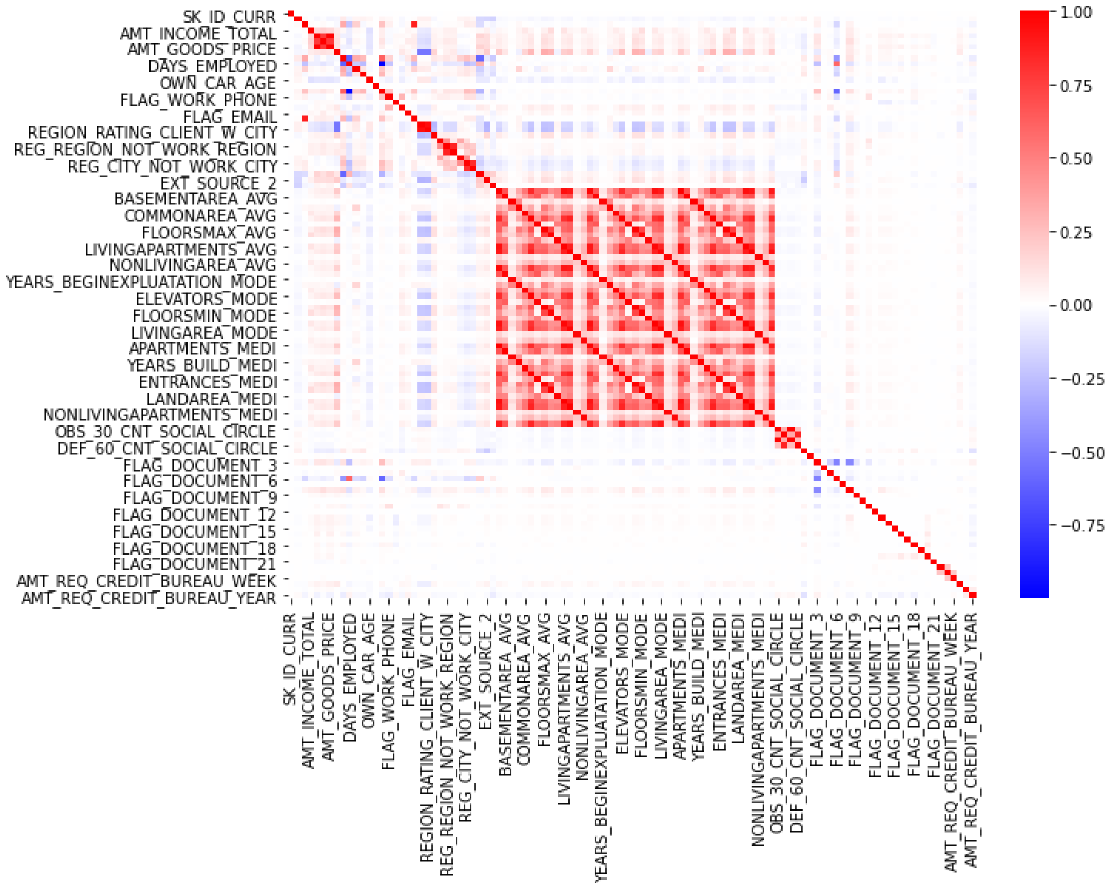
[9]:
```
plt.figure(figsize=(10,7))
sns.countplot(df["TARGET"])
plt.title("Checking whether the dataset is balanced ot not")
```

[9]: Text(0.5, 1.0, 'Checking whether the dataset is balanced ot not')



[10]:
```
plt.figure(figsize=(10,7))
sns.heatmap(df.corr(),cmap="bwr")
```

[10]: <AxesSubplot:>

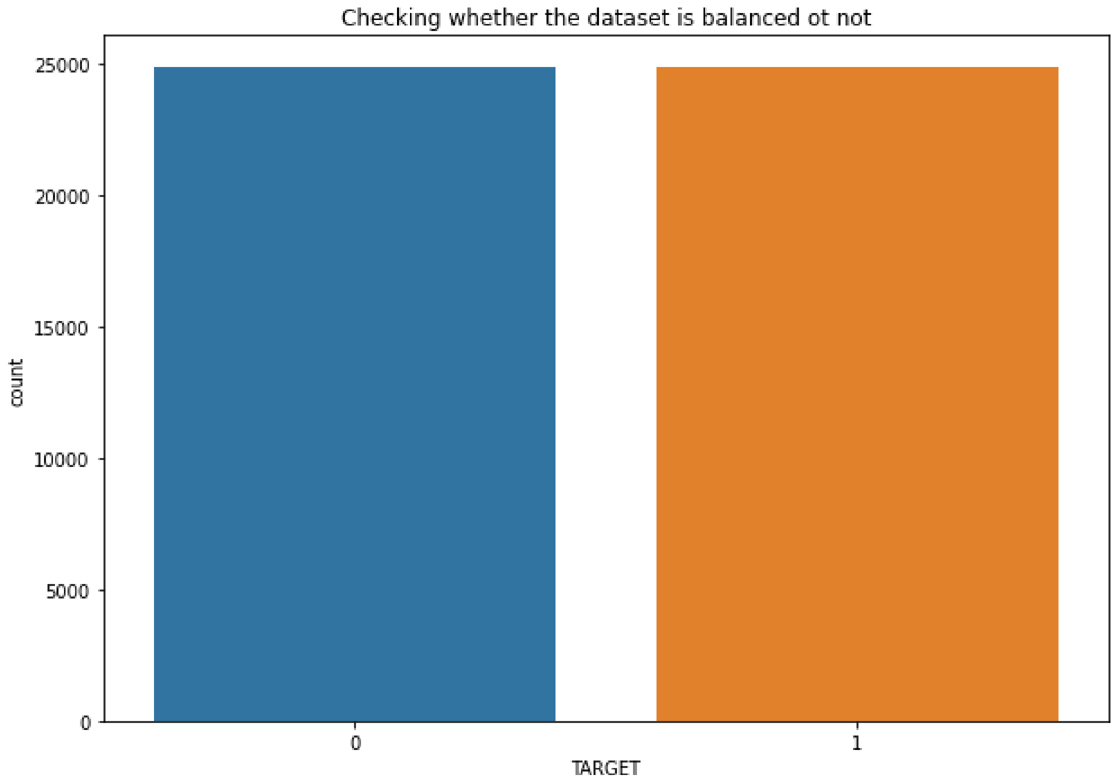


1.3 Balance the dataset using under-fitting random sampling

[11]:
```
shuffled_df=df.sample(frac=1,random_state=25)
fraud_df=shuffled_df[shuffled_df["TARGET"]==1]
nonfraud_df=shuffled_df[shuffled_df["TARGET"]==0].
 ↪sample(n=24825,random_state=25)
balanced_df=pd.concat([fraud_df,nonfraud_df])
```

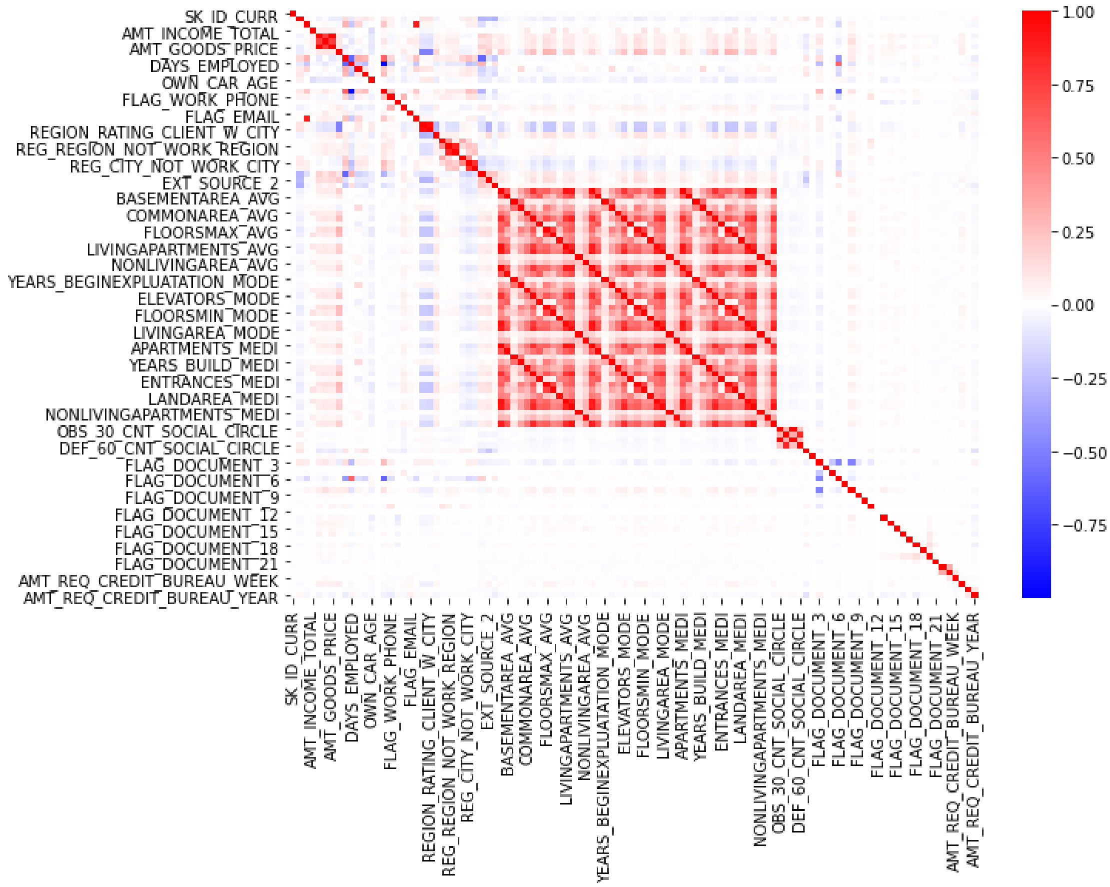
[12]:
```
plt.figure(figsize=(10,7))
sns.countplot(balanced_df["TARGET"])
plt.title("Checking whether the dataset is balanced ot not")
```

[12]:
```
Text(0.5, 1.0, 'Checking whether the dataset is balanced ot not')
```



[13]:
```
plt.figure(figsize=(10,7))
sns.heatmap(balanced_df.corr(),cmap="bwr")
```

[13]: <AxesSubplot:>



[14]:

```
balanced_df.head()
```

[14]:

```
        SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
122136      241602       1         Cash loans           F            N
32365       137520       1         Cash loans           M            Y
95288       210632       1         Cash loans           M            Y
243096      381398       1         Cash loans           F            Y
61628       171473       1         Cash loans           M            N

       FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  \
122136               Y             0           66600.0    808650.0
32365                Y             0          135000.0    512064.0
95288                N             0          180000.0   1078200.0
243096               Y             1          117000.0    539100.0
61628                Y             0          180000.0    900000.0

        AMT_ANNUITY  …  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20  \
122136      31464.0  …                 0                0                0
32365       25033.5  …                 0                0                0
```

```
95288       31522.5  …                0                0                0
243096      27652.5  …                0                0                0
61628       57649.5  …                0                0                0

       FLAG_DOCUMENT_21 AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
122136                0                        NaN                       NaN
32365                 0                        0.0                       0.0
95288                 0                        0.0                       0.0
243096                0                        0.0                       0.0
61628                 0                        0.0                       0.0

        AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
122136                         NaN                        NaN
32365                          0.0                        0.0
95288                          0.0                        0.0
243096                         0.0                        0.0
61628                          0.0                        0.0

        AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
122136                        NaN                         NaN
32365                         0.0                         4.0
95288                         1.0                         0.0
243096                        0.0                         3.0
61628                         0.0                         1.0

[5 rows x 122 columns]
```

[15]:
```
balanced_df.describe(include="object")
```

[15]:
```
       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                45558       30740        33534           34232

       NAME_TYPE_SUITE NAME_INCOME_TYPE            NAME_EDUCATION_TYPE  \
count            49470            49650                          49650
unique               7                6                              5
top      Unaccompanied          Working  Secondary / secondary special
freq             40400            27857                          36986

       NAME_FAMILY_STATUS  NAME_HOUSING_TYPE OCCUPATION_TYPE  \
count               49650              49650           35402
unique                  5                  6              18
top               Married  House / apartment        Laborers
freq                30843              43361           10110
```

```
       WEEKDAY_APPR_PROCESS_START       ORGANIZATION_TYPE FONDKAPREMONT_MODE  \
count                       49650                   49650              14589
unique                          7                      58                  4
top                       TUESDAY  Business Entity Type 3   reg oper account
freq                         8921                   11771              11107

        HOUSETYPE_MODE WALLSMATERIAL_MODE EMERGENCYSTATE_MODE
count            23116              22791               24411
unique               3                  7                   2
top     block of flats       Stone, brick                  No
freq             22655              10097               23997
```

[16]:
```
balanced_df.
 ↪drop(columns=(["WEEKDAY_APPR_PROCESS_START","FLAG_OWN_REALTY","NAME_TYPE_SUITE","FONDKAPREM
balanced_df.describe(include="object")
```

[16]:
```
       NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR NAME_INCOME_TYPE  \
count               49650       49650        49650            49650
unique                  2           2            2                6
top            Cash loans           F            N          Working
freq                45558       30740        33534            27857

                  NAME_EDUCATION_TYPE NAME_FAMILY_STATUS  NAME_HOUSING_TYPE  \
count                           49650              49650              49650
unique                              5                  5                  6
top     Secondary / secondary special            Married  House / apartment
freq                            36986              30843              43361

       OCCUPATION_TYPE       ORGANIZATION_TYPE  HOUSETYPE_MODE  \
count            35402                   49650           23116
unique              18                      58               3
top           Laborers  Business Entity Type 3  block of flats
freq             10110                   11771           22655

       WALLSMATERIAL_MODE
count               22791
unique                  7
top          Stone, brick
freq                10097
```

1.4 Encode the columns that is required for the model

[17]:
```
from sklearn.preprocessing import LabelEncoder
```

[18]:
```
le=LabelEncoder()
balanced_df["NAME_CONTRACT_TYPE"]=le.
 ↪fit_transform(balanced_df["NAME_CONTRACT_TYPE"])
```

```
balanced_df["CODE_GENDER"]=le.fit_transform(balanced_df["CODE_GENDER"])
balanced_df["FLAG_OWN_CAR"]=le.fit_transform(balanced_df["FLAG_OWN_CAR"])
balanced_df["NAME_INCOME_TYPE"]=le.
 ↪fit_transform(balanced_df["NAME_INCOME_TYPE"])
balanced_df["NAME_EDUCATION_TYPE"]=le.
 ↪fit_transform(balanced_df["NAME_EDUCATION_TYPE"])
balanced_df["NAME_FAMILY_STATUS"]=le.
 ↪fit_transform(balanced_df["NAME_FAMILY_STATUS"])
balanced_df["NAME_HOUSING_TYPE"]=le.
 ↪fit_transform(balanced_df["NAME_HOUSING_TYPE"])
balanced_df["OCCUPATION_TYPE"]=le.fit_transform(balanced_df["OCCUPATION_TYPE"])
balanced_df["ORGANIZATION_TYPE"]=le.
 ↪fit_transform(balanced_df["ORGANIZATION_TYPE"])
balanced_df["HOUSETYPE_MODE"]=le.fit_transform(balanced_df["HOUSETYPE_MODE"])
balanced_df["NAME_CONTRACT_TYPE"]=le.
 ↪fit_transform(balanced_df["NAME_CONTRACT_TYPE"])
balanced_df["WALLSMATERIAL_MODE"]=le.
 ↪fit_transform(balanced_df["WALLSMATERIAL_MODE"])
```

[19]:

```
balanced_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49650 entries, 122136 to 68357
Columns: 117 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int32(10), int64(42)
memory usage: 43.8 MB
```

[20]:

```
balanced_df.head()
```

[20]:

```
        SK_ID_CURR  TARGET  NAME_CONTRACT_TYPE  CODE_GENDER  FLAG_OWN_CAR  \
122136      241602       1                   0            0             0
32365       137520       1                   0            1             1
95288       210632       1                   0            1             1
243096      381398       1                   0            0             1
61628       171473       1                   0            1             0

        CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
122136             0           66600.0    808650.0      31464.0
32365              0          135000.0    512064.0      25033.5
95288              0          180000.0   1078200.0      31522.5
243096             1          117000.0    539100.0      27652.5
61628              0          180000.0    900000.0      57649.5

        AMT_GOODS_PRICE  …  FLAG_DOCUMENT_18  FLAG_DOCUMENT_19  \
122136         675000.0  …                 0                 0
32365          360000.0  …                 0                 0
95288          900000.0  …                 0                 0
```

```
243096        450000.0  ...                 0                  0
61628         900000.0  ...                 0                  0

        FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  AMT_REQ_CREDIT_BUREAU_HOUR  \
122136                 0                 0                         NaN
32365                  0                 0                         0.0
95288                  0                 0                         0.0
243096                 0                 0                         0.0
61628                  0                 0                         0.0

        AMT_REQ_CREDIT_BUREAU_DAY  AMT_REQ_CREDIT_BUREAU_WEEK  \
122136                        NaN                         NaN
32365                         0.0                         0.0
95288                         0.0                         0.0
243096                        0.0                         0.0
61628                         0.0                         0.0

        AMT_REQ_CREDIT_BUREAU_MON  AMT_REQ_CREDIT_BUREAU_QRT  \
122136                        NaN                        NaN
32365                         0.0                        0.0
95288                         0.0                        1.0
243096                        0.0                        0.0
61628                         0.0                        0.0

        AMT_REQ_CREDIT_BUREAU_YEAR
122136                         NaN
32365                          4.0
95288                          0.0
243096                         3.0
61628                          1.0

[5 rows x 117 columns]
```

[21]:
```
x=balanced_df.drop(["TARGET"],axis=1)
y=balanced_df[["TARGET"]]
```

[22]:
```
x.head()
```

[22]:
```
        SK_ID_CURR  NAME_CONTRACT_TYPE  CODE_GENDER  FLAG_OWN_CAR  \
122136      241602                   0            0             0
32365       137520                   0            1             1
95288       210632                   0            1             1
243096      381398                   0            0             1
61628       171473                   0            1             0

        CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
122136             0           66600.0    808650.0      31464.0
```

```
32365                0          135000.0    512064.0      25033.5
95288                0          180000.0   1078200.0      31522.5
243096               1          117000.0    539100.0      27652.5
61628                0          180000.0    900000.0      57649.5

        AMT_GOODS_PRICE  NAME_INCOME_TYPE  …  FLAG_DOCUMENT_18  \
122136         675000.0                 2  …                 0
32365          360000.0                 0  …                 0
95288          900000.0                 5  …                 0
243096         450000.0                 5  …                 0
61628          900000.0                 5  …                 0

        FLAG_DOCUMENT_19  FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  \
122136                 0                 0                 0
32365                  0                 0                 0
95288                  0                 0                 0
243096                 0                 0                 0
61628                  0                 0                 0

        AMT_REQ_CREDIT_BUREAU_HOUR  AMT_REQ_CREDIT_BUREAU_DAY  \
122136                         NaN                        NaN
32365                          0.0                        0.0
95288                          0.0                        0.0
243096                         0.0                        0.0
61628                          0.0                        0.0

        AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
122136                         NaN                        NaN
32365                          0.0                        0.0
95288                          0.0                        0.0
243096                         0.0                        0.0
61628                          0.0                        0.0

        AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
122136                        NaN                         NaN
32365                         0.0                         4.0
95288                         1.0                         0.0
243096                        0.0                         3.0
61628                         0.0                         1.0

[5 rows x 116 columns]
```

[23]:
```
y.head()
```

[23]:
```
        TARGET
122136       1
32365        1
```

```
95288        1
243096       1
61628        1
```

[24]:
```
from sklearn.model_selection import train_test_split
```

[25]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪25,random_state=25)
```

[26]:
```
x_train.shape,x_test.shape
```

[26]:
```
((37237, 116), (12413, 116))
```

[27]:
```
y_train.shape,y_test.shape
```

[27]:
```
((37237, 1), (12413, 1))
```

[28]:
```
from sklearn.preprocessing import StandardScaler
```

[29]:
```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

1.5 Architect the Deep learning Model

[30]:
```
import tensorflow as tf
```

[31]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout
```

[32]:
```
model=Sequential()
model.add(Dense(256,activation="relu",input_shape=(x_train.shape[1],)))
model.add(Dropout(0.3))
model.add(Dense(128,activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(64,activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(1,activation="sigmoid"))
```

[33]:
```
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               29952
```

```
 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: 71,169
Trainable params: 71,169
Non-trainable params: 0
_________________________________________________________________
```

[34]:
```python
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])
```

[35]:
```python
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100)
```

```
Epoch 1/100
1164/1164 [==============================] - 7s 5ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 2/100
1164/1164 [==============================] - 6s 5ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 3/100
1164/1164 [==============================] - 6s 5ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 4/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 5/100
1164/1164 [==============================] - 5s 5ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 6/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 7/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 8/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 9/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
```

```
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 10/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 11/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 12/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 13/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 14/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 15/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 16/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 17/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 18/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 19/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 20/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 21/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 22/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 23/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 24/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 25/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
```

```
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 26/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 27/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 28/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 29/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 30/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 31/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 32/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 33/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 34/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 35/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 36/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 37/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 38/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 39/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 40/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 41/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
```

```
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 42/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 43/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 44/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 45/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 46/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 47/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 48/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 49/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 50/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 51/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 52/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 53/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 54/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 55/100
1164/1164 [==============================] - 4s 3ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 56/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 57/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
```

```
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 58/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 59/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 60/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 61/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 62/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 63/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 64/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 65/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 66/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 67/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 68/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 69/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 70/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 71/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 72/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 73/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
```

```
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 74/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 75/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 76/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 77/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 78/100
1164/1164 [==============================] - 5s 5ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 79/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 80/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 81/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 82/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 83/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 84/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 85/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 86/100
1164/1164 [==============================] - 5s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 87/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 88/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 89/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
```

```
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 90/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 91/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 92/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 93/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 94/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 95/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 96/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 97/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 98/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 99/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
Epoch 100/100
1164/1164 [==============================] - 4s 4ms/step - loss: nan - accuracy:
0.5005 - val_loss: nan - val_accuracy: 0.4984
```

[35]:
```
<keras.callbacks.History at 0x29c504b2fa0>
```

[36]:
```python
pred=(model.predict(x_test)>0.5)*1.0
pred
```

[36]:
```
array([[0.],
       [0.],
       [0.],
       …,
       [0.],
       [0.],
       [0.]])
```

[37]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

[38]:
```
print(confusion_matrix(pred,y_test))
```

```
[[6187 6226]
 [   0    0]]
```

[39]:
```
print(classification_report(pred,y_test))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.50      0.67     12413
         1.0       0.00      0.00      0.00         0

    accuracy                           0.50     12413
   macro avg       0.50      0.25      0.33     12413
weighted avg       1.00      0.50      0.67     12413
```

1.6 Calculate Sensitivity as a metrics

Sensitivity=TP/(FN+TP)

Specificity=TN/(FP+TN)

[40]:
```
print("Sensitivity of the dataset is",6187/(0+6187))
```

```
Sensitivity of the dataset is 1.0
```

1.7 Calculate area under receiver operating characteristics curve

[41]:
```
from sklearn.metrics import roc_curve
```

[42]:
```
fpr,tpr,thresholds=roc_curve(pred,y_test)
print(fpr)
print("\n")
print(tpr)
print("\n")
print(thresholds)
```

```
[0.         0.50157093 1.        ]


[nan nan nan]


[2 1 0]
```

1.8 THANK YOU...!!!