term_deposits

January 20, 2023

1 Term Deposits

This is a classification project where I try to use SMOTE for my first time.

Dataset: https://www.kaggle.com/datasets/prakharrathi25/banking-dataset-marketing-targets?select=train.csv

The purpose of the following work is to work with classification problems where one target class is over-represented.

It is based on the Codebasics video: https://www.youtube.com/watch?v=JnlM4yLFNuo&t=1413s

2 Importing Libraries

[165]:
```
import pandas as pd
import numpy as np
import opendatasets as od
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

sns.set_style("darkgrid")
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (15, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'

import warnings
warnings.simplefilter(action='ignore')
```

3 Importing Data

[166]:
```
od.download('https://www.kaggle.com/datasets/prakharrathi25/
  ↪banking-dataset-marketing-targets?select=train.csv')
```

```
Skipping, found downloaded files in ".\banking-dataset-marketing-targets" (use
force=True to force download)
```

[167]:
```
bank = pd.read_csv('./banking-dataset-marketing-targets/train.csv', delimiter=';
  ↪').rename(columns={'y':'subscription'})
bank
```

[167]:

	age	job	marital	education	default	balance	housing	loan	\
0	58	management	married	tertiary	no	2143	yes	no	
1	44	technician	single	secondary	no	29	yes	no	
2	33	entrepreneur	married	secondary	no	2	yes	yes	
3	47	blue-collar	married	unknown	no	1506	yes	no	
4	33	unknown	single	unknown	no	1	no	no	
…	…	…	…	…	…	…	…	…	…
45206	51	technician	married	tertiary	no	825	no	no	
45207	71	retired	divorced	primary	no	1729	no	no	
45208	72	retired	married	secondary	no	5715	no	no	
45209	57	blue-collar	married	secondary	no	668	no	no	
45210	37	entrepreneur	married	secondary	no	2971	no	no	

	contact	day	month	duration	campaign	pdays	previous	poutcome	\
0	unknown	5	may	261	1	-1	0	unknown	
1	unknown	5	may	151	1	-1	0	unknown	
2	unknown	5	may	76	1	-1	0	unknown	
3	unknown	5	may	92	1	-1	0	unknown	
4	unknown	5	may	198	1	-1	0	unknown	
…	…	…	…	…	…	…	…	…	…
45206	cellular	17	nov	977	3	-1	0	unknown	
45207	cellular	17	nov	456	2	-1	0	unknown	
45208	cellular	17	nov	1127	5	184	3	success	
45209	telephone	17	nov	508	4	-1	0	unknown	
45210	cellular	17	nov	361	2	188	11	other	

	subscription
0	no
1	no
2	no
3	no
4	no
…	…
45206	yes
45207	yes
45208	yes
45209	no
45210	no

[45211 rows x 17 columns]

4 EDA

A few questions for EDA:

1- What is the average number of contacts received by customers who subscribed and ones who did not?
2- Is a previous succesfull campaign more likely to lead to a subscription?
3- Which contact type is the most succesful?
4- Which previous contact month is the most frequent among those who subscribed?
5- What is the average balance of new subscribers?

4.1 General Information

[168]:
```
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   age           45211 non-null  int64 
 1   job           45211 non-null  object
 2   marital       45211 non-null  object
 3   education     45211 non-null  object
 4   default       45211 non-null  object
 5   balance       45211 non-null  int64 
 6   housing       45211 non-null  object
 7   loan          45211 non-null  object
 8   contact       45211 non-null  object
 9   day           45211 non-null  int64 
 10  month         45211 non-null  object
 11  duration      45211 non-null  int64 
 12  campaign      45211 non-null  int64 
 13  pdays         45211 non-null  int64 
 14  previous      45211 non-null  int64 
 15  poutcome      45211 non-null  object
 16  subscription  45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

There are no missing values.

[169]:
```
bank[bank.duplicated(keep=False)]
```

[169]:
```
Empty DataFrame
Columns: [age, job, marital, education, default, balance, housing, loan,
contact, day, month, duration, campaign, pdays, previous, poutcome,
subscription]
Index: []
```

There are no duplicated values.

[170]:
```
bank.describe()
```

[170]:
```
                age        balance           day      duration      campaign  \
count  45211.000000   45211.000000  45211.000000  45211.000000  45211.000000
mean      40.936210    1362.272058     15.806419    258.163080      2.763841
std       10.618762    3044.765829      8.322476    257.527812      3.098021
min       18.000000   -8019.000000      1.000000      0.000000      1.000000
25%       33.000000      72.000000      8.000000    103.000000      1.000000
50%       39.000000     448.000000     16.000000    180.000000      2.000000
75%       48.000000    1428.000000     21.000000    319.000000      3.000000
max       95.000000  102127.000000     31.000000   4918.000000     63.000000

              pdays      previous
count  45211.000000  45211.000000
mean      40.197828      0.580323
std      100.128746      2.303441
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       -1.000000      0.000000
max      871.000000    275.000000
```

[171]:
```
bank.corr()
```

[171]:
```
                age   balance       day  duration  campaign     pdays  previous
age        1.000000  0.097783 -0.009120 -0.004648  0.004760 -0.023758  0.001288
balance    0.097783  1.000000  0.004503  0.021560 -0.014578  0.003435  0.016674
day       -0.009120  0.004503  1.000000 -0.030206  0.162490 -0.093044 -0.051710
duration  -0.004648  0.021560 -0.030206  1.000000 -0.084570 -0.001565  0.001203
campaign   0.004760 -0.014578  0.162490 -0.084570  1.000000 -0.088628 -0.032855
pdays     -0.023758  0.003435 -0.093044 -0.001565 -0.088628  1.000000  0.454820
previous   0.001288  0.016674 -0.051710  0.001203 -0.032855  0.454820  1.000000
```

4.2 Question 1 - What is the average number of contacts received by customers who subscribed and ones who did not?

Creating two datasets to distinguish who subscribed and who did not subscribe.

[172]:
```
sub=bank[bank.subscription == 'yes']
no_sub=bank[bank.subscription == 'no']
```

[173]:
```
sub.describe()
```

[173]:
```
                age       balance           day     duration     campaign  \
count  5289.000000   5289.000000  5289.000000  5289.000000  5289.000000
mean     41.670070   1804.267915    15.158253   537.294574     2.141047
```

```
std      13.497781   3501.104777     8.501875   392.525262     1.921826
min      18.000000  -3058.000000     1.000000     8.000000     1.000000
25%      31.000000    210.000000     8.000000   244.000000     1.000000
50%      38.000000    733.000000    15.000000   426.000000     2.000000
75%      50.000000   2159.000000    22.000000   725.000000     3.000000
max      95.000000  81204.000000    31.000000  3881.000000    32.000000

             pdays     previous
count  5289.000000  5289.000000
mean     68.702968     1.170354
std     118.822266     2.553272
min      -1.000000     0.000000
25%      -1.000000     0.000000
50%      -1.000000     0.000000
75%      98.000000     1.000000
max     854.000000    58.000000
```

[174]:
```
no_sub.describe()
```

[174]:
```
                age        balance           day      duration      campaign  \
count  39922.000000   39922.000000  39922.000000  39922.000000  39922.000000
mean      40.838986    1303.714969     15.892290    221.182806      2.846350
std       10.172662    2974.195473      8.294728    207.383237      3.212767
min       18.000000   -8019.000000      1.000000      0.000000      1.000000
25%       33.000000      58.000000      8.000000     95.000000      1.000000
50%       39.000000     417.000000     16.000000    164.000000      2.000000
75%       48.000000    1345.000000     21.000000    279.000000      3.000000
max       95.000000  102127.000000     31.000000   4918.000000     63.000000

             pdays      previous
count  39922.000000  39922.000000
mean      36.421372      0.502154
std       96.757135      2.256771
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       -1.000000      0.000000
max      871.000000    275.000000
```

On average, who subscribed received more previous contacts.

4.3 Question 2 - Is a previous succesfull campaign more likely to lead to a subscription?

[175]:
```
sub.poutcome.value_counts()
```

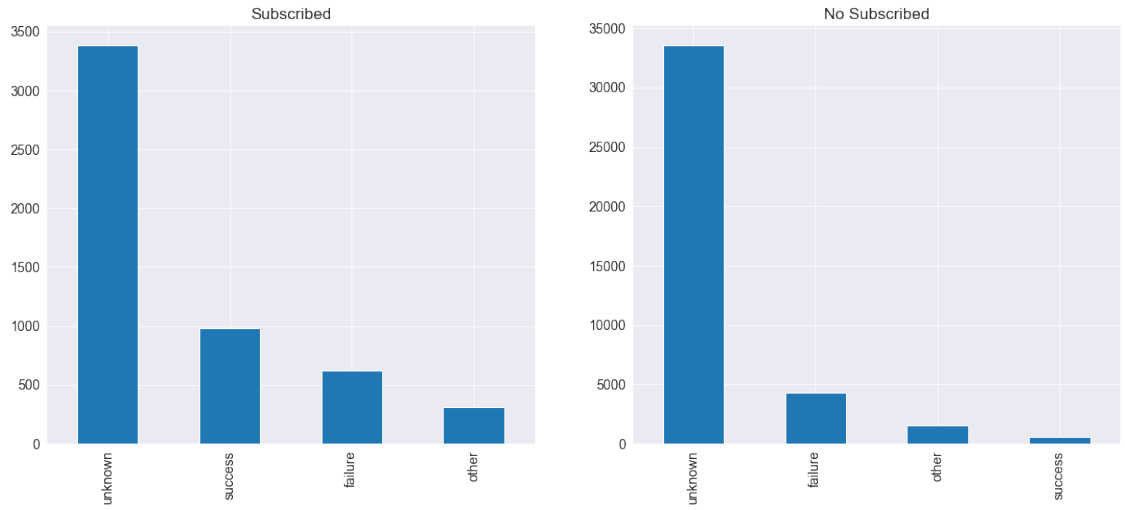
[175]:
```
unknown    3386
success     978
failure     618
other       307
Name: poutcome, dtype: int64
```

[176]:
```
no_sub.poutcome.value_counts()
```

[176]:
```
unknown    33573
failure     4283
other       1533
success      533
Name: poutcome, dtype: int64
```

[177]:
```
fig, ax = plt.subplots(1,2, figsize=(20,8))

sub.poutcome.value_counts().plot(kind='bar', ax=ax[0], title='Subscribed')
no_sub.poutcome.value_counts().plot(kind='bar', ax=ax[1], title='No␣
↪Subscribed');
```



For both situations the unknown class is the most common one.

By the way, among those ones who subscribed the previous campaign had more success.

4.4 Question 3 - Which contact type is the most succesful?

Dividing the number of people who subscribed with a contact type by the entire number of people contacted with that mean.

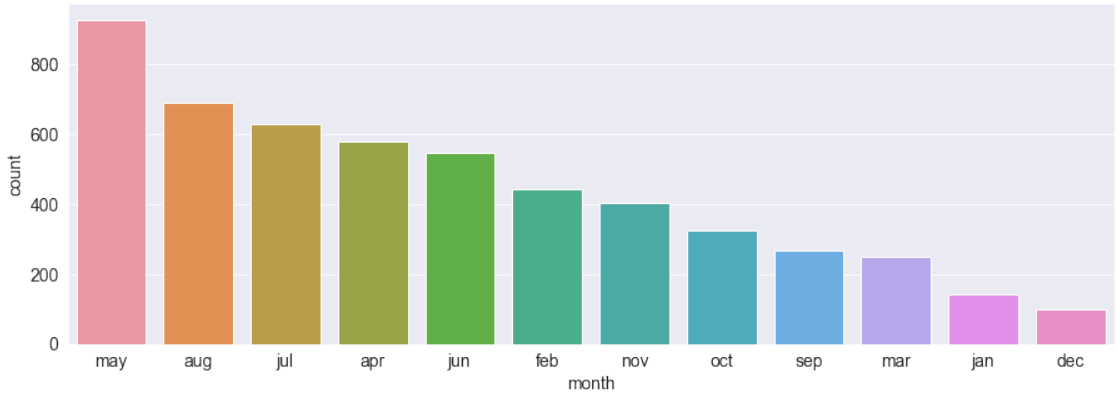
[178]:
```
(sub.contact.value_counts()/bank.contact.value_counts()).
↪sort_values(ascending=False)
```

[178]:
```
cellular     0.149189
telephone    0.134205
unknown      0.040707
Name: contact, dtype: float64
```

Cellular seems to be the most succesful one.

4.5 Question 4 - Which previous contact month is the most frequent among those who subscribed?

[179]:
```
sns.countplot(data=sub, x='month', order=sub.month.value_counts().index);
```



Probably the campaign was done during the spring and summer period.

4.6 Question 5 - What is the average balance of new subscribers?

[180]:
```
sub.balance.describe()
```

[180]:
```
count     5289.000000
mean      1804.267915
std       3501.104777
min      -3058.000000
25%        210.000000
50%        733.000000
75%       2159.000000
max      81204.000000
Name: balance, dtype: float64
```

[181]:
```
no_sub.balance.describe()
```

[181]:
```
count     39922.000000
mean       1303.714969
std        2974.195473
min       -8019.000000
```

```
25%          58.000000
50%         417.000000
75%        1345.000000
max      102127.000000
Name: balance, dtype: float64
```

The subscribers have, on average, a higher balance.

5 Feature Engineering

Dividing in inputs and target.

[182]:
```
X = bank.drop(columns='subscription')
y = bank.subscription
```

[183]:
```
X
```

[183]:
```
       age           job   marital  education default  balance housing loan  \
0       58    management   married   tertiary      no     2143     yes   no
1       44    technician    single  secondary      no       29     yes   no
2       33  entrepreneur   married  secondary      no        2     yes  yes
3       47   blue-collar   married    unknown      no     1506     yes   no
4       33       unknown    single    unknown      no        1      no   no
…      …             …       …         …      …      …   …
45206   51    technician   married   tertiary      no      825      no   no
45207   71       retired  divorced    primary      no     1729      no   no
45208   72       retired   married  secondary      no     5715      no   no
45209   57   blue-collar   married  secondary      no      668      no   no
45210   37  entrepreneur   married  secondary      no     2971      no   no

         contact  day month  duration  campaign  pdays  previous poutcome
0        unknown    5   may       261         1     -1         0  unknown
1        unknown    5   may       151         1     -1         0  unknown
2        unknown    5   may        76         1     -1         0  unknown
3        unknown    5   may        92         1     -1         0  unknown
4        unknown    5   may       198         1     -1         0  unknown
…          …   …   …       …         …    …        …       …
45206   cellular   17   nov       977         3     -1         0  unknown
45207   cellular   17   nov       456         2     -1         0  unknown
45208   cellular   17   nov      1127         5    184         3  success
45209  telephone   17   nov       508         4     -1         0  unknown
45210   cellular   17   nov       361         2    188        11    other

[45211 rows x 16 columns]
```

[184]:
```
y
```

[184]:
```
0         no
1         no
2         no
3         no
4         no
        …
45206    yes
45207    yes
45208    yes
45209     no
45210     no
Name: subscription, Length: 45211, dtype: object
```

5.1 Defining Categorical and Numerical Columns

[185]:
```
numerical_cols = X.select_dtypes(include=np.number).columns.to_list()
categorical_cols = X.select_dtypes(include='object').columns.to_list()
```

[186]:
```
numerical_cols
```

[186]:
```
['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous']
```

[187]:
```
categorical_cols
```

[187]:
```
['job',
 'marital',
 'education',
 'default',
 'housing',
 'loan',
 'contact',
 'month',
 'poutcome']
```

5.2 Scaling Numerical Columns

[188]:
```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler().fit(X[numerical_cols])

X[numerical_cols] = scaler.transform(X[numerical_cols])
```

[189]:
```
X
```

[189]:
```
            age           job   marital  education default   balance housing  \
0      1.266667    management   married   tertiary      no  1.250000     yes
1      0.333333    technician    single  secondary      no -0.308997     yes
```

```
2     -0.400000  entrepreneur   married  secondary      no -0.328909     yes
3      0.533333   blue-collar   married    unknown      no  0.780236     yes
4     -0.400000       unknown    single    unknown      no -0.329646      no
…           …             …       …         …      …       …      …
45206  0.800000    technician   married   tertiary      no  0.278024      no
45207  2.133333       retired  divorced    primary      no  0.944690      no
45208  2.200000       retired   married  secondary      no  3.884218      no
45209  1.200000   blue-collar   married  secondary      no  0.162242      no
45210 -0.133333  entrepreneur   married  secondary      no  1.860619      no

      loan    contact       day month  duration  campaign  pdays  previous  \
0       no    unknown -0.846154   may  0.375000      -0.5    0.0       0.0
1       no    unknown -0.846154   may -0.134259      -0.5    0.0       0.0
2      yes    unknown -0.846154   may -0.481481      -0.5    0.0       0.0
3       no    unknown -0.846154   may -0.407407      -0.5    0.0       0.0
4       no    unknown -0.846154   may  0.083333      -0.5    0.0       0.0
…    …        …       …   …        …       …    …        …
45206   no   cellular  0.076923   nov  3.689815       0.5    0.0       0.0
45207   no   cellular  0.076923   nov  1.277778       0.0    0.0       0.0
45208   no   cellular  0.076923   nov  4.384259       1.5  185.0       3.0
45209   no  telephone  0.076923   nov  1.518519       1.0    0.0       0.0
45210   no   cellular  0.076923   nov  0.837963       0.0  189.0      11.0

      poutcome
0      unknown
1      unknown
2      unknown
3      unknown
4      unknown
…        …
45206  unknown
45207  unknown
45208  success
45209  unknown
45210    other

[45211 rows x 16 columns]
```

5.3 One-Hot Encoding Categorical Columns

[190]:
```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse=False).fit(X[categorical_cols])

encoded_cols = list(encoder.get_feature_names_out(categorical_cols))

X[encoded_cols] = encoder.transform(X[categorical_cols])
```

[191]:
```
X.drop(columns=categorical_cols, inplace=True)
```

5.4 Changing “yes” and “no” into 0 and 1

[192]:
```
subs_dict = {'no':0, 'yes':1}

y = y.map(subs_dict)
```

[193]:
```
y
```

[193]:
```
0        0
1        0
2        0
3        0
4        0
        ..
45206    1
45207    1
45208    1
45209    0
45210    0
Name: subscription, Length: 45211, dtype: int64
```

6 Splitting the Dataest

[194]:
```
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(X,y, test_size=0.20,␣
 ↪random_state=42)
```

7 Creating the Machine Learning Model

[195]:
```
from xgboost import XGBClassifier

ml_model = XGBClassifier(n_jobs=-1, n_estimators=1000,␣
 ↪early_stopping_rounds=50, random_state=42)

ml_model.fit(X_train, y_train, eval_set=[(X_train, y_train), (X_val, y_val)])
```

```
[0]     validation_0-logloss:0.50882    validation_1-logloss:0.51269
[1]     validation_0-logloss:0.40632    validation_1-logloss:0.41326
[2]     validation_0-logloss:0.34284    validation_1-logloss:0.35205
[3]     validation_0-logloss:0.30024    validation_1-logloss:0.31194
[4]     validation_0-logloss:0.27136    validation_1-logloss:0.28540
[5]     validation_0-logloss:0.25087    validation_1-logloss:0.26762
[6]     validation_0-logloss:0.23562    validation_1-logloss:0.25386
```

```
[7]     validation_0-logloss:0.22489    validation_1-logloss:0.24418
[8]     validation_0-logloss:0.21467    validation_1-logloss:0.23446
[9]     validation_0-logloss:0.20816    validation_1-logloss:0.22900
[10]    validation_0-logloss:0.20238    validation_1-logloss:0.22402
[11]    validation_0-logloss:0.19811    validation_1-logloss:0.22068
[12]    validation_0-logloss:0.19492    validation_1-logloss:0.21786
[13]    validation_0-logloss:0.19257    validation_1-logloss:0.21655
[14]    validation_0-logloss:0.19002    validation_1-logloss:0.21482
[15]    validation_0-logloss:0.18766    validation_1-logloss:0.21355
[16]    validation_0-logloss:0.18416    validation_1-logloss:0.21096
[17]    validation_0-logloss:0.18221    validation_1-logloss:0.20967
[18]    validation_0-logloss:0.18011    validation_1-logloss:0.20852
[19]    validation_0-logloss:0.17838    validation_1-logloss:0.20798
[20]    validation_0-logloss:0.17678    validation_1-logloss:0.20753
[21]    validation_0-logloss:0.17453    validation_1-logloss:0.20597
[22]    validation_0-logloss:0.17389    validation_1-logloss:0.20567
[23]    validation_0-logloss:0.17236    validation_1-logloss:0.20530
[24]    validation_0-logloss:0.17145    validation_1-logloss:0.20536
[25]    validation_0-logloss:0.17014    validation_1-logloss:0.20522
[26]    validation_0-logloss:0.16827    validation_1-logloss:0.20457
[27]    validation_0-logloss:0.16706    validation_1-logloss:0.20438
[28]    validation_0-logloss:0.16660    validation_1-logloss:0.20447
[29]    validation_0-logloss:0.16556    validation_1-logloss:0.20455
[30]    validation_0-logloss:0.16502    validation_1-logloss:0.20468
[31]    validation_0-logloss:0.16426    validation_1-logloss:0.20436
[32]    validation_0-logloss:0.16284    validation_1-logloss:0.20370
[33]    validation_0-logloss:0.16187    validation_1-logloss:0.20367
[34]    validation_0-logloss:0.16059    validation_1-logloss:0.20351
[35]    validation_0-logloss:0.15880    validation_1-logloss:0.20282
[36]    validation_0-logloss:0.15825    validation_1-logloss:0.20284
[37]    validation_0-logloss:0.15768    validation_1-logloss:0.20258
[38]    validation_0-logloss:0.15617    validation_1-logloss:0.20196
[39]    validation_0-logloss:0.15433    validation_1-logloss:0.20121
[40]    validation_0-logloss:0.15353    validation_1-logloss:0.20125
[41]    validation_0-logloss:0.15268    validation_1-logloss:0.20146
[42]    validation_0-logloss:0.15190    validation_1-logloss:0.20148
[43]    validation_0-logloss:0.15124    validation_1-logloss:0.20135
[44]    validation_0-logloss:0.15054    validation_1-logloss:0.20120
[45]    validation_0-logloss:0.14996    validation_1-logloss:0.20116
[46]    validation_0-logloss:0.14923    validation_1-logloss:0.20111
[47]    validation_0-logloss:0.14795    validation_1-logloss:0.20101
[48]    validation_0-logloss:0.14719    validation_1-logloss:0.20092
[49]    validation_0-logloss:0.14705    validation_1-logloss:0.20098
[50]    validation_0-logloss:0.14679    validation_1-logloss:0.20107
[51]    validation_0-logloss:0.14631    validation_1-logloss:0.20095
[52]    validation_0-logloss:0.14624    validation_1-logloss:0.20094
[53]    validation_0-logloss:0.14588    validation_1-logloss:0.20101
[54]    validation_0-logloss:0.14476    validation_1-logloss:0.20079
```

```
[55]    validation_0-logloss:0.14397    validation_1-logloss:0.20095
[56]    validation_0-logloss:0.14331    validation_1-logloss:0.20085
[57]    validation_0-logloss:0.14249    validation_1-logloss:0.20082
[58]    validation_0-logloss:0.14194    validation_1-logloss:0.20077
[59]    validation_0-logloss:0.14180    validation_1-logloss:0.20085
[60]    validation_0-logloss:0.14142    validation_1-logloss:0.20099
[61]    validation_0-logloss:0.14128    validation_1-logloss:0.20096
[62]    validation_0-logloss:0.14121    validation_1-logloss:0.20100
[63]    validation_0-logloss:0.14046    validation_1-logloss:0.20138
[64]    validation_0-logloss:0.13919    validation_1-logloss:0.20115
[65]    validation_0-logloss:0.13825    validation_1-logloss:0.20078
[66]    validation_0-logloss:0.13793    validation_1-logloss:0.20107
[67]    validation_0-logloss:0.13708    validation_1-logloss:0.20147
[68]    validation_0-logloss:0.13669    validation_1-logloss:0.20137
[69]    validation_0-logloss:0.13640    validation_1-logloss:0.20158
[70]    validation_0-logloss:0.13523    validation_1-logloss:0.20171
[71]    validation_0-logloss:0.13413    validation_1-logloss:0.20199
[72]    validation_0-logloss:0.13275    validation_1-logloss:0.20214
[73]    validation_0-logloss:0.13204    validation_1-logloss:0.20206
[74]    validation_0-logloss:0.13159    validation_1-logloss:0.20208
[75]    validation_0-logloss:0.13138    validation_1-logloss:0.20221
[76]    validation_0-logloss:0.13024    validation_1-logloss:0.20186
[77]    validation_0-logloss:0.13011    validation_1-logloss:0.20177
[78]    validation_0-logloss:0.12965    validation_1-logloss:0.20176
[79]    validation_0-logloss:0.12951    validation_1-logloss:0.20178
[80]    validation_0-logloss:0.12923    validation_1-logloss:0.20205
[81]    validation_0-logloss:0.12892    validation_1-logloss:0.20223
[82]    validation_0-logloss:0.12851    validation_1-logloss:0.20241
[83]    validation_0-logloss:0.12790    validation_1-logloss:0.20242
[84]    validation_0-logloss:0.12760    validation_1-logloss:0.20249
[85]    validation_0-logloss:0.12691    validation_1-logloss:0.20257
[86]    validation_0-logloss:0.12662    validation_1-logloss:0.20261
[87]    validation_0-logloss:0.12632    validation_1-logloss:0.20280
[88]    validation_0-logloss:0.12626    validation_1-logloss:0.20277
[89]    validation_0-logloss:0.12581    validation_1-logloss:0.20284
[90]    validation_0-logloss:0.12572    validation_1-logloss:0.20283
[91]    validation_0-logloss:0.12514    validation_1-logloss:0.20277
[92]    validation_0-logloss:0.12473    validation_1-logloss:0.20285
[93]    validation_0-logloss:0.12431    validation_1-logloss:0.20297
[94]    validation_0-logloss:0.12428    validation_1-logloss:0.20305
[95]    validation_0-logloss:0.12395    validation_1-logloss:0.20300
[96]    validation_0-logloss:0.12332    validation_1-logloss:0.20324
[97]    validation_0-logloss:0.12272    validation_1-logloss:0.20338
[98]    validation_0-logloss:0.12210    validation_1-logloss:0.20351
[99]    validation_0-logloss:0.12176    validation_1-logloss:0.20371
[100]   validation_0-logloss:0.12104    validation_1-logloss:0.20371
[101]   validation_0-logloss:0.12065    validation_1-logloss:0.20382
[102]   validation_0-logloss:0.12043    validation_1-logloss:0.20403
```

```
[103]   validation_0-logloss:0.12002    validation_1-logloss:0.20402
[104]   validation_0-logloss:0.11983    validation_1-logloss:0.20388
[105]   validation_0-logloss:0.11920    validation_1-logloss:0.20405
[106]   validation_0-logloss:0.11895    validation_1-logloss:0.20440
[107]   validation_0-logloss:0.11816    validation_1-logloss:0.20470
[108]   validation_0-logloss:0.11779    validation_1-logloss:0.20485
```

[195]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=50, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=1000,
              n_jobs=-1, num_parallel_tree=1, predictor='auto', random_state=42,
              reg_alpha=0, reg_lambda=1, …)
```

[196]:
```
ml_model.best_ntree_limit
```

[196]:
```
59
```

[197]:
```
ml_model = XGBClassifier(n_jobs=-1, n_estimators=59, random_state=42)
ml_model.fit(X_train, y_train)
```

[197]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=59,
              n_jobs=-1, num_parallel_tree=1, predictor='auto', random_state=42,
              reg_alpha=0, reg_lambda=1, …)
```

Checking the accuracy.

[198]:
```
from sklearn.metrics import accuracy_score

print('Train Accuracy: {}'.format(accuracy_score(y_train, ml_model.
  ↪predict(X_train))))
print('Validation Accuracy: {}'.format(accuracy_score(y_val, ml_model.
  ↪predict(X_val))))
```

```
Train Accuracy: 0.9421311656713116
Validation Accuracy: 0.9086586309852925
```

Checking the classification report.

[199]:
```
from sklearn.metrics import classification_report

print(classification_report(y_val, ml_model.predict(X_val)))
```

```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      7952
           1       0.66      0.49      0.57      1091

    accuracy                           0.91      9043
   macro avg       0.80      0.73      0.76      9043
weighted avg       0.90      0.91      0.90      9043
```

On class 1 statistics are very bad.

8 Dataset Balancing

It will be used SMOTE.

[200]:
```
from imblearn.over_sampling import SMOTE

smote = SMOTE(sampling_strategy='minority')

# Creating two new X and y with SMOTE

X_sm, y_sm = smote.fit_resample(X,y)
```

[201]:
```
X_sm
```

[201]:

	age	balance	day	duration	campaign	pdays	previous	\
0	1.266667	1.250000	-0.846154	0.375000	-0.500000	0.000000	0.0	
1	0.333333	-0.308997	-0.846154	-0.134259	-0.500000	0.000000	0.0	
2	-0.400000	-0.328909	-0.846154	-0.481481	-0.500000	0.000000	0.0	
3	0.533333	0.780236	-0.846154	-0.407407	-0.500000	0.000000	0.0	
4	-0.400000	-0.329646	-0.846154	0.083333	-0.500000	0.000000	0.0	
…	…	…	…	…	…	…	…	
79839	-0.525361	0.381973	-0.196907	0.528270	-0.500000	0.000000	0.0	
79840	-0.339454	0.152553	-0.923077	-0.128297	-0.500000	93.000000	5.0	
79841	0.604165	1.035969	-1.066875	5.832860	-0.500000	459.017038	1.0	
79842	0.057784	2.029909	-0.066674	3.125773	-0.405563	0.000000	0.0	
79843	-0.130841	0.012890	-1.000000	0.286518	-0.500000	0.000000	0.0	

	job_admin.	job_blue-collar	job_entrepreneur	…	month_jun	\
0	0.0	0.0	0.0	…	0.0	
1	0.0	0.0	0.0	…	0.0	
2	0.0	0.0	1.0	…	0.0	
3	0.0	1.0	0.0	…	0.0	

```
4             0.0              0.0               0.0  …        0.0
…             …                …                 …  …         …
79839         0.0              0.0               0.0  …        0.0
79840         0.0              0.0               0.0  …        0.0
79841         0.0              0.0               0.0  …        0.0
79842         0.0              1.0               0.0  …        0.0
79843         0.0              0.0               0.0  …        0.0

       month_mar  month_may  month_nov  month_oct  month_sep  \
0            0.0        1.0        0.0        0.0   0.000000
1            0.0        1.0        0.0        0.0   0.000000
2            0.0        1.0        0.0        0.0   0.000000
3            0.0        1.0        0.0        0.0   0.000000
4            0.0        1.0        0.0        0.0   0.000000
…            …          …          …          …         …
79839        0.0        1.0        0.0        0.0   0.000000
79840        0.0        0.0        0.0        0.0   0.000000
79841        0.0        0.0        0.0        0.0   0.994321
79842        0.0        1.0        0.0        0.0   0.000000
79843        0.0        0.0        0.0        0.0   0.000000

       poutcome_failure  poutcome_other  poutcome_success  poutcome_unknown
0                   0.0             0.0               0.0               1.0
1                   0.0             0.0               0.0               1.0
2                   0.0             0.0               0.0               1.0
3                   0.0             0.0               0.0               1.0
4                   0.0             0.0               0.0               1.0
…                   …               …                 …                 …
79839               0.0             0.0               0.0               1.0
79840               0.0             0.0               1.0               0.0
79841               1.0             0.0               0.0               0.0
79842               0.0             0.0               0.0               1.0
79843               0.0             0.0               0.0               1.0

[79844 rows x 51 columns]
```

[202]:
```
y_sm
```

[202]:
```
0        0
1        0
2        0
3        0
4        0
        ..
79839    1
79840    1
79841    1
```

```
79842    1
79843    1
Name: subscription, Length: 79844, dtype: int64
```

Seeing the difference.

[203]:
```
y.value_counts()
```

[203]:
```
0    39922
1     5289
Name: subscription, dtype: int64
```

[204]:
```
y_sm.value_counts()
```

[204]:
```
0    39922
1    39922
Name: subscription, dtype: int64
```

Now classes are balanced.

8.1 Splitting

[205]:
```
X_train_sm, X_val_sm, y_train_sm, y_val_sm = train_test_split(X_sm, y_sm,␣
↪stratify=y_sm ,test_size=0.20, random_state=42)
```

8.2 Creating the Model

[206]:
```
ml_model_sm = XGBClassifier(n_jobs=-1, n_estimators=1000,␣
↪early_stopping_rounds=50, random_state=42)
ml_model_sm.fit(X_train_sm, y_train_sm, eval_set=[(X_train_sm, y_train_sm),␣
↪(X_val_sm, y_val_sm)])
```

```
[0]    validation_0-logloss:0.54846    validation_1-logloss:0.54985
[1]    validation_0-logloss:0.46629    validation_1-logloss:0.46905
[2]    validation_0-logloss:0.41063    validation_1-logloss:0.41435
[3]    validation_0-logloss:0.36898    validation_1-logloss:0.37407
[4]    validation_0-logloss:0.34180    validation_1-logloss:0.34671
[5]    validation_0-logloss:0.31449    validation_1-logloss:0.31960
[6]    validation_0-logloss:0.29737    validation_1-logloss:0.30252
[7]    validation_0-logloss:0.28380    validation_1-logloss:0.28939
[8]    validation_0-logloss:0.26877    validation_1-logloss:0.27441
[9]    validation_0-logloss:0.25026    validation_1-logloss:0.25583
[10]   validation_0-logloss:0.23967    validation_1-logloss:0.24523
[11]   validation_0-logloss:0.22924    validation_1-logloss:0.23486
[12]   validation_0-logloss:0.22044    validation_1-logloss:0.22671
[13]   validation_0-logloss:0.21327    validation_1-logloss:0.21999
[14]   validation_0-logloss:0.20774    validation_1-logloss:0.21479
[15]   validation_0-logloss:0.19887    validation_1-logloss:0.20610
[16]   validation_0-logloss:0.19506    validation_1-logloss:0.20265
```

```
[17]    validation_0-logloss:0.18792    validation_1-logloss:0.19600
[18]    validation_0-logloss:0.18232    validation_1-logloss:0.19087
[19]    validation_0-logloss:0.17929    validation_1-logloss:0.18811
[20]    validation_0-logloss:0.17495    validation_1-logloss:0.18400
[21]    validation_0-logloss:0.17084    validation_1-logloss:0.18025
[22]    validation_0-logloss:0.16611    validation_1-logloss:0.17551
[23]    validation_0-logloss:0.16292    validation_1-logloss:0.17261
[24]    validation_0-logloss:0.16090    validation_1-logloss:0.17079
[25]    validation_0-logloss:0.15843    validation_1-logloss:0.16851
[26]    validation_0-logloss:0.15619    validation_1-logloss:0.16663
[27]    validation_0-logloss:0.15439    validation_1-logloss:0.16511
[28]    validation_0-logloss:0.15118    validation_1-logloss:0.16184
[29]    validation_0-logloss:0.14877    validation_1-logloss:0.15952
[30]    validation_0-logloss:0.14684    validation_1-logloss:0.15792
[31]    validation_0-logloss:0.14536    validation_1-logloss:0.15689
[32]    validation_0-logloss:0.14292    validation_1-logloss:0.15514
[33]    validation_0-logloss:0.14150    validation_1-logloss:0.15399
[34]    validation_0-logloss:0.13935    validation_1-logloss:0.15209
[35]    validation_0-logloss:0.13738    validation_1-logloss:0.15034
[36]    validation_0-logloss:0.13555    validation_1-logloss:0.14890
[37]    validation_0-logloss:0.13409    validation_1-logloss:0.14769
[38]    validation_0-logloss:0.13305    validation_1-logloss:0.14690
[39]    validation_0-logloss:0.13114    validation_1-logloss:0.14562
[40]    validation_0-logloss:0.12972    validation_1-logloss:0.14444
[41]    validation_0-logloss:0.12866    validation_1-logloss:0.14355
[42]    validation_0-logloss:0.12682    validation_1-logloss:0.14237
[43]    validation_0-logloss:0.12563    validation_1-logloss:0.14166
[44]    validation_0-logloss:0.12320    validation_1-logloss:0.13941
[45]    validation_0-logloss:0.12238    validation_1-logloss:0.13898
[46]    validation_0-logloss:0.12133    validation_1-logloss:0.13822
[47]    validation_0-logloss:0.12070    validation_1-logloss:0.13788
[48]    validation_0-logloss:0.11981    validation_1-logloss:0.13720
[49]    validation_0-logloss:0.11868    validation_1-logloss:0.13637
[50]    validation_0-logloss:0.11730    validation_1-logloss:0.13545
[51]    validation_0-logloss:0.11662    validation_1-logloss:0.13505
[52]    validation_0-logloss:0.11588    validation_1-logloss:0.13493
[53]    validation_0-logloss:0.11515    validation_1-logloss:0.13438
[54]    validation_0-logloss:0.11309    validation_1-logloss:0.13251
[55]    validation_0-logloss:0.11254    validation_1-logloss:0.13223
[56]    validation_0-logloss:0.11114    validation_1-logloss:0.13131
[57]    validation_0-logloss:0.11081    validation_1-logloss:0.13126
[58]    validation_0-logloss:0.10964    validation_1-logloss:0.13044
[59]    validation_0-logloss:0.10838    validation_1-logloss:0.12955
[60]    validation_0-logloss:0.10748    validation_1-logloss:0.12888
[61]    validation_0-logloss:0.10671    validation_1-logloss:0.12859
[62]    validation_0-logloss:0.10561    validation_1-logloss:0.12779
[63]    validation_0-logloss:0.10486    validation_1-logloss:0.12758
[64]    validation_0-logloss:0.10446    validation_1-logloss:0.12738
```

```
[65]    validation_0-logloss:0.10365    validation_1-logloss:0.12718
[66]    validation_0-logloss:0.10335    validation_1-logloss:0.12703
[67]    validation_0-logloss:0.10283    validation_1-logloss:0.12709
[68]    validation_0-logloss:0.10234    validation_1-logloss:0.12692
[69]    validation_0-logloss:0.10100    validation_1-logloss:0.12586
[70]    validation_0-logloss:0.10079    validation_1-logloss:0.12580
[71]    validation_0-logloss:0.10009    validation_1-logloss:0.12556
[72]    validation_0-logloss:0.09915    validation_1-logloss:0.12486
[73]    validation_0-logloss:0.09792    validation_1-logloss:0.12407
[74]    validation_0-logloss:0.09727    validation_1-logloss:0.12405
[75]    validation_0-logloss:0.09653    validation_1-logloss:0.12388
[76]    validation_0-logloss:0.09611    validation_1-logloss:0.12356
[77]    validation_0-logloss:0.09509    validation_1-logloss:0.12291
[78]    validation_0-logloss:0.09430    validation_1-logloss:0.12249
[79]    validation_0-logloss:0.09394    validation_1-logloss:0.12239
[80]    validation_0-logloss:0.09344    validation_1-logloss:0.12238
[81]    validation_0-logloss:0.09277    validation_1-logloss:0.12228
[82]    validation_0-logloss:0.09242    validation_1-logloss:0.12221
[83]    validation_0-logloss:0.09199    validation_1-logloss:0.12204
[84]    validation_0-logloss:0.09171    validation_1-logloss:0.12220
[85]    validation_0-logloss:0.09136    validation_1-logloss:0.12215
[86]    validation_0-logloss:0.09078    validation_1-logloss:0.12220
[87]    validation_0-logloss:0.09070    validation_1-logloss:0.12226
[88]    validation_0-logloss:0.09049    validation_1-logloss:0.12227
[89]    validation_0-logloss:0.09020    validation_1-logloss:0.12235
[90]    validation_0-logloss:0.08995    validation_1-logloss:0.12227
[91]    validation_0-logloss:0.08970    validation_1-logloss:0.12226
[92]    validation_0-logloss:0.08919    validation_1-logloss:0.12212
[93]    validation_0-logloss:0.08911    validation_1-logloss:0.12215
[94]    validation_0-logloss:0.08847    validation_1-logloss:0.12166
[95]    validation_0-logloss:0.08823    validation_1-logloss:0.12176
[96]    validation_0-logloss:0.08783    validation_1-logloss:0.12176
[97]    validation_0-logloss:0.08764    validation_1-logloss:0.12173
[98]    validation_0-logloss:0.08746    validation_1-logloss:0.12177
[99]    validation_0-logloss:0.08733    validation_1-logloss:0.12170
[100]   validation_0-logloss:0.08664    validation_1-logloss:0.12122
[101]   validation_0-logloss:0.08614    validation_1-logloss:0.12115
[102]   validation_0-logloss:0.08582    validation_1-logloss:0.12119
[103]   validation_0-logloss:0.08509    validation_1-logloss:0.12072
[104]   validation_0-logloss:0.08498    validation_1-logloss:0.12077
[105]   validation_0-logloss:0.08473    validation_1-logloss:0.12064
[106]   validation_0-logloss:0.08443    validation_1-logloss:0.12073
[107]   validation_0-logloss:0.08409    validation_1-logloss:0.12059
[108]   validation_0-logloss:0.08357    validation_1-logloss:0.12022
[109]   validation_0-logloss:0.08319    validation_1-logloss:0.12011
[110]   validation_0-logloss:0.08282    validation_1-logloss:0.12006
[111]   validation_0-logloss:0.08226    validation_1-logloss:0.11988
[112]   validation_0-logloss:0.08196    validation_1-logloss:0.11984
```

```
[113]   validation_0-logloss:0.08174    validation_1-logloss:0.11987
[114]   validation_0-logloss:0.08119    validation_1-logloss:0.11955
[115]   validation_0-logloss:0.08073    validation_1-logloss:0.11926
[116]   validation_0-logloss:0.08045    validation_1-logloss:0.11918
[117]   validation_0-logloss:0.08034    validation_1-logloss:0.11916
[118]   validation_0-logloss:0.07997    validation_1-logloss:0.11939
[119]   validation_0-logloss:0.07965    validation_1-logloss:0.11929
[120]   validation_0-logloss:0.07926    validation_1-logloss:0.11937
[121]   validation_0-logloss:0.07888    validation_1-logloss:0.11925
[122]   validation_0-logloss:0.07849    validation_1-logloss:0.11914
[123]   validation_0-logloss:0.07818    validation_1-logloss:0.11918
[124]   validation_0-logloss:0.07762    validation_1-logloss:0.11894
[125]   validation_0-logloss:0.07754    validation_1-logloss:0.11892
[126]   validation_0-logloss:0.07733    validation_1-logloss:0.11892
[127]   validation_0-logloss:0.07707    validation_1-logloss:0.11882
[128]   validation_0-logloss:0.07676    validation_1-logloss:0.11874
[129]   validation_0-logloss:0.07673    validation_1-logloss:0.11876
[130]   validation_0-logloss:0.07670    validation_1-logloss:0.11879
[131]   validation_0-logloss:0.07632    validation_1-logloss:0.11880
[132]   validation_0-logloss:0.07614    validation_1-logloss:0.11873
[133]   validation_0-logloss:0.07604    validation_1-logloss:0.11878
[134]   validation_0-logloss:0.07583    validation_1-logloss:0.11871
[135]   validation_0-logloss:0.07536    validation_1-logloss:0.11869
[136]   validation_0-logloss:0.07509    validation_1-logloss:0.11876
[137]   validation_0-logloss:0.07474    validation_1-logloss:0.11869
[138]   validation_0-logloss:0.07435    validation_1-logloss:0.11860
[139]   validation_0-logloss:0.07383    validation_1-logloss:0.11837
[140]   validation_0-logloss:0.07365    validation_1-logloss:0.11833
[141]   validation_0-logloss:0.07323    validation_1-logloss:0.11837
[142]   validation_0-logloss:0.07305    validation_1-logloss:0.11823
[143]   validation_0-logloss:0.07259    validation_1-logloss:0.11826
[144]   validation_0-logloss:0.07250    validation_1-logloss:0.11826
[145]   validation_0-logloss:0.07218    validation_1-logloss:0.11824
[146]   validation_0-logloss:0.07185    validation_1-logloss:0.11836
[147]   validation_0-logloss:0.07172    validation_1-logloss:0.11831
[148]   validation_0-logloss:0.07144    validation_1-logloss:0.11826
[149]   validation_0-logloss:0.07125    validation_1-logloss:0.11830
[150]   validation_0-logloss:0.07099    validation_1-logloss:0.11821
[151]   validation_0-logloss:0.07084    validation_1-logloss:0.11822
[152]   validation_0-logloss:0.07059    validation_1-logloss:0.11828
[153]   validation_0-logloss:0.07047    validation_1-logloss:0.11830
[154]   validation_0-logloss:0.07044    validation_1-logloss:0.11831
[155]   validation_0-logloss:0.07017    validation_1-logloss:0.11829
[156]   validation_0-logloss:0.07000    validation_1-logloss:0.11829
[157]   validation_0-logloss:0.06983    validation_1-logloss:0.11825
[158]   validation_0-logloss:0.06950    validation_1-logloss:0.11816
[159]   validation_0-logloss:0.06920    validation_1-logloss:0.11825
[160]   validation_0-logloss:0.06918    validation_1-logloss:0.11827
```

```
[161]   validation_0-logloss:0.06916    validation_1-logloss:0.11828
[162]   validation_0-logloss:0.06906    validation_1-logloss:0.11822
[163]   validation_0-logloss:0.06895    validation_1-logloss:0.11824
[164]   validation_0-logloss:0.06892    validation_1-logloss:0.11821
[165]   validation_0-logloss:0.06883    validation_1-logloss:0.11823
[166]   validation_0-logloss:0.06851    validation_1-logloss:0.11813
[167]   validation_0-logloss:0.06817    validation_1-logloss:0.11787
[168]   validation_0-logloss:0.06773    validation_1-logloss:0.11774
[169]   validation_0-logloss:0.06744    validation_1-logloss:0.11768
[170]   validation_0-logloss:0.06695    validation_1-logloss:0.11747
[171]   validation_0-logloss:0.06664    validation_1-logloss:0.11752
[172]   validation_0-logloss:0.06648    validation_1-logloss:0.11763
[173]   validation_0-logloss:0.06617    validation_1-logloss:0.11758
[174]   validation_0-logloss:0.06587    validation_1-logloss:0.11769
[175]   validation_0-logloss:0.06555    validation_1-logloss:0.11771
[176]   validation_0-logloss:0.06520    validation_1-logloss:0.11773
[177]   validation_0-logloss:0.06482    validation_1-logloss:0.11752
[178]   validation_0-logloss:0.06459    validation_1-logloss:0.11753
[179]   validation_0-logloss:0.06450    validation_1-logloss:0.11765
[180]   validation_0-logloss:0.06421    validation_1-logloss:0.11753
[181]   validation_0-logloss:0.06388    validation_1-logloss:0.11719
[182]   validation_0-logloss:0.06368    validation_1-logloss:0.11723
[183]   validation_0-logloss:0.06362    validation_1-logloss:0.11721
[184]   validation_0-logloss:0.06346    validation_1-logloss:0.11726
[185]   validation_0-logloss:0.06325    validation_1-logloss:0.11735
[186]   validation_0-logloss:0.06301    validation_1-logloss:0.11726
[187]   validation_0-logloss:0.06277    validation_1-logloss:0.11728
[188]   validation_0-logloss:0.06267    validation_1-logloss:0.11736
[189]   validation_0-logloss:0.06243    validation_1-logloss:0.11723
[190]   validation_0-logloss:0.06222    validation_1-logloss:0.11726
[191]   validation_0-logloss:0.06206    validation_1-logloss:0.11728
[192]   validation_0-logloss:0.06203    validation_1-logloss:0.11729
[193]   validation_0-logloss:0.06190    validation_1-logloss:0.11733
[194]   validation_0-logloss:0.06160    validation_1-logloss:0.11737
[195]   validation_0-logloss:0.06139    validation_1-logloss:0.11727
[196]   validation_0-logloss:0.06137    validation_1-logloss:0.11727
[197]   validation_0-logloss:0.06134    validation_1-logloss:0.11727
[198]   validation_0-logloss:0.06128    validation_1-logloss:0.11727
[199]   validation_0-logloss:0.06124    validation_1-logloss:0.11727
[200]   validation_0-logloss:0.06107    validation_1-logloss:0.11741
[201]   validation_0-logloss:0.06087    validation_1-logloss:0.11748
[202]   validation_0-logloss:0.06066    validation_1-logloss:0.11756
[203]   validation_0-logloss:0.06053    validation_1-logloss:0.11760
[204]   validation_0-logloss:0.06036    validation_1-logloss:0.11769
[205]   validation_0-logloss:0.06030    validation_1-logloss:0.11774
[206]   validation_0-logloss:0.06019    validation_1-logloss:0.11778
[207]   validation_0-logloss:0.05989    validation_1-logloss:0.11797
[208]   validation_0-logloss:0.05956    validation_1-logloss:0.11791
```

```
[209]   validation_0-logloss:0.05942    validation_1-logloss:0.11798
[210]   validation_0-logloss:0.05934    validation_1-logloss:0.11806
[211]   validation_0-logloss:0.05923    validation_1-logloss:0.11806
[212]   validation_0-logloss:0.05913    validation_1-logloss:0.11816
[213]   validation_0-logloss:0.05897    validation_1-logloss:0.11812
[214]   validation_0-logloss:0.05860    validation_1-logloss:0.11805
[215]   validation_0-logloss:0.05825    validation_1-logloss:0.11788
[216]   validation_0-logloss:0.05810    validation_1-logloss:0.11799
[217]   validation_0-logloss:0.05782    validation_1-logloss:0.11792
[218]   validation_0-logloss:0.05752    validation_1-logloss:0.11804
[219]   validation_0-logloss:0.05727    validation_1-logloss:0.11808
[220]   validation_0-logloss:0.05713    validation_1-logloss:0.11809
[221]   validation_0-logloss:0.05683    validation_1-logloss:0.11818
[222]   validation_0-logloss:0.05663    validation_1-logloss:0.11812
[223]   validation_0-logloss:0.05632    validation_1-logloss:0.11816
[224]   validation_0-logloss:0.05611    validation_1-logloss:0.11827
[225]   validation_0-logloss:0.05586    validation_1-logloss:0.11813
[226]   validation_0-logloss:0.05574    validation_1-logloss:0.11806
[227]   validation_0-logloss:0.05544    validation_1-logloss:0.11817
[228]   validation_0-logloss:0.05531    validation_1-logloss:0.11806
[229]   validation_0-logloss:0.05529    validation_1-logloss:0.11802
[230]   validation_0-logloss:0.05524    validation_1-logloss:0.11805
[231]   validation_0-logloss:0.05522    validation_1-logloss:0.11805
```

[206]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=50, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=1000,
              n_jobs=-1, num_parallel_tree=1, predictor='auto', random_state=42,
              reg_alpha=0, reg_lambda=1, …)
```

[207]:
```
ml_model_sm.best_ntree_limit
```

[207]:
```
182
```

[208]:
```
ml_model_sm = XGBClassifier(n_jobs=-1, n_estimators=183, random_state=42)
ml_model_sm.fit(X_train_sm, y_train_sm)
```

[208]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
```

```
max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
missing=nan, monotone_constraints='()', n_estimators=183,
n_jobs=-1, num_parallel_tree=1, predictor='auto', random_state=42,
reg_alpha=0, reg_lambda=1, …)
```

Checking the accuracy.

[209]:
```
print('Train Accuracy: {}'.format(accuracy_score(y_train_sm, ml_model_sm.
 ↪predict(X_train_sm))))
print('Validation Accuracy: {}'.format(accuracy_score(y_val_sm, ml_model_sm.
 ↪predict(X_val_sm))))
```

```
Train Accuracy: 0.980477495107632
Validation Accuracy: 0.9492767236520759
```

Checking the Classification Report.

[210]:
```
print(classification_report(y_val, ml_model.predict(X_val)))
print(classification_report(y_val_sm, ml_model_sm.predict(X_val_sm)))
```

```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      7952
           1       0.66      0.49      0.57      1091

    accuracy                           0.91      9043
   macro avg       0.80      0.73      0.76      9043
weighted avg       0.90      0.91      0.90      9043

              precision    recall  f1-score   support

           0       0.94      0.96      0.95      7985
           1       0.96      0.94      0.95      7984

    accuracy                           0.95     15969
   macro avg       0.95      0.95      0.95     15969
weighted avg       0.95      0.95      0.95     15969
```

The performance has improved a lot.