instacartfe

April 20, 2021

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt

import numpy as np
import gc
gc.enable()
from IPython.display import display
from xgboost import XGBClassifier
from sklearn import metrics, model_selection
import re
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix ,classification_report,auc␣
↪,precision_recall_curve ,average_precision_score, roc_auc_score,roc_curve,auc
from sklearn.metrics import f1_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import f1_score
from sklearn.preprocessing import OneHotEncoder
import xgboost as xgb
from sklearn.feature_extraction.text import TfidfVectorizer
import os
```

[2]:
```
from functools import partial
```

[3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[4]:
```
import seaborn as sns
```

[5]:
```
from hyperopt import  Trials, fmin, hp, tpe
from sklearn.model_selection import StratifiedKFold
from hyperopt.pyll.base import scope
```

[ ]:
```
# path ='../input/instacart-market-basket-analysis/'
path ="../"
```

0.1 Reading datasets

[ ]:
```
dtype = {
    "order_id": 'uint32',
    "user_id": 'uint32',
    "eval_set": 'category',
    "order_number": 'uint8',
    "order_dow": 'uint8',
    "order_hour_of_day": 'uint8',
    "days_since_prior_order": 'float16'
}
order = pd.read_csv(path+"orders.csv", dtype=dtype )
# filling days_since_prior_order as 30 as choosing recent value like 0 means␣
 ↪people has bought product recently
order.days_since_prior_order=order.days_since_prior_order.fillna(30)
order.head()
order.eval_set.replace({'prior':1 , 'train':2 , 'test':3} , inplace =True)
order
```

[ ]:

	order_id	user_id	eval_set	order_number	order_dow	\
0	2539329	1	1	1	2	
1	2398795	1	1	2	3	
2	473747	1	1	3	3	
3	2254736	1	1	4	4	
4	431534	1	1	5	4	
…	…	…	…	…	…	
3421078	2266710	206209	1	10	5	
3421079	1854736	206209	1	11	4	
3421080	626363	206209	1	12	1	
3421081	2977660	206209	1	13	1	
3421082	272231	206209	2	14	6	

	order_hour_of_day	days_since_prior_order
0	8	30.0
1	7	15.0
2	12	21.0
3	7	29.0
4	15	28.0
…	…	…
3421078	18	29.0
3421079	10	30.0
3421080	12	18.0
3421081	12	7.0
3421082	14	30.0

```
[3421083 rows x 7 columns]
```

[ ]:
```
dtype = {
    "order_id": 'uint32',
    "product_id": 'uint32',
    "add_to_cart_order": 'uint8',
    "reordered": 'uint8'
}
order_product_train = pd.read_csv(path+"order_products__train.csv" ,␣
 ↪dtype=dtype  )
order_product_train.head(5)
```

[ ]:

	order_id	product_id	add_to_cart_order	reordered
0	1	49302	1	1
1	1	11109	2	1
2	1	10246	3	0
3	1	49683	4	0
4	1	43633	5	1

[ ]:
```
dtype = {
    "order_id": 'uint32',
    "product_id": 'uint32',
    "add_to_cart_order": 'uint8',
    "reordered": 'uint8'
}
order_product_prior = pd.read_csv(path+"order_products__prior.csv"  ,␣
 ↪dtype=dtype )
order_product_prior.head(5)
```

[ ]:

	order_id	product_id	add_to_cart_order	reordered
0	2	33120	1	1
1	2	28985	2	1
2	2	9327	3	0
3	2	45918	4	1
4	2	30035	5	0

[ ]:
```
dtype = {
    "product_id": 'uint32',
    "product_name": 'category',
    "aisle_id": 'uint16',
    "department_id": 'uint16'
}
product = pd.read_csv(path+"products.csv", dtype=dtype  )
product.head(5)
```

[ ]:
```
   product_id                                       product_name  aisle_id  \
0           1                       Chocolate Sandwich Cookies        61
1           2                                 All-Seasons Salt       104
2           3             Robust Golden Unsweetened Oolong Tea        94
3           4  Smart Ones Classic Favorites Mini Rigatoni Wit…        38
4           5                          Green Chile Anytime Sauce         5

   department_id
0             19
1             13
2              7
3              1
4             13
```

Merging order Prior with order , to get product_id and reordered status of a order

- order_Prior contains order_no and product_no information
- order contains only order number

[ ]:
```
order_product_prior_ = order_product_prior.merge(order ,on ='order_id' , how
↪='inner')
order_product_prior_
```

[ ]:
```
          order_id  product_id  add_to_cart_order  reordered  user_id  \
0                2       33120                  1          1   202279
1                2       28985                  2          1   202279
2                2        9327                  3          0   202279
3                2       45918                  4          1   202279
4                2       30035                  5          0   202279
…              …         …                  …          …       …
32434484   3421083       39678                  6          1    25247
32434485   3421083       11352                  7          0    25247
32434486   3421083        4600                  8          0    25247
32434487   3421083       24852                  9          1    25247
32434488   3421083        5020                 10          1    25247

          eval_set  order_number  order_dow  order_hour_of_day  \
0                1             3          5                  9
1                1             3          5                  9
2                1             3          5                  9
3                1             3          5                  9
4                1             3          5                  9
…              …           …          …                  …
32434484         1            24          2                  6
32434485         1            24          2                  6
32434486         1            24          2                  6
32434487         1            24          2                  6
```

```
32434488         1            24          2                  6

          days_since_prior_order
0                            8.0
1                            8.0
2                            8.0
3                            8.0
4                            8.0
…                            …
32434484                    21.0
32434485                    21.0
32434486                    21.0
32434487                    21.0
32434488                    21.0

[32434489 rows x 10 columns]
```

Features using user_id

- Each order number is associated with one order number and order_id
- (order_id - order_number) –> [(2-3),(3421083 ,24)]
- Last order number is randomly split to train and test sets
- Order_number - max number of orders by user - last order no
- ttl_cnt_product_user - total order made by user

[ ]:
```
user_feature =order_product_prior_.groupby('user_id')['order_number'].
 ↪agg(['max' , 'count' ]).reset_index()
user_feature.rename(columns={'max':'order_number' ,'count':
 ↪'ttl_cnt_product_user'} , inplace =True)
display(user_feature)
```

```
        user_id  order_number  ttl_cnt_product_user
0             1            10                    59
1             2            14                   195
2             3            12                    88
3             4             5                    18
4             5             4                    37
…           …             …                     …
206204   206205             3                    32
206205   206206            67                   285
206206   206207            16                   223
206207   206208            49                   677
206208   206209            13                   129

[206209 rows x 3 columns]
```

Average number of product bought by user per order

- get count of product in an order by user than find mean count of product bought by user
- Avg_no_prod_perOrder -Average number of product bought by user

[ ]:
```
# get count of product in an order by user
user_prod_cnt=order_product_prior_[['user_id','order_id','product_id']].
 ↪groupby(['user_id','order_id'])['product_id'].agg(['count']).reset_index()
display(user_prod_cnt)
# get mean count of product bought by user
user_prod_avg=user_prod_cnt.groupby('user_id')['count'].agg('mean').
 ↪reset_index()
user_prod_avg.rename(columns={'count':'Avg_no_prod_perOrder'} , inplace =True)
display(user_prod_avg)
```

	user_id	order_id	count
0	1	431534	8
1	1	473747	5
2	1	550135	5
3	1	2254736	5
4	1	2295261	6
…	…	…	…
3214869	206209	2307371	3
3214870	206209	2558525	3
3214871	206209	2977660	9
3214872	206209	3154581	13
3214873	206209	3186442	2

[3214874 rows x 3 columns]

	user_id	Avg_no_prod_perOrder
0	1	5.900000
1	2	13.928571
2	3	7.333333
3	4	3.600000
4	5	9.250000
…	…	…
206204	206205	10.666667
206205	206206	4.253731
206206	206207	13.937500
206207	206208	13.816327
206208	206209	9.923077

[206209 rows x 2 columns]

- days_since_prior_order :Average no of days since user made prior order

[ ]:
```
usr_avg_dspo=order_product_prior_[['user_id' ,'days_since_prior_order']].
 ↪groupby('user_id')['days_since_prior_order'].agg('mean').reset_index()
display(usr_avg_dspo)
```

```
        user_id  days_since_prior_order
0             1                21.078125
1             2                16.906250
2             3                13.593750
3             4                18.609375
4             5                19.109375
…           …                      …
206204   206205                25.625000
206205   206206                 4.406250
206206   206207                16.500000
206207   206208                 7.843750
206208   206209                21.250000

[206209 rows x 2 columns]
```

- Take mode of dow and get which day of week most order is made
- Day of week on which user makes most order

[ ]:
```
usr_dow_max=order_product_prior_[['user_id' ,'order_dow']].
 ↪groupby('user_id')['order_dow'].agg([lambda x:x.value_counts().index[0]]).
 ↪reset_index()
usr_dow_max.rename(columns={'<lambda>':'max_dow'} , inplace =True)
display(usr_dow_max)
```

```
        user_id  max_dow
0             1        4
1             2        2
2             3        0
3             4        4
4             5        3
…           …      …
206204   206205        4
206205   206206        0
206206   206207        1
206207   206208        2
206208   206209        1

[206209 rows x 2 columns]
```

- Take mode of doh and get which day of hour most order is made
- which hour of day user makes most order
- max_hour_of_day

[ ]:
```
usr_hod_max=order_product_prior_[['user_id' ,'order_hour_of_day']].
 ↪groupby('user_id')['order_hour_of_day'].agg([lambda x:x.value_counts().
 ↪index[0]]).reset_index()
usr_hod_max.rename(columns={'<lambda>':'max_hour_of_day'} , inplace =True)
display(usr_hod_max)
```

```
        user_id  max_hour_of_day
0             1                7
1             2                9
2             3               16
3             4               15
4             5               18
…           …                 …
206204   206205               12
206205   206206               18
206206   206207               12
206207   206208               15
206208   206209               12

[206209 rows x 2 columns]
```

- How often does user reorder
- usr_ro_ratio : User reorder ratio

[ ]:
```
usr_ro_ratio=order_product_prior_[['user_id' ,'reordered']].
 ↪groupby('user_id')['reordered'].agg('mean').reset_index()
usr_ro_ratio.rename(columns={'reordered':'usr_ro_ratio'} , inplace =True)
display(usr_ro_ratio)
```

```
        user_id  usr_ro_ratio
0             1      0.694915
1             2      0.476923
2             3      0.625000
3             4      0.055556
4             5      0.378378
…           …            …
206204   206205      0.250000
206205   206206      0.473684
206206   206207      0.587444
206207   206208      0.707533
206208   206209      0.472868

[206209 rows x 2 columns]
```

- Merging all feature created

[ ]:
```
user_feature=user_feature.merge(user_prod_avg  , on='user_id' , how ='left')
user_feature=user_feature.merge(usr_avg_dspo  , on='user_id' , how ='left')
user_feature=user_feature.merge(usr_hod_max  , on='user_id' , how ='left')
user_feature=user_feature.merge(usr_ro_ratio  , on='user_id' , how ='left')
user_feature=user_feature.merge(usr_dow_max  , on='user_id' , how ='left')
display(user_feature)
del [user_prod_avg ,usr_avg_dspo ,usr_hod_max ,usr_dow_max,usr_ro_ratio]
gc.collect()
```

```
        user_id  order_number  ttl_cnt_product_user  Avg_no_prod_perOrder  \
0             1            10                    59              5.900000
1             2            14                   195             13.928571
2             3            12                    88              7.333333
3             4             5                    18              3.600000
4             5             4                    37              9.250000
…           …             …                     …                     …
206204   206205             3                    32             10.666667
206205   206206            67                   285              4.253731
206206   206207            16                   223             13.937500
206207   206208            49                   677             13.816327
206208   206209            13                   129              9.923077

        days_since_prior_order  max_hour_of_day  usr_ro_ratio  max_dow
0                    21.078125                7      0.694915        4
1                    16.906250                9      0.476923        2
2                    13.593750               16      0.625000        0
3                    18.609375               15      0.055556        4
4                    19.109375               18      0.378378        3
…                          …                …             …       …
206204               25.625000               12      0.250000        4
206205                4.406250               18      0.473684        0
206206               16.500000               12      0.587444        1
206207                7.843750               15      0.707533        2
206208               21.250000               12      0.472868        1

[206209 rows x 8 columns]
```

[ ]: 23

- Creating Product Features
- product_name_length
- isglutenfree
- isOrganic

[ ]:
```
product_fe = product.copy()

#Product name length
product_fe['product_name_length'] = product_fe['product_name'].str.len()


def isglutenfree(pname):
    """
    pname: Product name string
    This function matches is product name contains gluten free in string return␣
 ↪true else false
    """
    if re.match(r'(.*gluten.*free)' ,pname ,re.I):
```

```
        return 1
    else :
        return 0

product_fe['isglutenfree']=product_fe['product_name'].apply(isglutenfree)

def isOrganic(pname):
    """
    pname: Product name string
    This function matches is product name contains organic in string return␣
 ↪true else false
    """
    if re.match(r'(.*organic.*)' ,pname ,re.I):
        return 1
    else :
        return 0

product_fe['is_organic']=product_fe['product_name'].apply(isOrganic)
product_fe.drop(columns=['product_name' ,'aisle_id'] , inplace =True)

print("Table product_fe")
display(product_fe)
```

```
Table product_fe
       product_id  department_id  product_name_length  isglutenfree  \
0               1             19                   26             0
1               2             13                   16             0
2               3              7                   36             0
3               4              1                   65             0
4               5             13                   25             0
…             …              …                    …             …
49683       49684              5                   41             0
49684       49685              1                   34             0
49685       49686              3                   16             0
49686       49687              8                   42             0
49687       49688             11                   22             0

       is_organic
0               0
1               0
2               0
3               0
4               0
…             …
49683           0
49684           0
49685           0
```

```
49686           0
49687           0

[49688 rows x 5 columns]
```

[ ]:
```
del [product,order_product_prior,order_product_prior_,order_product_train ,order ]
gc.collect()
```

[ ]:
```
20
```

[ ]:
```
# reading order and product prior
dtype = {
    "order_id": 'uint32',
    "user_id": 'uint32',
    "eval_set": 'category',
    "order_number": 'uint8',
    "order_dow": 'uint8',
    "order_hour_of_day": 'uint8',
    "days_since_prior_order": 'float16'
}
order = pd.read_csv(path+"orders.csv", dtype=dtype ,usecols=['order_id' ,"user_id" ,'order_number'] )
print("Order Table : ")
display(order)


dtype = {
    "order_id": 'uint32',
    "product_id": 'uint32',
    "add_to_cart_order": 'uint8',
    "reordered": 'uint8',

}

order_product_prior = pd.read_csv(path+"order_products__prior.csv"  ,dtype=dtype )
print("Order product Prior Table : ")
display(order_product_prior)
```

```
Order Table :
```

	order_id	user_id	order_number
0	2539329	1	1
1	2398795	1	2
2	473747	1	3
3	2254736	1	4
4	431534	1	5

```
…          …       …            …
3421078   2266710   206209            10
3421079   1854736   206209            11
3421080    626363   206209            12
3421081   2977660   206209            13
3421082    272231   206209            14

[3421083 rows x 3 columns]
```

```
Order product Prior Table :

          order_id  product_id  add_to_cart_order  reordered
0                2       33120                  1          1
1                2       28985                  2          1
2                2        9327                  3          0
3                2       45918                  4          1
4                2       30035                  5          0
…              …          …                  …          …
32434484   3421083       39678                  6          1
32434485   3421083       11352                  7          0
32434486   3421083        4600                  8          0
32434487   3421083       24852                  9          1
32434488   3421083        5020                 10          1

[32434489 rows x 4 columns]
```

[ ]:
```
# merging product prior with order to get user_id
order_product_prior_ = pd.merge(order_product_prior , order , how ="inner" ,␣
 ↪on="order_id")
# Getting the reorder status of last ordered product by user
prior_last_order = order_product_prior_.groupby("user_id")["order_number"].
 ↪aggregate("max").reset_index()
# merging product prior to get  user_id
# lastest order number can belong to train or test points
order_prior_latest=pd.merge(order_product_prior_ ,prior_last_order , on␣
 ↪=['user_id' ,'order_number'] , how='inner' )
order_prior_latest=order_prior_latest[["user_id", "product_id", "reordered"]]
# reordered_last if last order was ordered or not
order_prior_latest.rename(columns={'reordered' :'reordered_last'} , inplace␣
 ↪=True)
display(order_prior_latest)
```

```
         user_id  product_id  reordered_last
0          59897        9755               1
1          59897       31487               0
2          59897       37510               1
3          59897       14576               1
4          59897       22105               0
```

```
…          …          …              …
2139783    79937       23400              0
2139784    79937       39812              0
2139785    79937       15795              0
2139786   103510       49187              0
2139787   103510       20126              0

[2139788 rows x 3 columns]
```

- ur_pr_count : sum of product reordered by user
- ur_pr_reordered : count of product ordered by user

[ ]:
```
# getting count of product bought and its reorder count
order_product_prior_=order_product_prior_.groupby(['user_id'␣
 ↪,'product_id'])['reordered'].agg(['count' ,'sum']).reset_index()
order_product_prior_.rename(columns={'count' :'ur_pr_count' , 'sum' :
 ↪'ur_pr_reordered'} ,inplace =True)

order_product_prior_=pd.merge(order_product_prior_,order_prior_latest ,on=␣
 ↪['user_id' ,'product_id'] , how ='left')
display(order_product_prior_)
```

```
          user_id  product_id  ur_pr_count  ur_pr_reordered  reordered_last
0               1         196           10                9             1.0
1               1       10258            9                8             1.0
2               1       10326            1                0             NaN
3               1       12427           10                9             1.0
4               1       13032            3                2             1.0
…             …          …            …                …                …
13307948   206209       43961            3                2             NaN
13307949   206209       44325            1                0             NaN
13307950   206209       48370            1                0             NaN
13307951   206209       48697            1                0             NaN
13307952   206209       48742            2                1             NaN

[13307953 rows x 5 columns]
```

Reading order product train

- This will let us know product_id for train data

[ ]:
```
#  droping order_number
order.drop(columns =['order_number'] , inplace =True)

# reading order product prior
dtype = {
    "order_id": 'uint32',
    "product_id": 'uint32',
```

```
    "add_to_cart_order": 'uint8',
    "reordered": 'uint8',

}

order_product_train = pd.read_csv(path+"order_products__train.csv"  ,␣
 ↪dtype=dtype  , usecols=['order_id'])

# grouping by order_id to get order number of product_id in train
order_product_train=order_product_train.groupby('order_id').agg('count').
 ↪reset_index()

print("Order product train  Table : ")
display(order_product_train)
```

```
Order product train  Table :
```

	order_id
0	1
1	36
2	38
3	96
4	98
…	…
131204	3421049
131205	3421056
131206	3421058
131207	3421063
131208	3421070

```
[131209 rows x 1 columns]
```

0.1.1 Now getting order_id and user id for train and test data

- For test data order using submission csv
- For train data using order_products___train
- Later we will merge with order_product_prior_ to get specific feature for test and train data

[ ]:
```
# reading test data
dtype = {
    "order_id": 'uint32'
}

# test =pd.read_csv("../input/pcamarket-analysis/sample_submission.csv" ,dtype␣
 ↪=dtype, usecols=['order_id'])
test =pd.read_csv("../sample_submission.csv" ,dtype =dtype,␣
 ↪usecols=['order_id'])
```

```
# merging with order to get user_id, order_id
train=pd.merge(order_product_train ,order , how ='inner' , on =['order_id'] )
# merging with order to get user_id
test=pd.merge(test ,order , how ='inner' , on =['order_id'] )

print("Train table ")
display(train)
print("Test table ")
display(test)
```

```
Train table
```

	order_id	user_id
0	1	112108
1	36	79431
2	38	42756
3	96	17227
4	98	56463
…	…	…
131204	3421049	189544
131205	3421056	83898
131206	3421058	136952
131207	3421063	169679
131208	3421070	139822

```
[131209 rows x 2 columns]
```

```
Test table
```

	order_id	user_id
0	17	36855
1	34	35220
2	137	187107
3	182	115892
4	257	35581
…	…	…
74995	3420740	195822
74996	3420877	15955
74997	3420888	162374
74998	3420989	109358
74999	3421054	153585

```
[75000 rows x 2 columns]
```

0.1.2 train and test table

```
[ ]: train =pd.merge(train ,order_product_prior_ , how='inner', on ='user_id' )
test =pd.merge(test ,order_product_prior_ , how='inner', on ='user_id' )
print("Train table")
display(train)
print("Test table")
display(test)
del [order_product_prior_ ,order_prior_latest ,prior_last_order]
gc.collect()
```

```
Train table
```

	order_id	user_id	product_id	ur_pr_count	ur_pr_reordered	\
0	1	112108	2067	1	0	
1	1	112108	5707	2	1	
2	1	112108	11109	2	1	
3	1	112108	14947	3	2	
4	1	112108	22035	2	1	
…	…	…	…	…	…	
8474656	3421070	139822	31506	1	0	
8474657	3421070	139822	34035	1	0	
8474658	3421070	139822	35347	1	0	
8474659	3421070	139822	35951	5	4	
8474660	3421070	139822	46149	9	8	

	reordered_last
0	NaN
1	1.0
2	NaN
3	1.0
4	NaN
…	…
8474656	NaN
8474657	NaN
8474658	NaN
8474659	NaN
8474660	1.0

```
[8474661 rows x 6 columns]
```

```
Test table
```

	order_id	user_id	product_id	ur_pr_count	ur_pr_reordered	\
0	17	36855	1283	1	0	
1	17	36855	6291	1	0	
2	17	36855	7035	1	0	
3	17	36855	11494	1	0	
4	17	36855	13107	3	2	

```
…           …       …          …           …                …
4833287   3421054   153585       46397           1                0
4833288   3421054   153585       47145           1                0
4833289   3421054   153585       47226           1                0
4833290   3421054   153585       48790           2                1
4833291   3421054   153585       48808           1                0

         reordered_last
0                   NaN
1                   NaN
2                   NaN
3                   NaN
4                   1.0
…                   …
4833287             NaN
4833288             NaN
4833289             NaN
4833290             NaN
4833291             NaN

[4833292 rows x 6 columns]
```

[ ]:
```
20
```

[ ]:
```
# reading product table
dtype = {
    "product_id": 'uint32',
    "product_name": 'category',
    "aisle_id": 'uint16',
    "department_id": 'uint16',

}

product = pd.read_csv(path+"products.csv", dtype=dtype  )
print("Product table")
display(product)
```

```
Product table

       product_id                                          product_name  \
0               1                          Chocolate Sandwich Cookies
1               2                                    All-Seasons Salt
2               3                Robust Golden Unsweetened Oolong Tea
3               4  Smart Ones Classic Favorites Mini Rigatoni Wit…
4               5                             Green Chile Anytime Sauce
…           …                                                 …
49683       49684          Vodka, Triple Distilled, Twist of Vanilla
49684       49685                  En Croute Roast Hazelnut Cranberry
```

```
49685       49686                                  Artisan Baguette
49686       49687          Smartblend Healthy Metabolism Dry Cat Food
49687       49688                                Fresh Foaming Cleanser

       aisle_id  department_id
0            61             19
1           104             13
2            94              7
3            38              1
4             5             13
…           …              …
49683      124              5
49684       42              1
49685      112              3
49686       41              8
49687       73             11

[49688 rows x 4 columns]
```

[ ]:
```
# merging train and test dataset with product to get product name
train=pd.merge(train , product , how ='inner' , on ='product_id')

print("Train table")
display(train)
test=pd.merge(test , product , how ='inner' , on ='product_id')

print("Test table")
display(test)
del product
gc.collect()
```

```
Train table

         order_id  user_id  product_id  ur_pr_count  ur_pr_reordered  \
0               1   112108        2067            1                0
1              96    17227        2067            1                0
2            3243   206024        2067            1                0
3           12950    11456        2067            1                0
4           17683   177724        2067            1                0
…             …       …          …              …                …
8474656   3417483    88512       13705            1                0
8474657   3418547   187144       22747            1                0
8474658   3418573   117202       25184            2                1
8474659   3419273    47713       38623            2                1
8474660   3420021    64216        2208            1                0

         reordered_last                                        product_name  \
0                   NaN  Plus Cranberry Almond + Antioxidants with Maca…
```

```
1                  NaN  Plus Cranberry Almond + Antioxidants with Maca…
2                  NaN  Plus Cranberry Almond + Antioxidants with Maca…
3                  0.0  Plus Cranberry Almond + Antioxidants with Maca…
4                  NaN  Plus Cranberry Almond + Antioxidants with Maca…
…                  …                                                 …
8474656            NaN                                    Yoga Bath Soak
8474657            NaN                  Vanilla Bean Sheep Milk Ice Cream
8474658            NaN                          PeroxiClear Lens Solution
8474659            NaN                            Wild Rice, 100% Natural
8474660            0.0                               Granola With Raisins

         aisle_id  department_id
0               3             19
1               3             19
2               3             19
3               3             19
4               3             19
…             …             …
8474656       109             11
8474657        37              1
8474658        44             11
8474659         4              9
8474660       121             14

[8474661 rows x 9 columns]
```

Test table

```
         order_id  user_id  product_id  ur_pr_count  ur_pr_reordered  \
0              17    36855        1283            1                0
1          657743    16994        1283            1                0
2          834223    92628        1283            1                0
3         1446886    75870        1283            1                0
4         1518981   118458        1283            3                2
…           …       …          …           …              …
4833287   3418159   104905       24836            1                0
4833288   3418520    88900        8102            2                1
4833289   3418520    88900       41872            2                1
4833290   3419574     1743       18920            1                0
4833291   3420740   195822       41431            1                0

         reordered_last                            product_name  aisle_id  \
0                  NaN  Organic Edamame & Mung Bean Fettuccine       100
1                  NaN  Organic Edamame & Mung Bean Fettuccine       100
2                  NaN  Organic Edamame & Mung Bean Fettuccine       100
3                  NaN  Organic Edamame & Mung Bean Fettuccine       100
4                  NaN  Organic Edamame & Mung Bean Fettuccine       100
…                  …                                       …       …
4833287            NaN          Cat Litter, Scoopable, Scented        41
```

```
4833288             NaN           Naty Diapers Size 1, 8-14 lbs        56
4833289             NaN           Herbal Tea, Organic Chamomile        94
4833290             NaN                       Turtle Sundae Mix       103
4833291             NaN      Strip-Free Hot Wax for Brow & Face        73

         department_id
0                   21
1                   21
2                   21
3                   21
4                   21
…                  …
4833287              8
4833288             18
4833289              7
4833290             19
4833291             11

[4833292 rows x 9 columns]
```

[ ]:
```
0
```

0.1.3 reading order_products___train to get reorder status for each entries

[ ]:
```
dtype = {
    "order_id": 'uint32',
    "product_id": 'uint32',
    "add_to_cart_order": 'uint8',
    "reordered": 'uint8',

}

train_target = pd.read_csv(path+"order_products__train.csv" , dtype=dtype␣
 ↪,usecols=['order_id' ,'product_id','reordered'] )
print("train_target table:")
display(train_target)
```

```
train_target table:

         order_id  product_id  reordered
0               1       49302          1
1               1       11109          1
2               1       10246          0
3               1       49683          0
4               1       43633          1
…             …          …         …
1384612   3421063       14233          1
1384613   3421063       35548          1
```

```
1384614   3421070       35951          1
1384615   3421070       16953          1
1384616   3421070        4724          1

[1384617 rows x 3 columns]
```

[ ]:
```
train_target = pd.merge(train_target , order , on ='order_id' , how ='inner')
train_target =train_target[["user_id", "product_id", "reordered"]]
print("train_target table:")
display(train_target)
```

```
train_target table:

         user_id  product_id  reordered
0         112108       49302          1
1         112108       11109          1
2         112108       10246          0
3         112108       49683          0
4         112108       43633          1
…           …          …          …
1384612   169679       14233          1
1384613   169679       35548          1
1384614   139822       35951          1
1384615   139822       16953          1
1384616   139822        4724          1

[1384617 rows x 3 columns]
```

[ ]:
```
train=pd.merge(train , train_target , how='left' , on=["user_id", "product_id"])
train['reordered'].fillna(0 , inplace =True)

print("Train table:")
display(train_target)

del train_target
gc.collect()
```

```
Train table:

         user_id  product_id  reordered
0         112108       49302          1
1         112108       11109          1
2         112108       10246          0
3         112108       49683          0
4         112108       43633          1
…           …          …          …
1384612   169679       14233          1
1384613   169679       35548          1
1384614   139822       35951          1
```

```
1384615   139822       16953          1
1384616   139822        4724          1

[1384617 rows x 3 columns]
```

[ ]:
```
60
```

0.1.4 Prepare dataset

we have no idea if product is reordered or not for many product

- 1- reordered
- 0- not reordered
- 3 - no idea

[ ]:
```
train.reordered_last.fillna(3,inplace =True)
test.reordered_last.fillna(3,inplace =True)
```

[ ]:
```
# droping product as we have product_name
train.drop(columns='product_name' , inplace =True)
test.drop(columns='product_name' , inplace =True)
```

0.1.5 merging feature engineered columns

[ ]:
```
# merge train and test  with user_feature
train=train.merge(user_feature  , on='user_id' , how='left')
# droping department_id as it is redundant
product_fe.drop(columns='department_id' , inplace =True)
# merge train and test  with product_fe
train=train.merge(product_fe  , on='product_id' , how='left')
```

[ ]:
```
train
```

[ ]:
```
         order_id  user_id  product_id  ur_pr_count  ur_pr_reordered  \
0               1   112108        2067            1                0
1              96    17227        2067            1                0
2            3243   206024        2067            1                0
3           12950    11456        2067            1                0
4           17683   177724        2067            1                0
…             …       …           …              …                …
8474656   3417483    88512       13705            1                0
8474657   3418547   187144       22747            1                0
8474658   3418573   117202       25184            2                1
8474659   3419273    47713       38623            2                1
8474660   3420021    64216        2208            1                0

         reordered_last  aisle_id  department_id  reordered  order_number  \
0                   3.0         3             19        0.0             3
```

```
1                   3.0         3             19        0.0             6
2                   3.0         3             19        0.0            42
3                   0.0         3             19        0.0             9
4                   3.0         3             19        0.0            29
…                   …           …             …         …               …
8474656             3.0       109             11        0.0             4
8474657             3.0        37              1        0.0            49
8474658             3.0        44             11        0.0             7
8474659             3.0         4              9        0.0             6
8474660             0.0       121             14        0.0             3

         ttl_cnt_product_user  Avg_no_prod_perOrder  days_since_prior_order  \
0                          21              7.000000               17.671875
1                          43              7.166667               21.531250
2                         335              7.976190                6.648438
3                         168             18.666667               12.203125
4                         396             13.655172               14.695312
…                         …                 …                     …
8474656                    39              9.750000               23.796875
8474657                   890             18.163265                5.964844
8474658                    31              4.428571               22.453125
8474659                   109             18.166667               12.367188
8474660                     8              2.666667               30.000000

         max_hour_of_day  usr_ro_ratio  max_dow  product_name_length  \
0                     10      0.428571        1                   60
1                     18      0.232558        1                   60
2                      8      0.620896        5                   60
3                     20      0.547619        2                   60
4                     14      0.608586        6                   60
…                     …         …             …                   …
8474656               20      0.230769        6                   14
8474657               20      0.823596        0                   33
8474658                9      0.161290        2                   25
8474659               15      0.412844        6                   23
8474660               17      0.000000        4                   20

         isglutenfree  is_organic
0                   0           0
1                   0           0
2                   0           0
3                   0           0
4                   0           0
…                   …           …
8474656             0           0
8474657             0           0
8474658             0           0
```

```
8474659              0           0
8474660              0           0

[8474661 rows x 19 columns]
```

[ ]:
```
# merge train and test  with user_feature
test=test.merge(user_feature  , on='user_id' , how='left')
# merge train and test  with product_fe
test=test.merge(product_fe  , on='product_id' , how='left')
```

[ ]:
```
test
```

[ ]:
```
         order_id  user_id  product_id  ur_pr_count  ur_pr_reordered  \
0              17    36855        1283            1                0
1          657743    16994        1283            1                0
2          834223    92628        1283            1                0
3         1446886    75870        1283            1                0
4         1518981   118458        1283            3                2
…               …       …           …            …                …
4833287   3418159   104905       24836            1                0
4833288   3418520    88900        8102            2                1
4833289   3418520    88900       41872            2                1
4833290   3419574     1743       18920            1                0
4833291   3420740   195822       41431            1                0

         reordered_last  aisle_id  department_id  order_number  \
0                   3.0       100             21             4
1                   3.0       100             21             9
2                   3.0       100             21            14
3                   3.0       100             21            21
4                   3.0       100             21            14
…                     …         …              …             …
4833287             3.0        41              8            15
4833288             3.0        56             18            75
4833289             3.0        94              7            75
4833290             3.0       103             19            36
4833291             3.0        73             11            72

         ttl_cnt_product_user  Avg_no_prod_perOrder  days_since_prior_order  \
0                          27              6.750000               18.437500
1                          61              6.777778               12.820312
2                         333             23.785714               11.789062
3                         241             11.476190               10.164062
4                         124              8.857143               14.304688
…                           …                     …                       …
4833287                   297             19.800000               14.976562
4833288                   661              8.813333                5.484375
```

```
4833289                  661             8.813333                5.484375
4833290                  176             4.888889                8.906250
4833291                  861            11.958333                2.820312

         max_hour_of_day  usr_ro_ratio  max_dow  product_name_length  \
0                      9      0.222222        0                   38
1                     20      0.344262        4                   38
2                      7      0.639640        0                   38
3                     10      0.609959        1                   38
4                     11      0.362903        6                   38
…                    …            …       …                    …
4833287               10      0.363636        0                   30
4833288               18      0.695915        1                   29
4833289               18      0.695915        1                   29
4833290               17      0.414773        2                   17
4833291               15      0.638792        3                   34

         isglutenfree  is_organic
0                   0           1
1                   0           1
2                   0           1
3                   0           1
4                   0           1
…                 …           …
4833287             0           0
4833288             0           0
4833289             0           1
4833290             0           0
4833291             0           0

[4833292 rows x 18 columns]
```

0.1.6 Using idf to fing rare products

[ ]:
```
# getting Tfidf reperesentation of product id  to find rare product
tfidf_prod=TfidfVectorizer(min_df=10)
tfidf_prod.fit(train['product_id'].astype('str'))
```

[ ]:
```
TfidfVectorizer(min_df=10)
```

[ ]:
```
# creating dictionary of feature and its idf score
product_idf=dict(zip(tfidf_prod.get_feature_names(),tfidf_prod.idf_))

# product who has idf score more than 90 percentile is taken as rare
rare_idf_value =np.percentile(tfidf_prod.idf_ , 90)

def idfmap(dictionary , value , rare_idf_value):
```

```
    """
    dictionary : feature and idf map
    value : value at particular index
    rare_idf_value : rare idf value if idf score is greater than 90 percentile␣
 ↪its rare
    """
    if dictionary.get(value):
        if dictionary.get(value)>= rare_idf_value :
            return 1
        else:
            return 0
    # if no value is found in dictionary return 0
    else:
        return 0
```

[ ]:
```
train['product_is_rare']=train['product_id'].apply(lambda prd :␣
 ↪idfmap(product_idf , str(prd) ,rare_idf_value))
test['product_is_rare']=test['product_id'].apply(lambda prd :␣
 ↪idfmap(product_idf , str(prd) ,rare_idf_value))
```

[ ]:
```
del [tfidf_prod ,rare_idf_value ,product_idf]
del [user_feature ,product_fe ,order]
gc.collect()
```

[ ]:
```
40
```

[ ]:
```
# saving dataset for future operations
train.to_parquet('train.gzip',compression='gzip')
test.to_parquet('test.gzip',compression='gzip')
```

1 Reading train and test data

[6]:
```
#aaaa
dir ='/content/drive/MyDrive/instacart/'
```

[7]:
```
train =pd.read_parquet(dir+'train.gzip')
print(train.shape)
# train = train.sample(400000, random_state=1)
# print("Sampling 400000 rows")
# print(train.shape)
```

```
(8474661, 20)
```

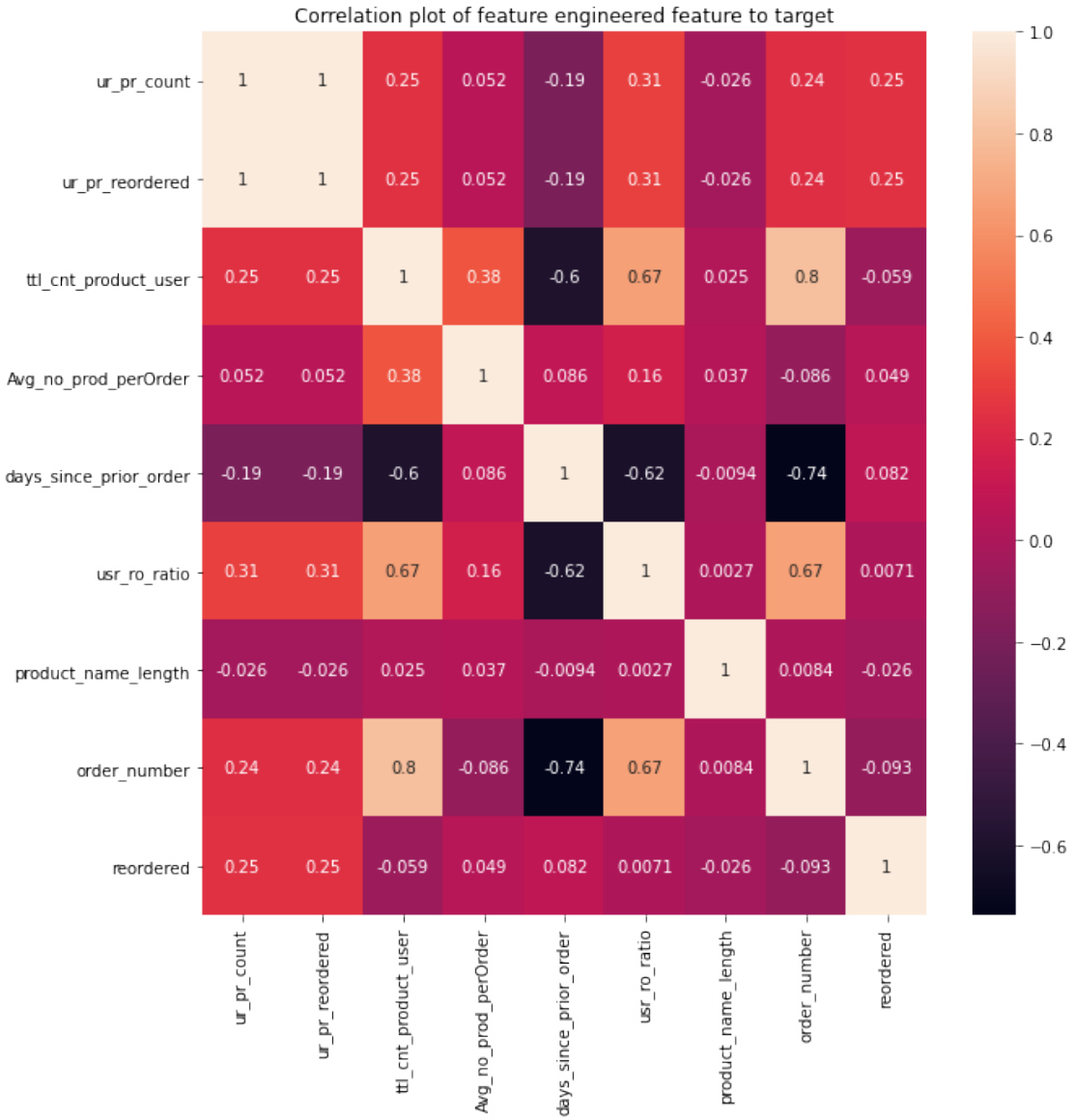
[8]:
```
# train['reordered_last']=train['reordered_last'].astype(np.int8)
# train['max_hour_of_day']=train['max_hour_of_day'].astype(np.int8)
# train['max_dow']=train['max_dow'].astype(np.int8)
```

[9]:
```
train.columns
```

[9]:
```
Index(['order_id', 'user_id', 'product_id', 'ur_pr_count', 'ur_pr_reordered',
       'reordered_last', 'aisle_id', 'department_id', 'reordered',
       'order_number', 'ttl_cnt_product_user', 'Avg_no_prod_perOrder',
       'days_since_prior_order', 'max_hour_of_day', 'usr_ro_ratio', 'max_dow',
       'product_name_length', 'isglutenfree', 'is_organic', 'product_is_rare'],
      dtype='object')
```

1.0.1 checking correlation

[10]:
```
numerical_feature=['ur_pr_count' ,'ur_pr_reordered'  ,'ttl_cnt_product_user',␣
↪'Avg_no_prod_perOrder',
     ␣
↪'days_since_prior_order','usr_ro_ratio','product_name_length','order_number'␣
↪,'reordered']
plt.figure(figsize=(10,10))
sns.heatmap(train.loc[:,numerical_feature].corr(), annot=True)
plt.title("Correlation plot of feature engineered feature to target")
plt.show()
```



- Removing ur_pr_count as it is highly correlated to one another - correlated feature do not add much information as they can be formed by linear combination of each other.

[11]:
```
# saving train target
y = train.reordered.values
# test_temp = test[["order_id", "product_id"]]
# droping ids
train.drop(columns =["order_id", "user_id", "reordered" ,'product_id'␣
 ↪,'department_id' ,'aisle_id' ,'ur_pr_count']  , inplace =True)
# test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
 ↪,'department_id' ,'aisle_id'] , inplace =True)
```

[12]:
```
print("Train shape " ,train.shape)
# print("Test shape " ,test.shape)
```

```
Train shape  (8474661, 13)
```

1.1 Split data

[70]:
```
# !pip install category_encoders
```

[14]:
```
from sklearn.model_selection import train_test_split
import category_encoders as ce
X_train, X_test, y_train, y_test = train_test_split(train, y, stratify=y,␣
↪test_size=0.2)
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

[15]:
```
from sklearn.preprocessing import StandardScaler
test_preprocessing_object ={}
def standardize(X_train, X_test ,test_preprocessing_object , flag ='train'):
    """
    This function standardize test and train columns if flag is train␣
↪standization will fit and trainsform
    if test it will just standardize.
    returns train ,test and  test_preprocessing_object if flag is train else␣
↪test data and test_preprocessing_object
    X_train : train data
    X_test  :test data
    test_preprocessing_object : dictionary containing object of columns␣
↪transformer for different column
    flag : string either train or test
    """
    std_columns␣
↪=['ur_pr_reordered','order_number','ttl_cnt_product_user','Avg_no_prod_perOrder','days_sinc
    scaler_objects =[]

    if(flag == 'train'):
        for col in std_columns:
            print(col)
            scaler = StandardScaler()
            scaler.fit(X_train.loc[:,col].values.reshape(-1,1))
            X_train.loc[:,col] =scaler.transform(X_train.loc[:,col].values.
↪reshape(-1,1))
            X_test.loc[:,col] =scaler.transform(X_test.loc[:,col].values.
↪reshape(-1,1))
```

```
            scaler_objects.append({col:scaler})
            del scaler
            gc.collect()
        test_preprocessing_object['std']=scaler_objects
    elif(flag =='test'):
        for item in test_preprocessing_object['std']:
            for col_item in item.items():
                col =col_item[0]
                scaler=col_item[1]
                print(col)
                X_test.loc[:,col] =scaler.transform(X_test.loc[:,col].values.
↪reshape(-1,1))
    if(flag=='train'):
        return X_train.copy() , X_test.copy() ,test_preprocessing_object
    else:
        return X_test.copy() ,test_preprocessing_object




def OHE(xtrain, column , xtest ):
    """
    This function One hot encode train and test data
    retruns encoded train and test data and encoder object
    xtrain : train data
    column : list of columns to apply encoding
    xtest  : test data
    """
    xtrain = xtrain.reset_index(drop=True)
    xtest =xtest.reset_index(drop =True)
    print("Before")
    print(xtrain.shape)
    print(xtest.shape)
    enc = OneHotEncoder(handle_unknown='ignore' ,sparse=False)
    print("Column name ", column)
    #Train encoding


    temp_train_ohe=enc.fit_transform(xtrain.loc[:,column].values.reshape(-1 ,1))
    # create column_name
    col_name=[column+"_"+str(i+1) for i in range(temp_train_ohe.shape[1])]
    # defining data type
    datatype ={ i:'bool' for i in col_name}
    temp_train_ohe = pd.DataFrame(temp_train_ohe ,columns=col_name )
    xtrain.drop(columns=column , inplace =True)
    new_train =pd.concat([xtrain,temp_train_ohe],axis=1)
```

```
    temp_test_ohe=enc.fit_transform(xtest.loc[:,column].values.reshape(-1 ,1))
    xtest.drop(columns=column , inplace =True)

    temp_test_ohe = pd.DataFrame(temp_test_ohe ,columns=col_name)

    new_test =pd.concat([xtest,temp_test_ohe] , axis =1)
    print("After")
    print(new_train.shape)
    print(new_test.shape)

    return new_train.copy() , new_test.copy(),enc

def ohe_Test(test , test_preprocessing_object ):
    """
    This function transform test data to one hot encoding
    return transformed test data
    test :test data
    test_preprocessing_object:dictionary containing object of columns␣
 ↪transformer for different column
    """
    test =test.reset_index(drop =True)
    ohe=test_preprocessing_object['OHE']
    for column in ohe:
        ohe_encoder = ohe[column]
        temp_test_ohe=ohe_encoder.transform(test.loc[:,column].values.
 ↪reshape(-1 ,1))
        print(temp_test_ohe.shape)
        col_name=[column+"_"+str(i+1) for i in range(temp_test_ohe.shape[1])]
        # defining data type
        datatype ={ i:'bool' for i in col_name}
        temp_test_ohe = pd.DataFrame(temp_test_ohe ,columns=col_name )
        test.drop(columns=column , inplace =True)
        new_test =pd.concat([test,temp_test_ohe] , axis =1)
        test =new_test.copy()

    return test
```

[16]:
```
# Standardizing numerical data
X_train, X_test ,test_preprocessing_object=standardize(X_train, X_test␣
 ↪,test_preprocessing_object , flag ='train')
```

```
ur_pr_reordered

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
```

order_number

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
```

ttl_cnt_product_user

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
```

```
Avg_no_prod_perOrder
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
days_since_prior_order
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
usr_ro_ratio
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
```

```
product_name_length
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
```

[17]:
```
def response_code_train(X_train,y_train ,X_test  ):
  '''
  This function response code categorical varible
  X_train : train data
  y_ train : train label
  X_test: test data
  retruns transformed train , test data along Target encoder object to apply it␣
↪on test data to fit transform test.
  '''

    response_column =['max_hour_of_day' ,'reordered_last','max_dow']
    reponse_dict ={}

    for col in response_column:
      print(col)
      encoder=ce.TargetEncoder(cols=col)

      X_train.loc[:,col] =encoder.fit_transform(X_train[col] , y_train)
      X_test.loc[:,col] =encoder.transform(X_test[col])
      reponse_dict[col]=encoder
      del encoder
      gc.collect()
```

```
    return X_train.copy() , X_test.copy() ,reponse_dict



def response_code_test( X_test ,response_dict):
"""
This function takes data and does fit transform based on column wise,
according to column specific encoder stored in reponse_dctionary
X_test : test data
response_dict : dictionary containing encoder object
return transformed test data
"""

  response_column =['max_hour_of_day' ,'reordered_last','max_dow']
  for col in response_column:
    encoder =response_dict[col]
    X_test.loc[:,col] =encoder.transform(X_test.loc[:,col])



  return X_test.copy()
```

[18]:
```
X_train , X_test ,reponse_dict =response_code_train(X_train,y_train ,X_test  )
```

```
max_hour_of_day

/usr/local/lib/python3.7/dist-packages/category_encoders/utils.py:21:
FutureWarning: is_categorical is deprecated and will be removed in a future
version.  Use is_categorical_dtype instead
  elif pd.api.types.is_categorical(cols):

reordered_last

/usr/local/lib/python3.7/dist-packages/category_encoders/utils.py:21:
FutureWarning: is_categorical is deprecated and will be removed in a future
version.  Use is_categorical_dtype instead
  elif pd.api.types.is_categorical(cols):

max_dow

/usr/local/lib/python3.7/dist-packages/category_encoders/utils.py:21:
FutureWarning: is_categorical is deprecated and will be removed in a future
version.  Use is_categorical_dtype instead
  elif pd.api.types.is_categorical(cols):
```

[19]:
```
del train
gc.collect()
```

[19]: 253

[ ]:
```
# # transforming train test data
# X_train , X_test , ohe_rl =OHE(X_train,'reordered_last' , X_test )
# X_train , X_test ,ohe_rl_mhod=OHE(X_train,'max_hour_of_day' , X_test )
# test_preprocessing_object['OHE']={'reordered_last' :ohe_rl ,'max_hour_of_day'
 ↪:ohe_rl_mhod}
```

2 Compute model Performance

- Key performance criterion F1 score
- we want false positive rate to be very low and Tpr to be high
- We wont be relying to much on AUC because of class imbalance
- We will be more intresting in optimizing precsion and recall

[20]:
```
def recall_score(tp ,fn ):
    return (tp)/(tp+fn)
def precision_score(tp ,fp):
    return (tp)/(tp+fp)
def f1_score_score(precision ,recall):
    return 2*((precision * recall)/(precision + recall))
def accuracy_score(tp, tn , fp ,fn):
    return (tp+tn)/(tp+tn+fp+fn)
def tpr_score(tp ,fn ):
    return (tp)/(tp+fn)
def fpr_score(fp ,tn ):
    return (fp)/(fp+tn)
def fnr_score(fn ,tp ):
    return (fn)/(fn+tp)
def tnr_score(fp ,tn ):
    return (tn)/(tn+fp)

_
def modelPerformance(tp , tn , fn , fp):
    """
    printing tpr,fpr ,tnr, precision , recall and f1 score
    tp:true positive
    tn: true negative
    fn : false negative
    fp : false positive
    """
    print("Accuracy" ,((tp+tn)/(tp+tn+fp+fn)))
    print("True positive rate " ,tpr_score(tp ,fn ))
    print("False positive rate " ,fpr_score(fp ,tn ))
    print("False negative  rate " ,fnr_score(fn ,tp ))
    print("True negative  rate " ,tnr_score(fp ,tn ))
    print("Precision " ,precision_score(tp ,fp))
    print("Recall" ,recall_score(tp ,fn ))
```

```
    print("F1 rate" ,f1_score_score(precision_score(tp ,fp),recall_score(tp ,fn␣
 ↪)))



def printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  ):
    """
    printing model performance
    y_train : True train target
    y_test : True test target
    y_train_pred : Train prediction
    y_test_pred : test prediction
    """
    print("Model Performance")
    print("Train: ")
    tn, fp, fn, tp=confusion_matrix(y_train ,y_train_pred ).ravel()
    modelPerformance(tp , tn , fn , fp)
    print("Test: ")
    tn, fp, fn, tp=confusion_matrix(y_test ,y_test_pred ).ravel()
    modelPerformance(tp , tn , fn , fp)

def plotAUC_ROC_Curve(y_train ,y_train_proba , title ="Decision Tree AUC-ROC␣
 ↪curve"):
    """
    Plot AUC-ROC curve
    y_train : True train target
    y_train_proba: positive class probality
    title : title of plot
    """
    fpr , tpr , threshold=roc_curve(y_train ,y_train_proba )
    print("\nAUC score " , auc(fpr , tpr))
    plt.plot([0,1], [0,1], linestyle='-.', label='No Skill')
    plt.plot(fpr , tpr ,linestyle='--')
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title(title)
    plt.legend()
    plt.show()

def plot_Precision_Recall(y_train ,y_train_proba ,title="Decision Tree␣
 ↪Precision-Recall curve"):
    """
    Plot Precision-Recall curve
    y_train : True train target
    y_train_proba: positive class probality
    title : title of plot
    """
```

```
    precision_ , recall_ , threshold_=precision_recall_curve(y_train␣
↪,y_train_proba)
    no_skill =len(y_train[y_train==1])/len(y_train)
    plt.plot([0,1] , [no_skill,no_skill] ,linestyle='-.' , label='noSkill')
    plt.plot(precision_ , recall_ ,linestyle='--')
    plt.xlabel("Recall")
    plt.ylabel("Precision")
    plt.title(title)
    plt.legend()
    plt.show()


def get_optimal_F1(y_test ,y_test_proba, label="Decision tree Precision recall␣
↪AUC"):
    """
    This function conputes  optimal F1 score
    retruns optimal threshold
    y_test :True Test target
    y_test_proba: Positive data probality
    label: label of plot

    """
    precision_ , recall_ , threshold_=precision_recall_curve(y_test␣
↪,y_test_proba)

    # find f1 score for all thresh
    fscore = (2 * precision_ * recall_) / (precision_ + recall_)
    # percentage of positive pts in data set
    no_skill =len(y_test[y_test==1])/len(y_test)
    # get the index which has max f1 score
    ix = np.argmax(fscore)
    print('Best Threshold=%f, F-Score=%.3f' % (threshold_[ix], fscore[ix]))

    plt.plot([0,1] , [no_skill,no_skill] ,linestyle='--' , label='noSkill')
    plt.plot(precision_ , recall_ ,linestyle=':')
    plt.scatter(recall_[ix], precision_[ix], marker='o', color='black',␣
↪label='Best')
    plt.xlabel("Recall")
    plt.ylabel("Precision")
    plt.title(label)
    plt.legend()
    plt.show()
    return threshold_[ix]

def Model_Perpormance_after_Thresholding(y_train,y_train_proba ,y_test,␣
↪y_test_proba , thresh ):
    """
```

```
    This function prints model preformance credentials after thresholding␣
↪probality
    y_train : True train target
    y_test : True test target
    y_train_pred : Train prediction
    y_test_pred : test prediction
    thresh :Threshold to convert probality to class

    """
    y_train_pred=np.where(y_train_proba<=thresh, 0 , 1 ).tolist()
    y_test_pred =np.where(y_test_proba<=thresh, 0 , 1 ).tolist()
    printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  )


def merge_products(x):
    """
    x : string input
    This function merge group to a list
    returns list of strings
    """
    return " ".join(list(x.astype('str')))


def generate_SubmissionCsv(test_sub, pred , thresh):
    """
    This function create submission csv based on prediction by model
    return submission csv
    test_sub : dataframe containing order_id and product_id

    pred:prediction probality of positive class

    thresh : optimal threshold to convert probality to class
    """

    # if pobality is greter than threshold predict 1 else 0
    test_sub["Pred"] = np.where(pred>=thresh ,1,0)
    # select all cases where prediction is 1
    test_sub = test_sub.loc[test_sub["Pred"].astype('int')==1]
    #group by order_id and create lsit of products
    test_sub = test_sub.groupby("order_id")["product_id"].
↪aggregate(merge_products).reset_index()
    test_sub.columns = ["order_id", "products"]
    # read submission dataframe and merge with test_sub on order_id
    sub_df = pd.read_csv("../sample_submission.csv", usecols=["order_id"])
    sub_df = pd.merge(sub_df, test_sub, how="left", on="order_id")
    # fill missing product as None
    sub_df["products"].fillna("None", inplace=True)
```

```
    return sub_df

def generate_SubmissionCsv_colob(test_sub, pred , thresh):
    """
    This function create submission csv based on prediction by model
    return submission csv, this is google colab specific function
    test_sub : dataframe containing order_id and product_id

    pred:prediction probality of positive class

    thresh : optimal threshold to convert probality to class
    """

    # if pobality is greter than threshold predict 1 else 0
    test_sub["Pred"] = np.where(pred>=thresh ,1,0)
    # select all cases where prediction is 1
    test_sub = test_sub.loc[test_sub["Pred"].astype('int')==1]
    #group by order_id and create lsit of products
    test_sub = test_sub.groupby("order_id")["product_id"].
↪aggregate(merge_products).reset_index()
    test_sub.columns = ["order_id", "products"]
    # read submission dataframe and merge with test_sub on order_id
    sub_df = pd.read_csv(dir+"/sample_submission.csv", usecols=["order_id"])
    sub_df = pd.merge(sub_df, test_sub, how="left", on="order_id")
    # fill missing product as None
    sub_df["products"].fillna("None", inplace=True)

    return sub_df
```

[ ]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[ ]:
```

```

2.1 KNN

- Optimizing Knn

[ ]:
```
def optimizeknn(params,X_train =X_train,y_train=y_train):
  kf = StratifiedKFold(n_splits =3)
  knn=KNeighborsClassifier(**params)

  f1s =[]
  for idx in kf.split(X=X_train ,y =y_train):
    train_idx , test_idx=idx[0] ,idx[1]
    xtrain =X_train.iloc[train_idx ,:]
    ytrain =y_train[train_idx]
```

```
    xtest =X_train.iloc[test_idx ,:]
    ytest =y_train[test_idx]


    knn.fit(xtrain , ytrain )

    y_pred_test=knn.predict(xtest)

    f1=f1_score(ytest , y_pred_test)
    print("F1 : ",f1)
    f1s.append(f1)

  return -1.0 *np.mean(f1s)
```

[ ]:
```
# define parameter space

param_space ={
    'n_neighbors' :scope.int(hp.quniform('n_neighbors' ,5 ,50 ,5)),
    'algorithm':hp.choice('algorithm',['ball_tree', 'kd_tree']),
    'metric':hp.choice('metric',["euclidean","manhattan","chebyshev"])
}

optimization_function = partial(optimizeknn ,X_train =X_train,y_train =y_train)

trail = Trials()

results =fmin(fn =optimization_function , space= param_space , algo =tpe.
 ↪suggest  , max_evals=5 ,trials=trail, )
```

[ ]:
```
print(results)
```

```
{'algorithm': 0, 'metric': 0, 'n_neighbors': 15.0}
```

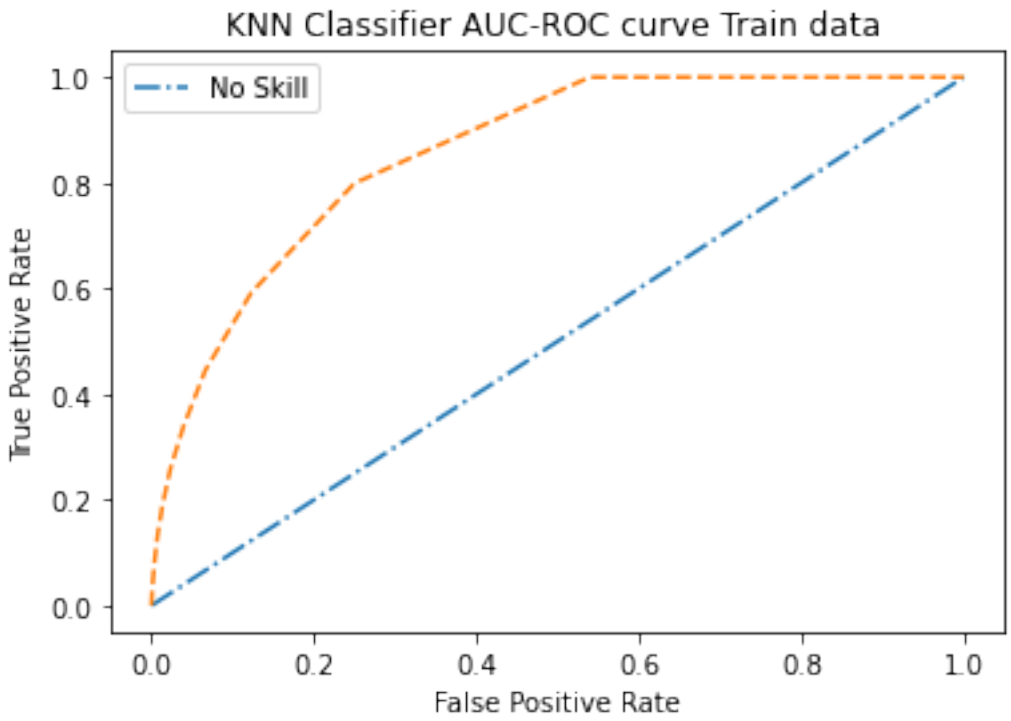
[ ]:
```
knn=KNeighborsClassifier(algorithm='ball_tree', metric= 'euclidean',␣
 ↪n_neighbors= 15 , n_jobs =-1)
knn.fit(X_train , y_train)
```

[ ]:
```
KNeighborsClassifier(algorithm='ball_tree', metric='euclidean', n_jobs=-1,
                     n_neighbors=15)
```

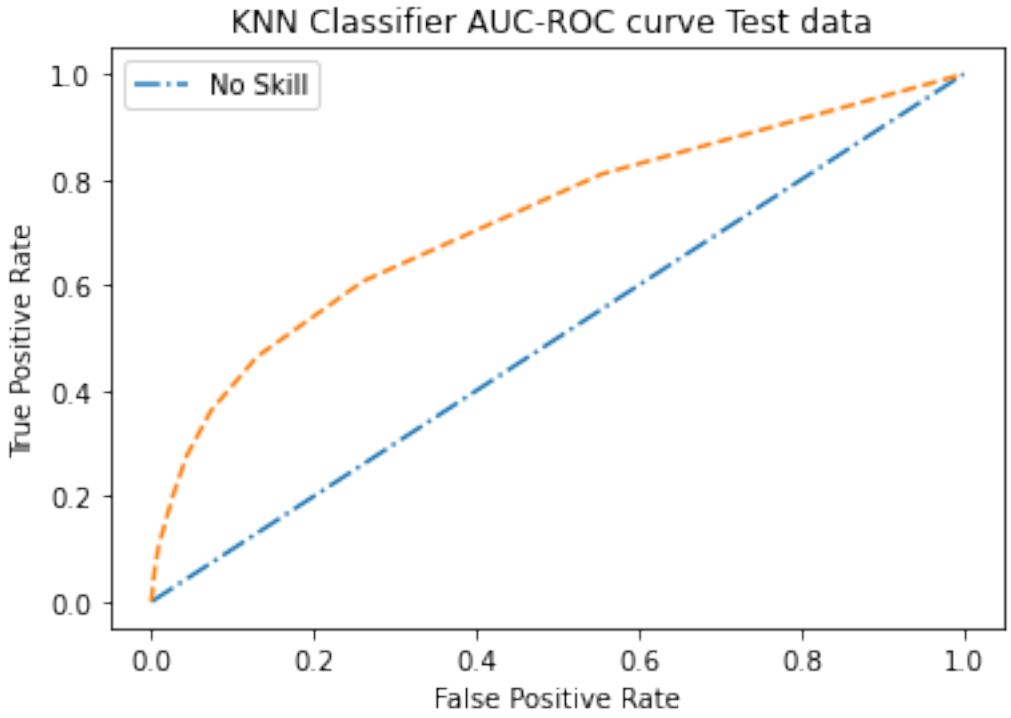
[ ]:
```
# making prediction
y_train_proba=knn.predict_proba(X_train)[:,1]
y_test_proba=knn.predict_proba(X_test)[:,1]
y_train_pred=np.where(y_train_proba>0.5,1 ,0)
y_test_pred=np.where(y_test_proba>0.5,1 ,0)
```

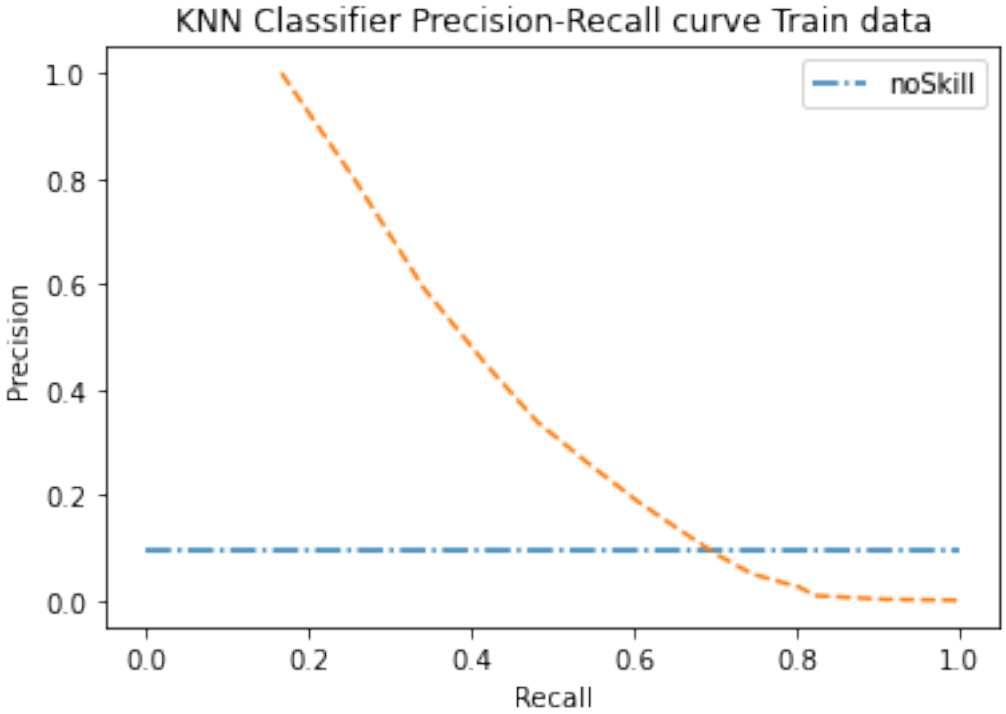
[ ]:
```
# printing model performance
printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  )

plotAUC_ROC_Curve(y_train ,y_train_proba , title ="KNN Classifier AUC-ROC curve␣
 ↪Train data ")
plotAUC_ROC_Curve(y_test ,y_test_proba , title ="KNN Classifier AUC-ROC curve␣
 ↪Test data ")

plot_Precision_Recall(y_train ,y_train_proba ,title="KNN Classifier␣
 ↪Precision-Recall curve Train data")
plot_Precision_Recall(y_test ,y_test_proba ,title="KNN Classifier␣
 ↪Precision-Recall curve Test data")
```

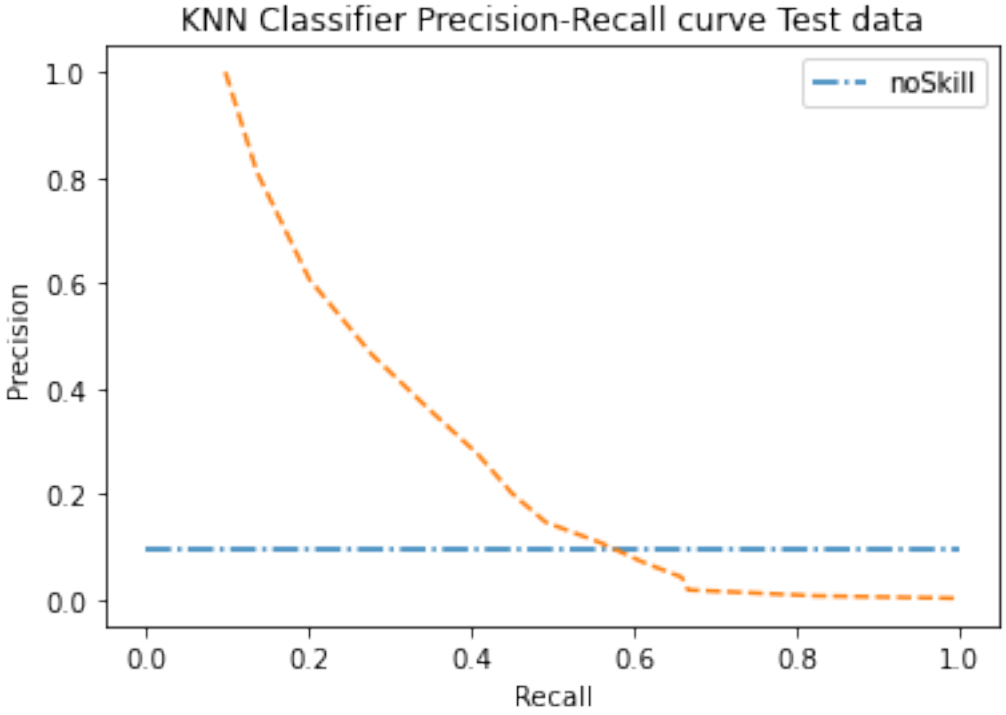
```
Model Performance
Train:
True positive rate  0.13201957753735188
False positive rate  0.007364748878675453
False negative  rate  0.8679804224626482
True negative  rate  0.9926352511213246
Precision  0.6583172768143867
Recall 0.13201957753735188
F1 rate 0.2199334835318099
Test:
True positive rate  0.10819165378670788
False positive rate  0.009192092585414475
False negative  rate  0.8918083462132921
True negative  rate  0.9908079074145856
Precision  0.5585106382978723
Recall 0.10819165378670788
F1 rate 0.18126888217522658

AUC score  0.8578968363514758
```




```
AUC score  0.7220363249785272
```

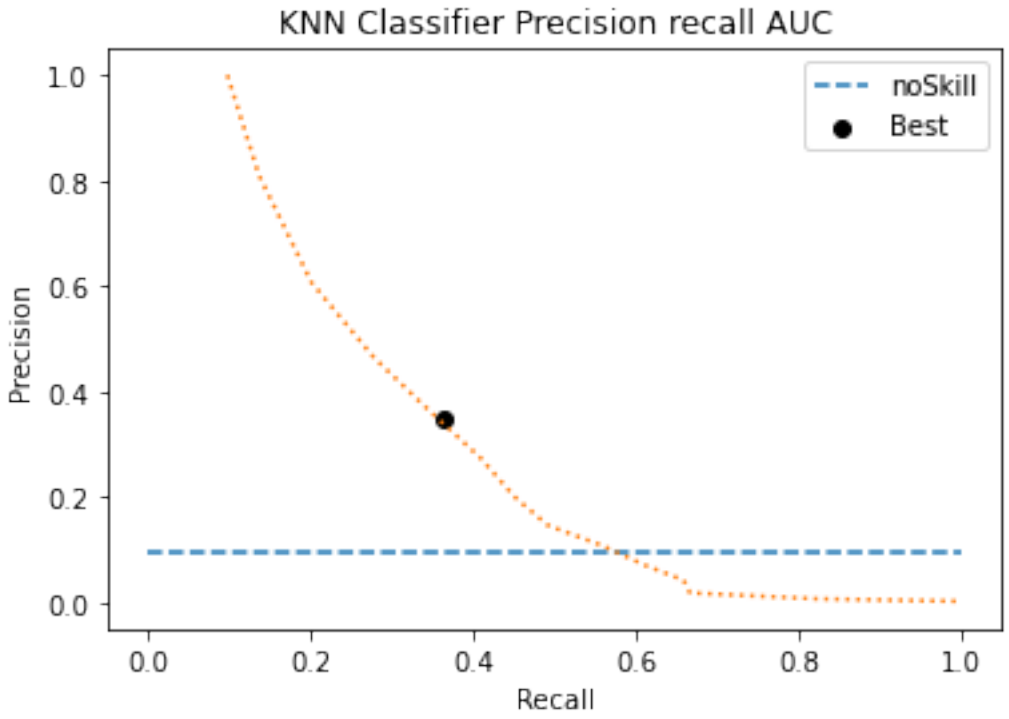



KNN Classifier Precision-Recall curve Train data
noSkill
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


KNN Classifier Precision-Recall curve Test data
noSkill
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]: thresh_knn=get_optimal_F1(y_test ,y_test_proba, label="KNN Classifier Precision␣
     ↪recall AUC")
     Model_Perpormance_after_Thresholding(y_train,y_train_proba ,y_test,␣
     ↪y_test_proba , thresh_knn )
```

```
Best Threshold=0.266667, F-Score=0.354
```




```
Model Performance
Train:
True positive rate  0.33887171561051005
False positive rate  0.03928788969488897
False negative  rate  0.66112828438949
True negative  rate  0.9607121103051111
Precision  0.4810751508502468
Recall 0.33887171561051005
F1 rate 0.39764225799138514
Test:
True positive rate  0.2761463163317877
False positive rate  0.04313638628938479
False negative  rate  0.7238536836682122
True negative  rate  0.9568636137106152
Precision  0.40760456273764256
```

```
Recall 0.2761463163317877
F1 rate 0.32923832923832924
```

[ ]:
```
test =pd.read_parquet('test.gzip')
```

[ ]:
```
test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
↪,'department_id' ,'aisle_id' ,'ur_pr_count'] , inplace =True)
```

[ ]:
```
# reading test data
test =pd.read_parquet('test.gzip')
test_temp = test[["order_id", "product_id"]]
# preparing test data
test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
↪,'department_id' ,'aisle_id' ,'ur_pr_count'] , inplace =True)
# standardizing test data
test , test_preprocessing_object=standardize(None, test␣
↪,test_preprocessing_object , flag ='test')
# One hot encoding test data
test =ohe_Test(test , test_preprocessing_object )
# making prediction on test data
predict_knn_test =knn.predict_proba(test)[:,1]
```

```
ur_pr_reordered
order_number
ttl_cnt_product_user
Avg_no_prod_perOrder
days_since_prior_order
usr_ro_ratio
product_name_length
(4833292, 3)
(4833292, 24)
```

[ ]:
```
# generate submission csv
submission_log=generate_SubmissionCsv(test_temp ,predict_knn_test ,thresh_knn)
```

[ ]:
```
# save submission csv
submission_log.to_csv("./submission/knn.csv",index=False)
```

- Score
- private :0.30895
- public :0.30967

2.2 SVC

- Tuning SVC

[ ]:
```
from  sklearn.svm import SVC
```

[ ]:
```
def optimizesvm(params,X_train =X_train,y_train=y_train):
  kf = StratifiedKFold(n_splits =3)
  svm =SVC(**params)


  f1s =[]
  for idx in kf.split(X=X_train ,y =y_train):
    train_idx , test_idx=idx[0] ,idx[1]
    xtrain =X_train.iloc[train_idx ,:]
    ytrain =y_train[train_idx]

    xtest =X_train.iloc[test_idx ,:]
    ytest =y_train[test_idx]


    svm.fit(xtrain , ytrain )

    y_pred_test=svm.predict(xtest)

    f1=f1_score(ytest , y_pred_test)
    print(f1)
    f1s.append(f1)

  return -1.0 *np.mean(f1s)
```

[ ]:
```
# define parameter space
param_space ={
    'C' :hp.uniform('C' ,0.1,200),
    'gamma' :hp.uniform('gamma' ,0.0001,1),
    'kernel':hp.choice('kernel',['rbf' ,'linear' , 'sigmoid']),
}

optimization_function = partial(optimizesvm ,X_train =X_train,y_train =y_train)

trail = Trials()

results =fmin(fn =optimization_function , space= param_space , algo =tpe.
 ↪suggest  , max_evals=10 ,trials=trail )
print(results)
```

-

[ ]:
```
params={'C': 23.708880283620815, 'gamma': 0.09713900631220027, 'kernel':␣
 ↪'sigmoid'}
```

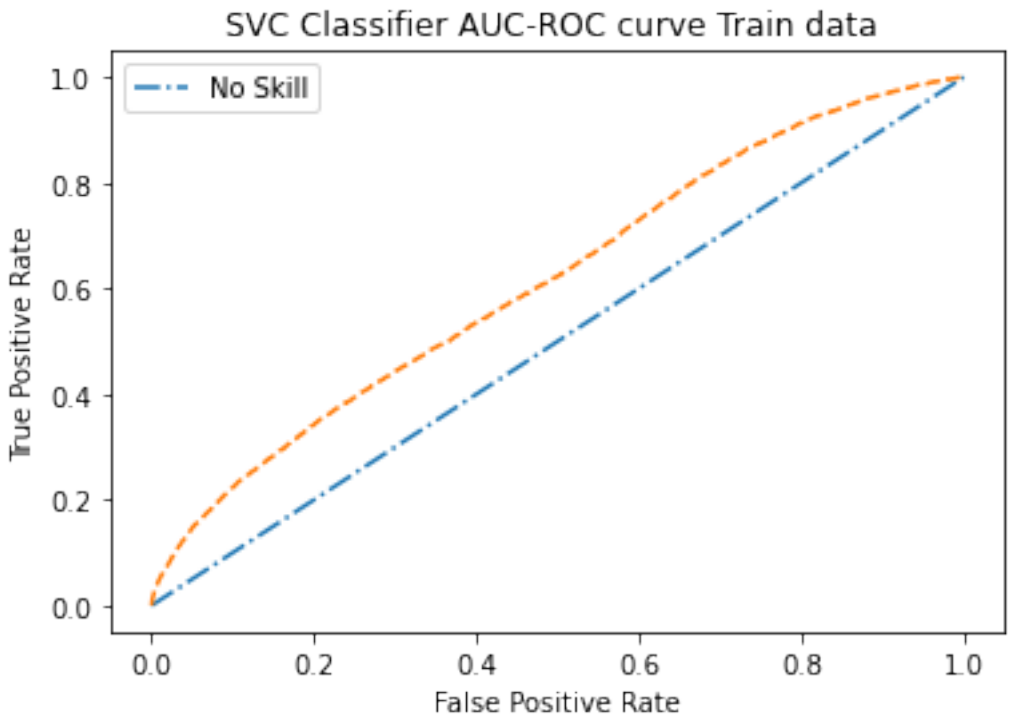
[ ]:
```
svm =SVC(**params ,max_iter =10000 ,probability=True )
svm.fit(X_train , y_train )
```

[ ]:
```
SVC(C=23.708880283620815, gamma=0.09713900631220027, kernel='sigmoid',
    max_iter=10000, probability=True)
```

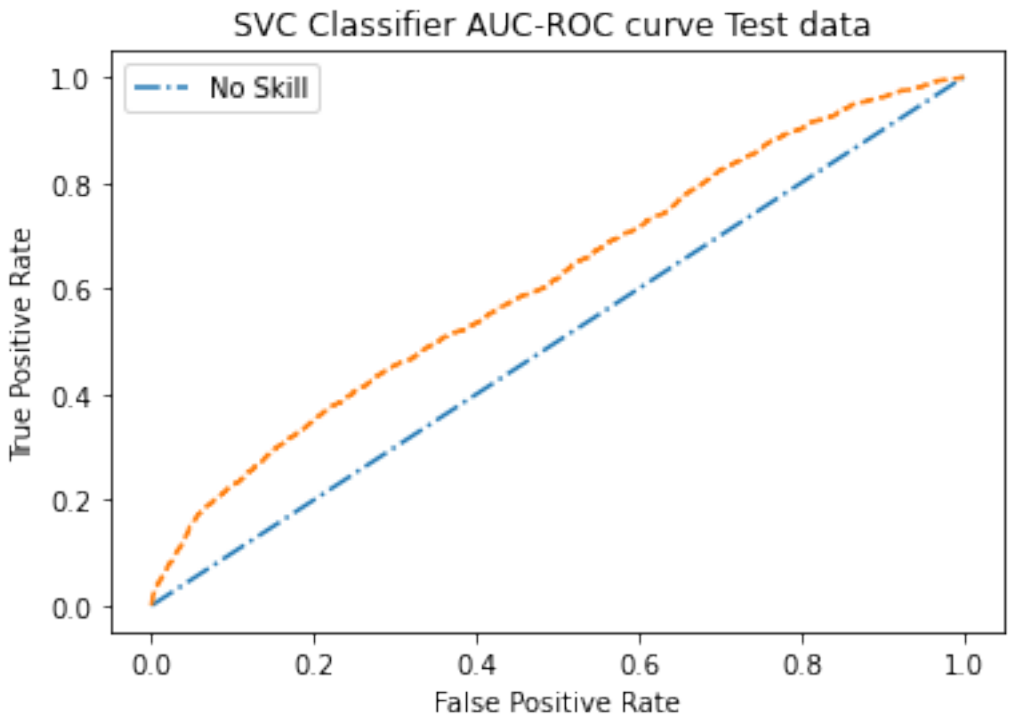
[ ]:
```
# making prediction
y_train_proba=svm.predict_proba(X_train)[:,1]
y_test_proba=svm.predict_proba(X_test)[:,1]
y_train_pred=np.where(y_train_proba>0.5,1 ,0)
y_test_pred=np.where(y_test_proba>0.5,1 ,0)
```

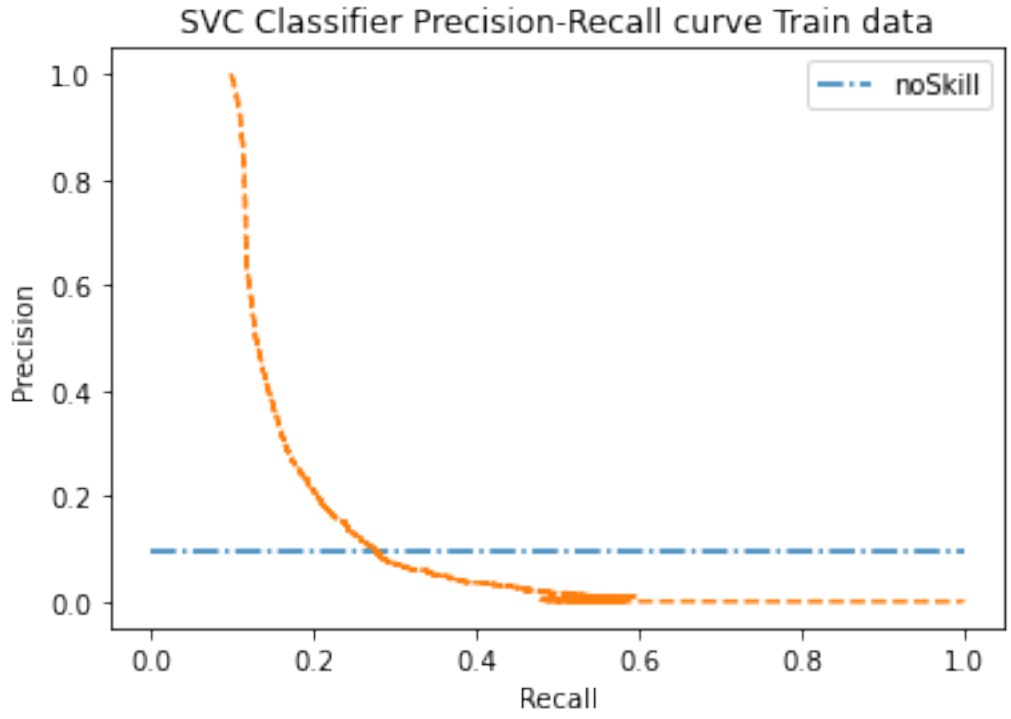
[ ]:
```
# printing model performance

printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  )

plotAUC_ROC_Curve(y_train ,y_train_proba , title ="SVC Classifier AUC-ROC curve␣
 ↪Train data ")
plotAUC_ROC_Curve(y_test ,y_test_proba , title ="SVC Classifier AUC-ROC curve␣
 ↪Test data ")

plot_Precision_Recall(y_train ,y_train_proba ,title="SVC Classifier␣
 ↪Precision-Recall curve Train data")
plot_Precision_Recall(y_test ,y_test_proba ,title="SVC Classifier␣
 ↪Precision-Recall curve Test data")
```

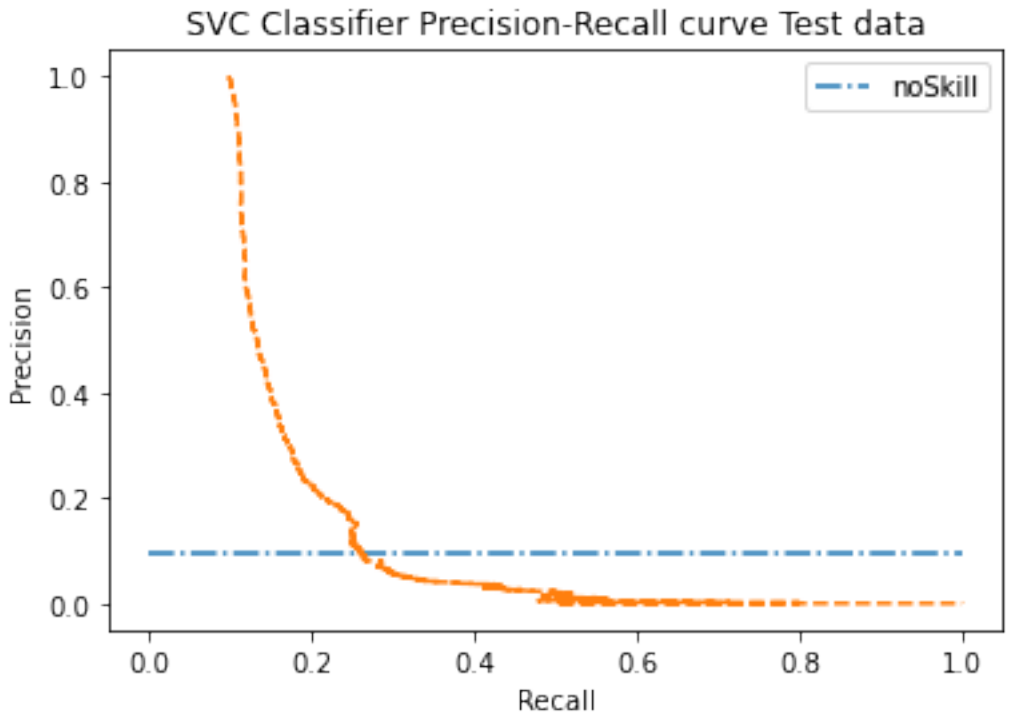
```
Model Performance
Train:
True positive rate  0.02331272539927872
False positive rate  0.002907137715266626
False negative  rate  0.9766872746007212
True negative  rate  0.9970928622847334
Precision  0.4629156010230179
Recall 0.02331272539927872
F1 rate 0.04438994481912937
Test:
True positive rate  0.024214322514167955
False positive rate  0.002879450689406944
False negative  rate  0.975785677485832
True negative  rate  0.997120549310593
Precision  0.47474747474747475
Recall 0.024214322514167955
F1 rate 0.04607843137254902

AUC score  0.6142299404002891
```




```
AUC score  0.6133552341844534
```

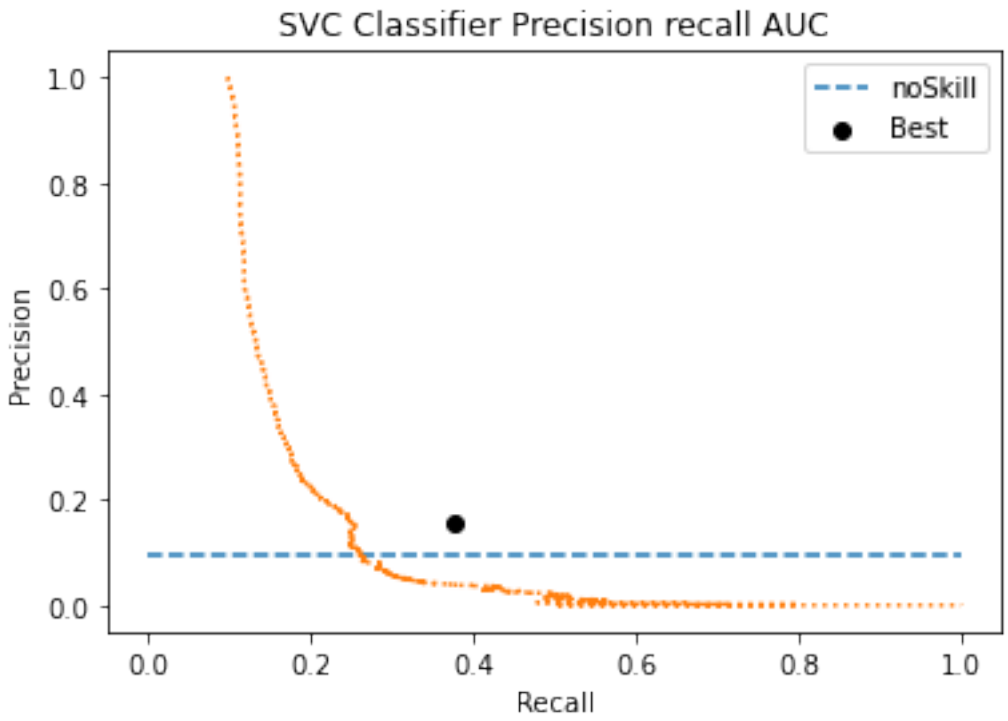

SVC Classifier AUC-ROC curve Test data
No Skill
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


SVC Classifier Precision-Recall curve Train data
noSkill
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0



[ ]:
```
thresh_SVC=get_optimal_F1(y_test ,y_test_proba, label="SVC Classifier Precision␣
↪recall AUC")
Model_Perpormance_after_Thresholding(y_train,y_train_proba ,y_test,␣
↪y_test_proba , thresh_SVC )
```

```
Best Threshold=0.112668, F-Score=0.221
```




```
Model Performance
Train:
True positive rate  0.36501803194229776
False positive rate  0.22103937095077247
False negative  rate  0.6349819680577022
True negative  rate  0.7789606290492276
Precision  0.15073666294346044
Recall 0.36501803194229776
F1 rate 0.21336344814605682
Test:
True positive rate  0.37609479649665123
False positive rate  0.21911512265352456
False negative  rate  0.6239052035033488
True negative  rate  0.7808848773464755
Precision  0.15574994666097716
Recall 0.37609479649665123
F1 rate 0.22027761013880504
```

[ ]:
```
# reading test data
test =pd.read_parquet('test.gzip')
test_temp = test[["order_id", "product_id"]]
# preparing test data
```

```
test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
 ↪,'department_id' ,'aisle_id' ,'ur_pr_count'] , inplace =True)
# standardizing test data
test , test_preprocessing_object=standardize(None, test␣
 ↪,test_preprocessing_object , flag ='test')
# One hot encoding test data
test =ohe_Test(test , test_preprocessing_object )
# making prediction on test data
predict_svm_test =svm.predict_proba(test)[:,1]
```

```
ur_pr_reordered
order_number
ttl_cnt_product_user
Avg_no_prod_perOrder
days_since_prior_order
usr_ro_ratio
product_name_length
(4833292, 3)
(4833292, 24)
```

[ ]:
```
#987
```

[ ]:
```
# generate submission csv
submission_log=generate_SubmissionCsv(test_temp ,predict_svm_test ,thresh_SVC)
```

[ ]:
```
submission_log.to_csv("./submission/svc.csv",index=False)
```

- Score
- private :0.19827
- public :0.20212

2.3 Logistic Regression

[ ]:
```
from sklearn.linear_model import LogisticRegression
```

- Hyperparameter tunning Logistic regression

[ ]:
```
distributions = {'penalty':["l1", "l2" ,'elasticnet'],'C': [100, 10, 1.0, 0.1,␣
 ↪0.01] ,'class_weight' :[None ,'balanced'] ,
                 'max_iter' :[10 ,20 ,30 ,40,50,60,70,80,90,100,110]}
```

[ ]:
```
logistic=LogisticRegression()
clf =RandomizedSearchCV(logistic , distributions ,scoring ='f1',n_iter = 30 ,cv␣
↪=3,verbose=3,return_train_score=True ,n_jobs=-1)
clf.fit(X_train , y_train)
```

Fitting Logistic regression with best tuned parameter Best parameters {'penalty': 'l2', 'max_iter': 30, 'class_weight': 'balanced', 'C': 0.01}

[ ]:
```
param ={'penalty': 'l2', 'max_iter': 30, 'class_weight': 'balanced', 'C': 0.01}
```

[ ]:
```
# Training with best parameter
logistic=LogisticRegression(**param)
logistic.fit(X_train , y_train)
```

```
/home/aditya/anaconda3/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

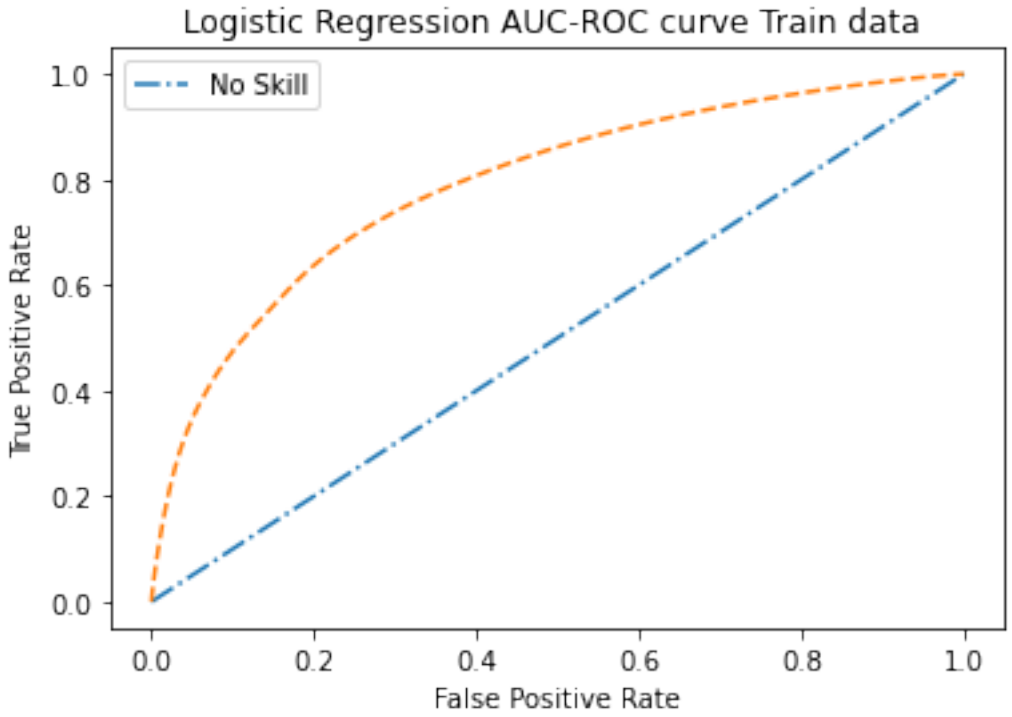
[ ]:
```
LogisticRegression(C=0.01, class_weight='balanced', max_iter=30)
```

[ ]:
```
# making prediction
y_train_pred=logistic.predict(X_train)
y_test_pred=logistic.predict(X_test)
y_train_proba=logistic.predict_proba(X_train)[:,1]
y_test_proba=logistic.predict_proba(X_test)[:,1]
```

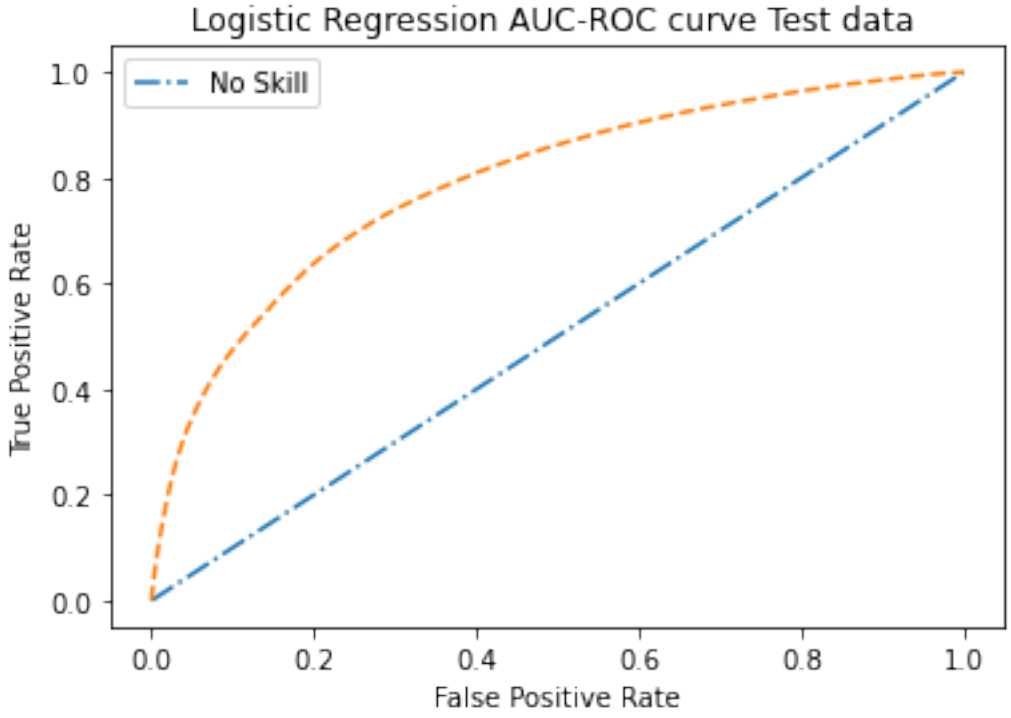
[ ]:
```
# printing model performance
printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  )

plotAUC_ROC_Curve(y_train ,y_train_proba , title ="Logistic Regression AUC-ROC␣
↪curve Train data ")
plotAUC_ROC_Curve(y_test ,y_test_proba , title ="Logistic Regression AUC-ROC␣
↪curve Test data ")

plot_Precision_Recall(y_train ,y_train_proba ,title="Logistic Regression␣
↪Precision-Recall curve Train data")
```

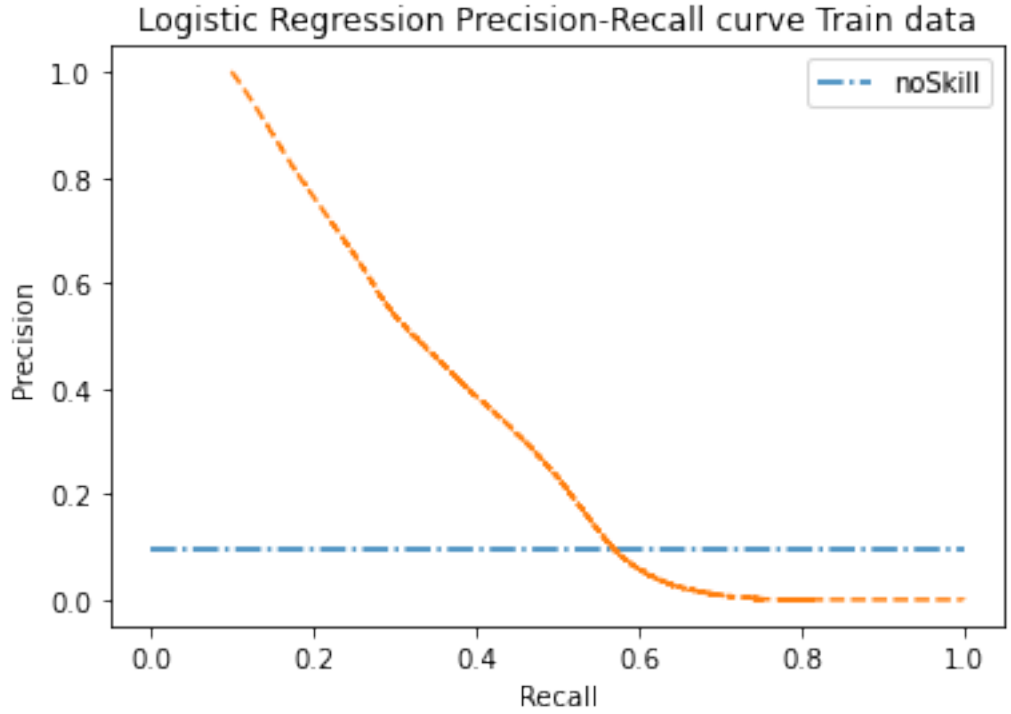
```
plot_Precision_Recall(y_test ,y_test_proba ,title="Logistic Regression␣
 ↪Precision-Recall curve Test data")
```

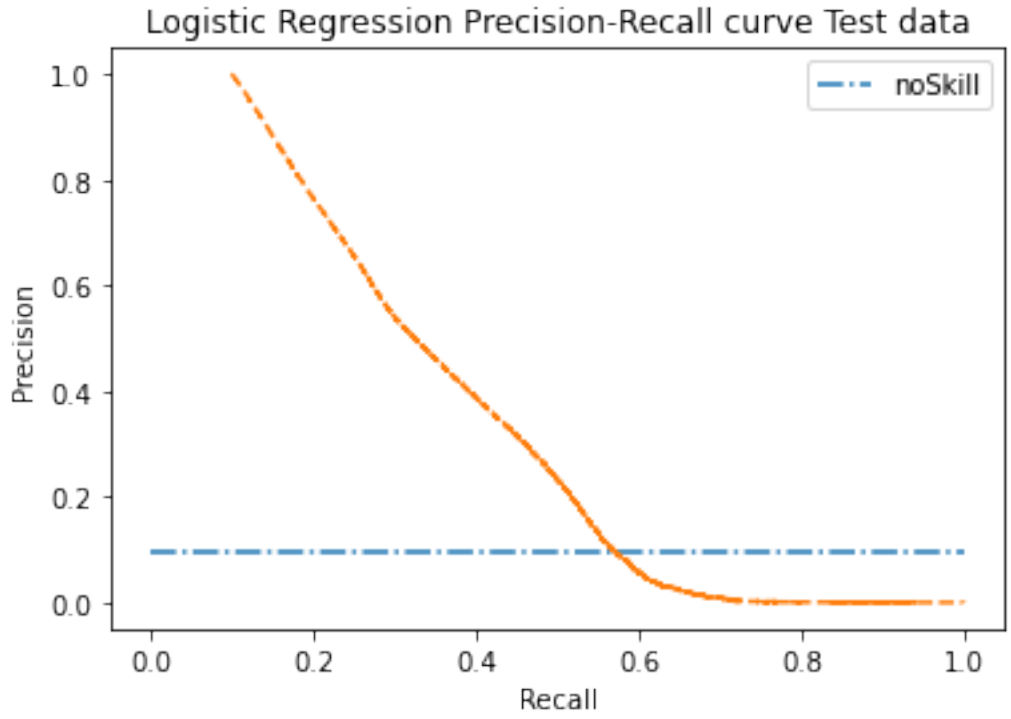
```
Model Performance
Train:
True positive rate  0.619021836669135
False positive rate  0.18649807599528437
False negative  rate  0.380978163330865
True negative  rate  0.8135019240047157
Precision  0.2646011623296878
Recall 0.619021836669135
F1 rate 0.37073253565309133
Test:
True positive rate  0.6197025910174041
False positive rate  0.18663809339457796
False negative  rate  0.38029740898259584
True negative  rate  0.8133619066054221
Precision  0.26466920536114563
Recall 0.6197025910174041
F1 rate 0.3709213545625403

AUC score  0.7902664066651505
```



```
AUC score  0.7908381306504623
```



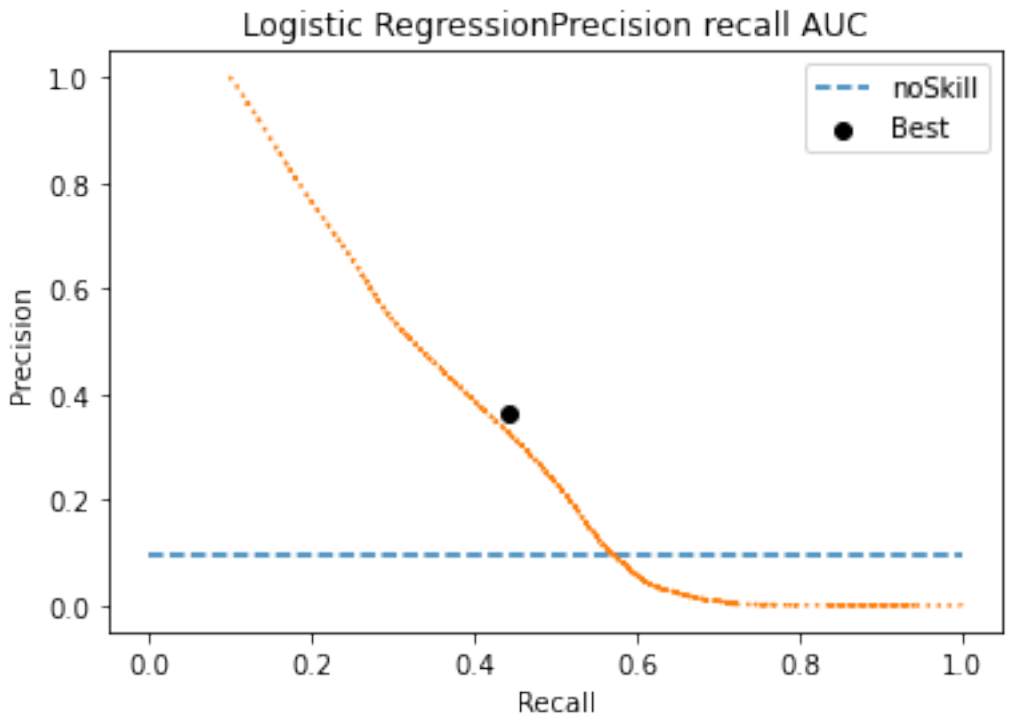




- Finding the optimal threshold from precision recall to maximize f1 score

[ ]:
```
thresh_lr=get_optimal_F1(y_test ,y_test_proba, label="Logistic␣
↪RegressionPrecision recall AUC")
Model_Perpormance_after_Thresholding(y_train,y_train_proba ,y_test,␣
↪y_test_proba , thresh_lr )
```

```
Best Threshold=0.661966, F-Score=0.399
```




```
Model Performance
Train:
True positive rate  0.4401222214011121
False positive rate  0.08382519963071404
False negative  rate  0.5598777785988879
True negative  rate  0.916174800369286
Precision  0.3627171689300212
Recall 0.4401222214011121
F1 rate 0.39768822519771846
Test:
True positive rate  0.44138388682773805
False positive rate  0.08384690236782355
False negative  rate  0.5586161131722619
True negative  rate  0.9161530976321764
Precision  0.3633194625140281
Recall 0.44138388682773805
F1 rate 0.39856515237765805
```

2.3.1 Preparing for submission

[ ]:
```
# reading test data
test =pd.read_parquet('test.gzip')
test_temp = test[["order_id", "product_id"]]
# preparing test data
```

```
test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
 ↪,'department_id' ,'aisle_id' ,'ur_pr_count'] , inplace =True)
# standardizing test data
test , test_preprocessing_object=standardize(None, test␣
 ↪,test_preprocessing_object , flag ='test')
# One hot encoding test data
test =ohe_Test(test , test_preprocessing_object )
# making prediction on test data
predict_logistic_test =logistic.predict_proba(test)[:,1]
```

```
ur_pr_reordered
order_number
ttl_cnt_product_user
Avg_no_prod_perOrder
days_since_prior_order
usr_ro_ratio
product_name_length
(4833292, 3)
(4833292, 24)
```

[ ]:
```
# generate submission csv
submission_log=generate_SubmissionCsv(test_temp ,predict_logistic_test␣
 ↪,thresh_lr)
```

[ ]:
```
# save submission csv
submission_log.to_csv("./submission/logisticRegression.csv",index=False)
```

- Score
- private :0.34327
- public 0.34449

3 Naive bayes

[ ]:
```
from sklearn.naive_bayes import GaussianNB  , BernoulliNB
```

- Hyperparamter tunning Gaussion Naive bayes

[ ]:
```
distributions = {'var_smoothing' :list(np.logspace(-9,4, num=20))}
gnb =GaussianNB()
clf =RandomizedSearchCV(gnb , distributions ,scoring ='f1',n_iter = 30 ,cv␣
 ↪=3,verbose=3,return_train_score=True ,n_jobs=-1)
clf.fit(X_train , y_train)
```

- Best hypertunned parameter
- 'var_smoothing': 0.03359818286283781
- Fitting with best hyperparameter

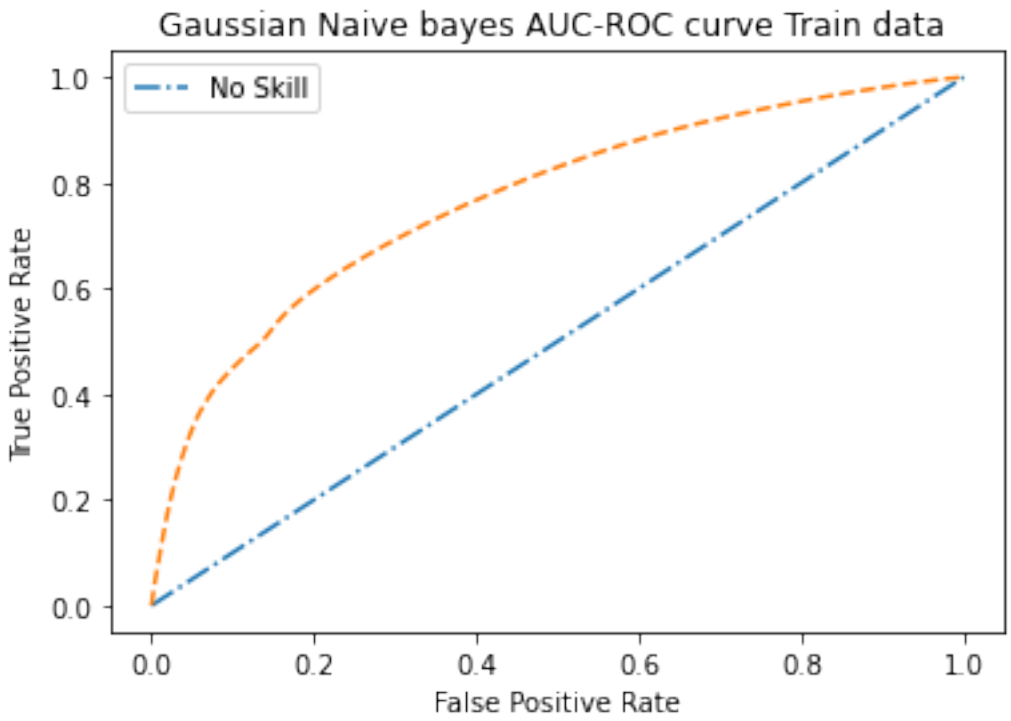
[ ]:
```
gnb =GaussianNB(var_smoothing= 0.03359818286283781)
gnb.fit(X_train , y_train)
```

[ ]:
```
GaussianNB(var_smoothing=0.03359818286283781)
```

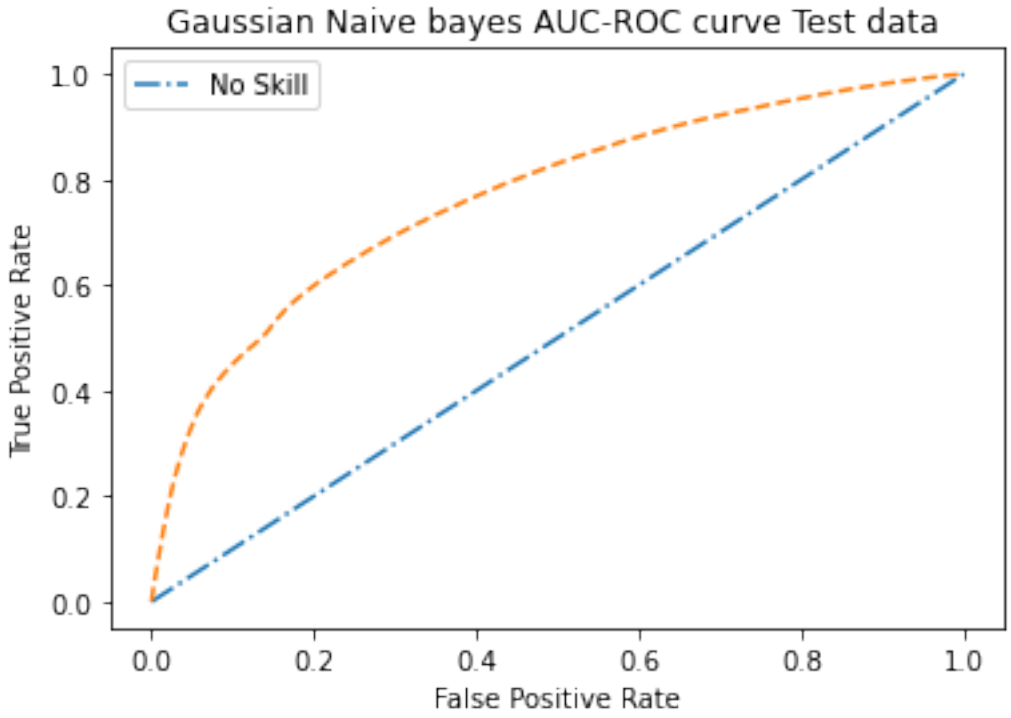
[ ]:
```
# making prediction
y_train_pred=gnb.predict(X_train)
y_test_pred=gnb.predict(X_test)
y_train_proba=gnb.predict_proba(X_train)[:,1]
y_test_proba=gnb.predict_proba(X_test)[:,1]
```

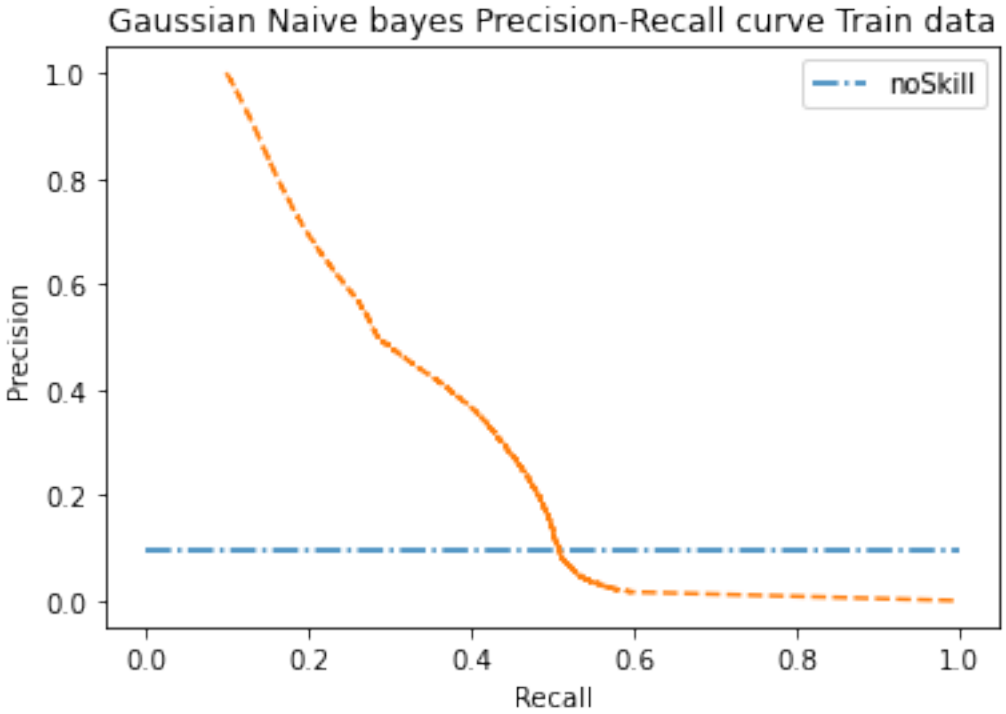
[ ]:
```
# printing model performance
printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  )

plotAUC_ROC_Curve(y_train ,y_train_proba , title ="Gaussian Naive bayes AUC-ROC␣
 ↪curve Train data ")
plotAUC_ROC_Curve(y_test ,y_test_proba , title ="Gaussian Naive bayes AUC-ROC␣
 ↪curve Test data ")

plot_Precision_Recall(y_train ,y_train_proba ,title="Gaussian Naive bayes␣
 ↪Precision-Recall curve Train data")
plot_Precision_Recall(y_test ,y_test_proba ,title="Gaussian Naive bayes␣
 ↪Precision-Recall curve Test data")
```

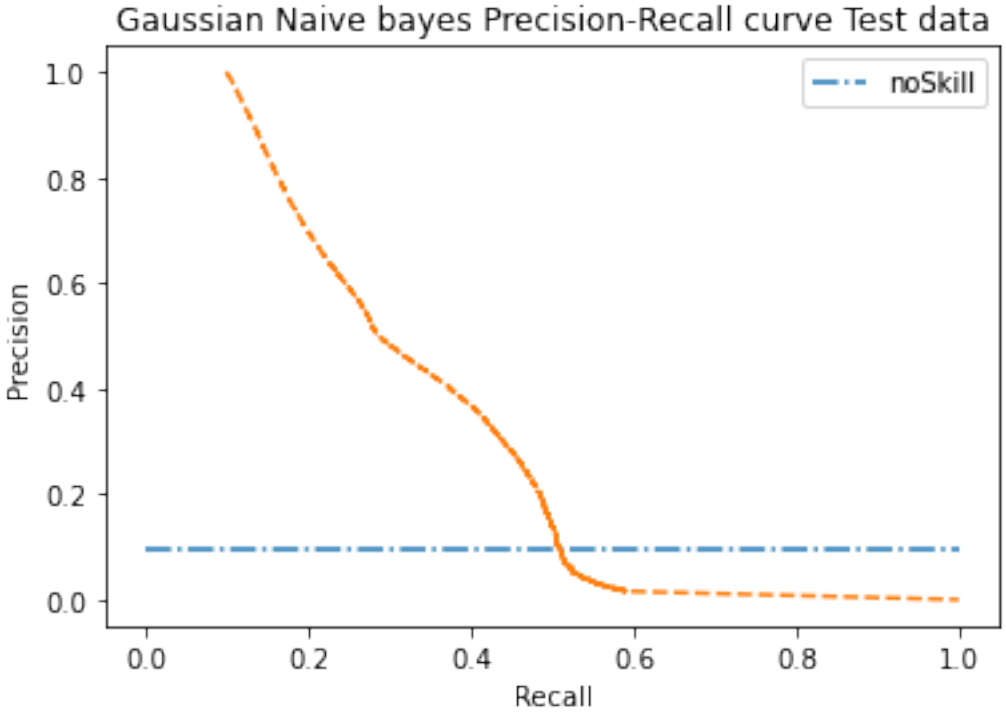
```
Model Performance
Train:
True positive rate  0.41607760395379595
False positive rate  0.08046045976985186
False negative  rate  0.583922396046204
True negative  rate  0.9195395402301482
Precision  0.3592080558933589
Recall 0.41607760395379595
F1 rate 0.3855570532453167
Test:
True positive rate  0.4164208367266914
False positive rate  0.0802410199533341
False negative  rate  0.5835791632733086
True negative  rate  0.9197589800466659
Precision  0.3600271214729046
Recall 0.4164208367266914
F1 rate 0.38617603043399207

AUC score  0.7660031868067414
```




```
AUC score  0.7665034398402442
```

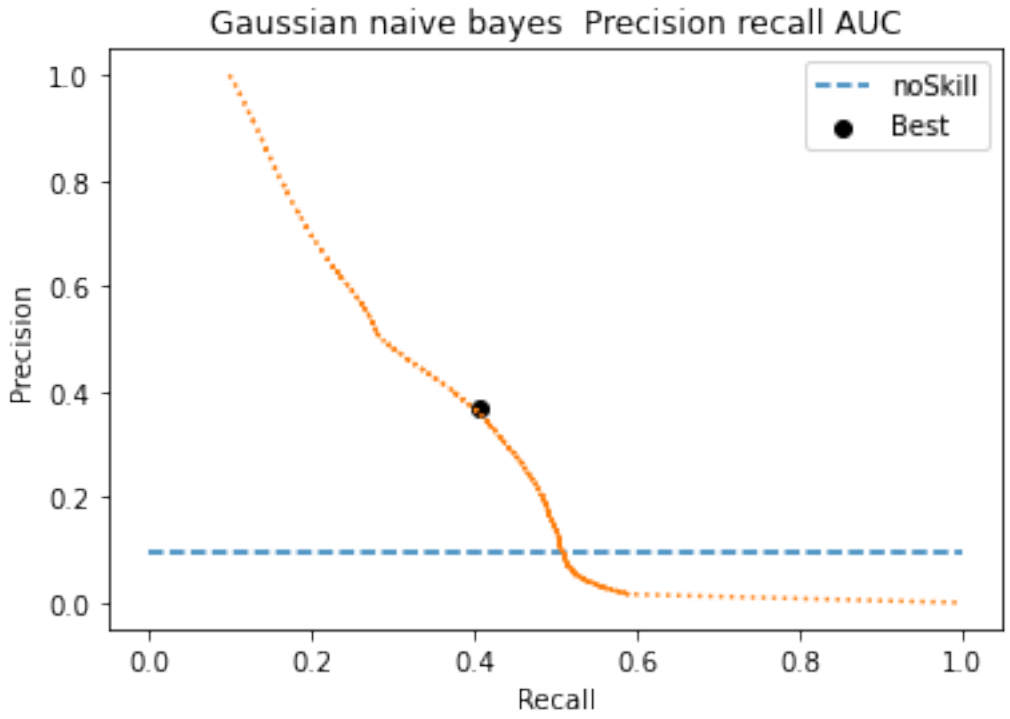



Gaussian Naive bayes Precision-Recall curve Train data
noSkill
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Gaussian Naive bayes Precision-Recall curve Test data
noSkill
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]: thresh_gnb=get_optimal_F1(y_test ,y_test_proba, label="Gaussian naive bayes ␣
     ↪Precision recall AUC")
     Model_Perpormance_after_Thresholding(y_train,y_train_proba ,y_test,␣
     ↪y_test_proba , thresh_gnb )
```

```
Best Threshold=0.578649, F-Score=0.387
```




```
Model Performance
Train:
True positive rate  0.40427624087750863
False positive rate  0.07466008051114095
False negative  rate  0.5957237591224913
True negative  rate  0.9253399194888591
Precision  0.36987429801443295
Recall 0.40427624087750863
F1 rate 0.3863108873178847
Test:
True positive rate  0.4045305100594215
False positive rate  0.07435285070051165
False negative  rate  0.5954694899405786
True negative  rate  0.9256471492994883
Precision  0.37098282205194877
```

```
Recall 0.4045305100594215
F1 rate 0.3870310515987533
```

3.0.1 Preparing for submission

[ ]:
```
# reading test data
test =pd.read_parquet('test.gzip')
test_temp = test[["order_id", "product_id"]]

# preparing test data
test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
 ↪,'department_id' ,'aisle_id' ,'ur_pr_count'] , inplace =True)
# standardizing test data
test , test_preprocessing_object=standardize(None, test␣
 ↪,test_preprocessing_object , flag ='test')
# One hot encoding test data
test =ohe_Test(test , test_preprocessing_object )
# making prediction on test data
predict_gnb_test =gnb.predict_proba(test)[:,1]
```

```
ur_pr_reordered
order_number
ttl_cnt_product_user
Avg_no_prod_perOrder
days_since_prior_order
usr_ro_ratio
product_name_length
(4833292, 3)
(4833292, 24)
```

[ ]:
```
# generate submission csv
submission_log=generate_SubmissionCsv(test_temp ,predict_gnb_test ,thresh_gnb)
```

[ ]:
```
# save submission csv
submission_log.to_csv("./submission/gnb.csv",index=False)
```

- score
- private :0.33670
- public :0.33727

3.1 Bernoulli Nb

3.1.1 Hyperpareter tunning Bernouli naive bayes

[ ]:
```
distributions = {'alpha' :list(np.logspace(-9,4, num=20))}
bnb =BernoulliNB()
clf =RandomizedSearchCV(bnb , distributions ,scoring ='f1',n_iter = 30 ,cv␣
 ↪=3,verbose=3,return_train_score=True ,n_jobs
```

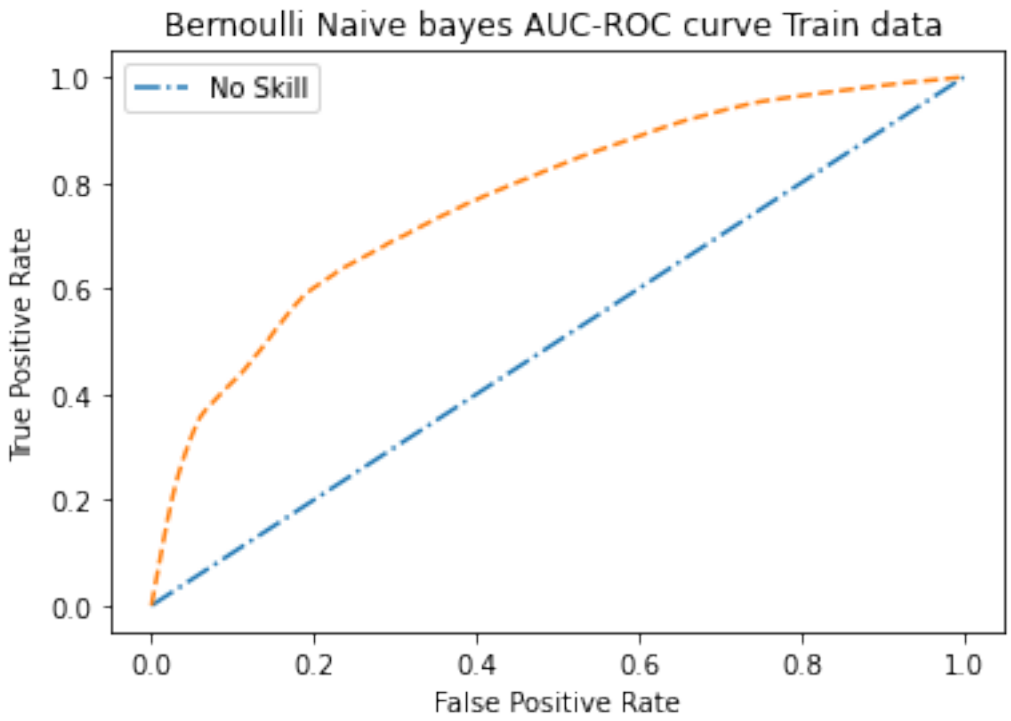
- Best hyperparam
- {'alpha': 428.13323987193957}
- Fitting with best Hyperprameter

[ ]:
```
bnb =BernoulliNB(alpha= 428.13323987193957)
bnb.fit(X_train , y_train)
```

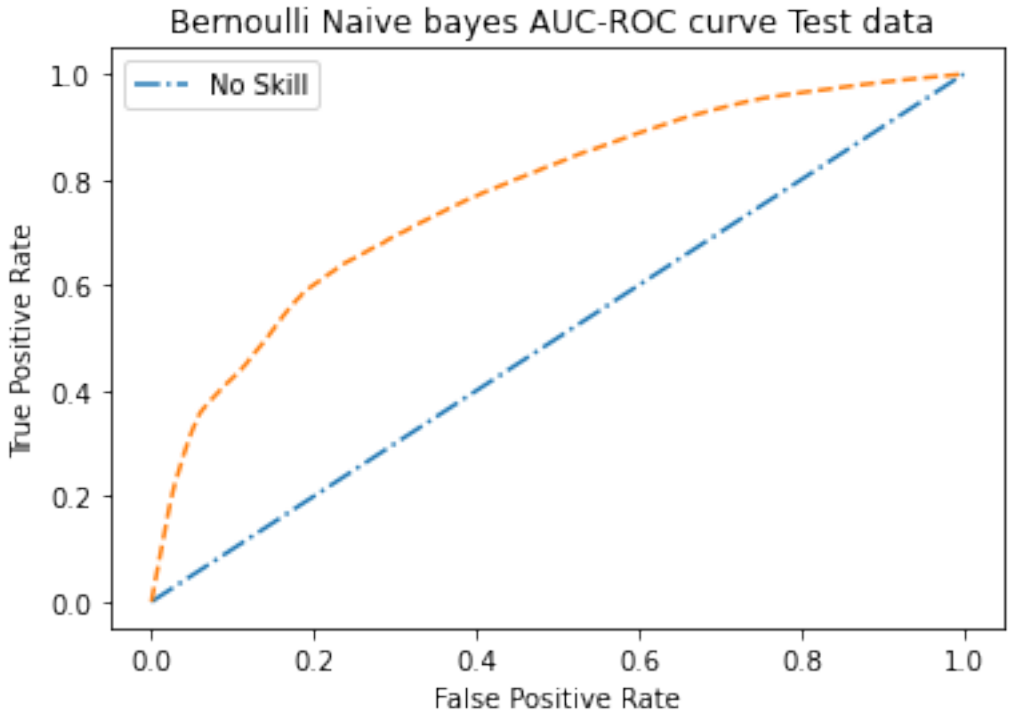
[ ]:
```
# making prediction
y_train_pred=bnb.predict(X_train)
y_test_pred=bnb.predict(X_test)
y_train_proba=bnb.predict_proba(X_train)[:,1]
y_test_proba=bnb.predict_proba(X_test)[:,1]
```

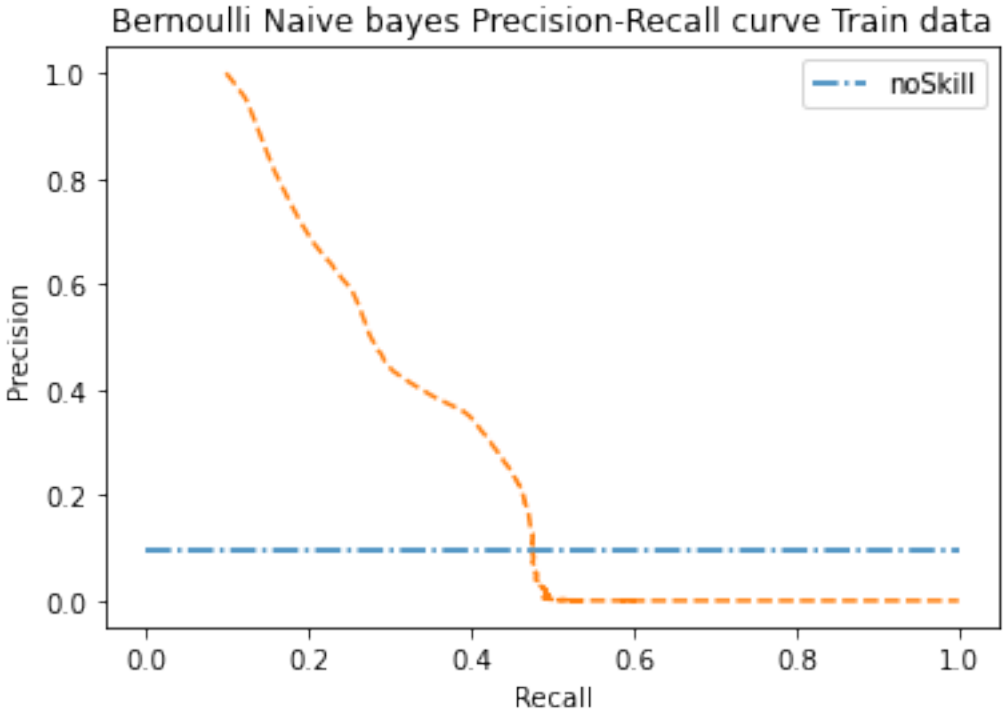
[ ]:
```
# printing model performance
printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  )

plotAUC_ROC_Curve(y_train ,y_train_proba , title ="Bernoulli Naive bayes␣
↪AUC-ROC curve Train data ")
plotAUC_ROC_Curve(y_test ,y_test_proba , title ="Bernoulli Naive bayes AUC-ROC␣
↪curve Test data ")

plot_Precision_Recall(y_train ,y_train_proba ,title="Bernoulli Naive bayes␣
↪Precision-Recall curve Train data")
plot_Precision_Recall(y_test ,y_test_proba ,title="Bernoulli Naive bayes␣
↪Precision-Recall curve Test data")
```

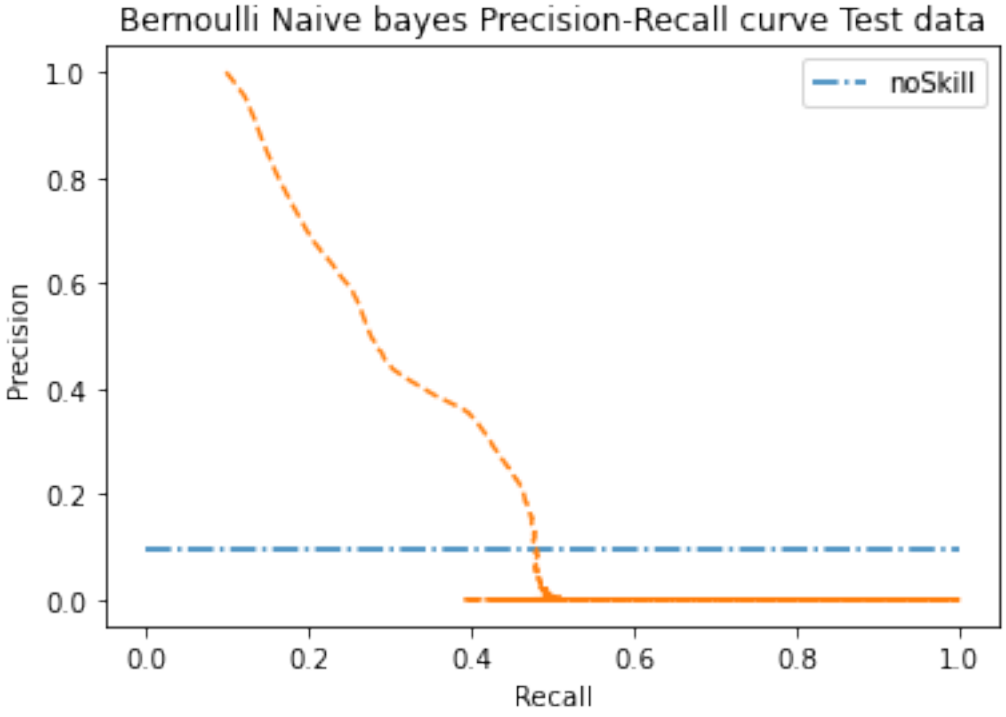
```
Model Performance
Train:
True positive rate  0.3598352484469708
False positive rate  0.06127027635466297
False negative  rate  0.6401647515530292
True negative  rate  0.938729723645337
Precision  0.38899051457377537
Recall 0.3598352484469708
F1 rate 0.37384530652504144
Test:
True positive rate  0.3606551443308298
False positive rate  0.061119510740481095
False negative  rate  0.6393448556691702
True negative  rate  0.938880489259519
Precision  0.39011784973180375
Recall 0.3606551443308298
F1 rate 0.37480839218710327

AUC score  0.7674538578303565
```




```
AUC score  0.7676416650938482
```

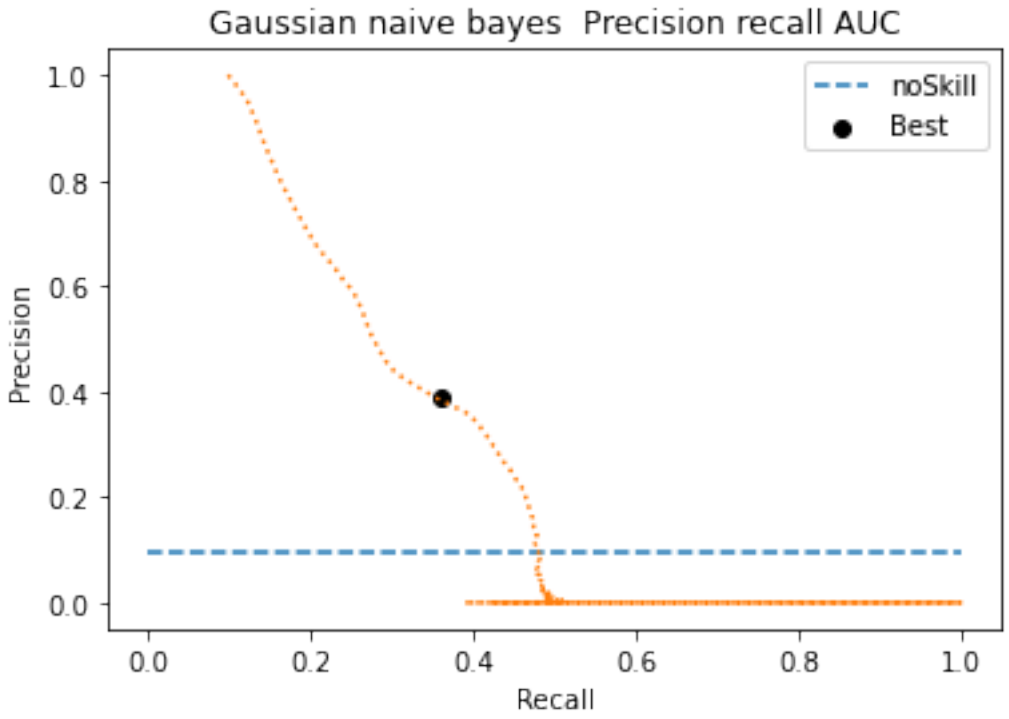



Bernoulli Naive bayes Precision-Recall curve Train data
noSkill
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Bernoulli Naive bayes Precision-Recall curve Test data
noSkill
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]: thresh_bnb=get_optimal_F1(y_test ,y_test_proba, label="Gaussian naive bayes ␣
     ↪Precision recall AUC")
     Model_Perpormance_after_Thresholding(y_train,y_train_proba ,y_test,␣
     ↪y_test_proba , thresh_bnb )
```

```
Best Threshold=0.501781, F-Score=0.375
```




```
Model Performance
Train:
True positive rate  0.35973118530930126
False positive rate  0.061225480731424246
False negative  rate  0.6402688146906987
True negative  rate  0.9387745192685758
Precision  0.38909560715083874
Recall 0.35973118530930126
F1 rate 0.3738376494226842
Test:
True positive rate  0.36058878532862787
False positive rate  0.06107635001517165
False negative  rate  0.6394112146713721
True negative  rate  0.9389236499848284
Precision  0.3902421508268644
```

```
Recall 0.36058878532862787
F1 rate 0.37482990211140854
```

3.1.2 Preparing for submission

[ ]:
```
# reading test data
test =pd.read_parquet('test.gzip')
test_temp = test[["order_id", "product_id"]]

# preparing test data
test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
 ↪,'department_id' ,'aisle_id' ,'ur_pr_count'] , inplace =True)
# standardizing test data
test , test_preprocessing_object=standardize(None, test␣
 ↪,test_preprocessing_object , flag ='test')
# One hot encoding test data
test =ohe_Test(test , test_preprocessing_object )
# making prediction on test data
predict_bnb_test =bnb.predict_proba(test)[:,1]
```

```
ur_pr_reordered
order_number
ttl_cnt_product_user
Avg_no_prod_perOrder
days_since_prior_order
usr_ro_ratio
product_name_length
(4833292, 3)
(4833292, 24)
```

[ ]:
```
# generate submission csv
submission_log=generate_SubmissionCsv(test_temp ,predict_bnb_test ,thresh_bnb)
# save submission csv
submission_log.to_csv("./submission/bnb.csv",index=False)
```

- Score
- private :0.32720
- public :0.32735

4 Decision Tree

5 Hyper tuning model

```
[ ]: distributions = {'criterion':["gini", "entropy"],'splitter': ["best", "random"]␣
     ↪,'max_depth' :[3,5,10 , 20 , 50,100] ,
                    'min_samples_split' :[3,5,10 , 20 , 50,100,150 ,200],
'min_samples_leaf' :[3,5,10 , 20 , 50,100,150 ,200]}
```

```
[ ]: decisionTree =DecisionTreeClassifier()
     clf =RandomizedSearchCV(decisionTree , distributions ,scoring ='f1',n_iter = 30␣
     ↪,cv =3,verbose=3,return_train_score=True ,n_jobs=-1)
```

```
[ ]: clf.fit(X_train, y_train)
```

5.1 Hyperparam tuned best prameters

- {'splitter': 'best', 'min_samples_split': 10, 'min_samples_leaf': 10, 'max_depth': 100, 'criterion': 'gini'}

5.1.1 Training with tuned parameter

```
[ ]: param ={'splitter': 'best', 'min_samples_split': 10, 'min_samples_leaf': 10,␣
     ↪'max_depth': 100, 'criterion': 'gini'}
     decisionTree =DecisionTreeClassifier(**param)
```

```
[ ]: decisionTree.fit(X_train, y_train)
```

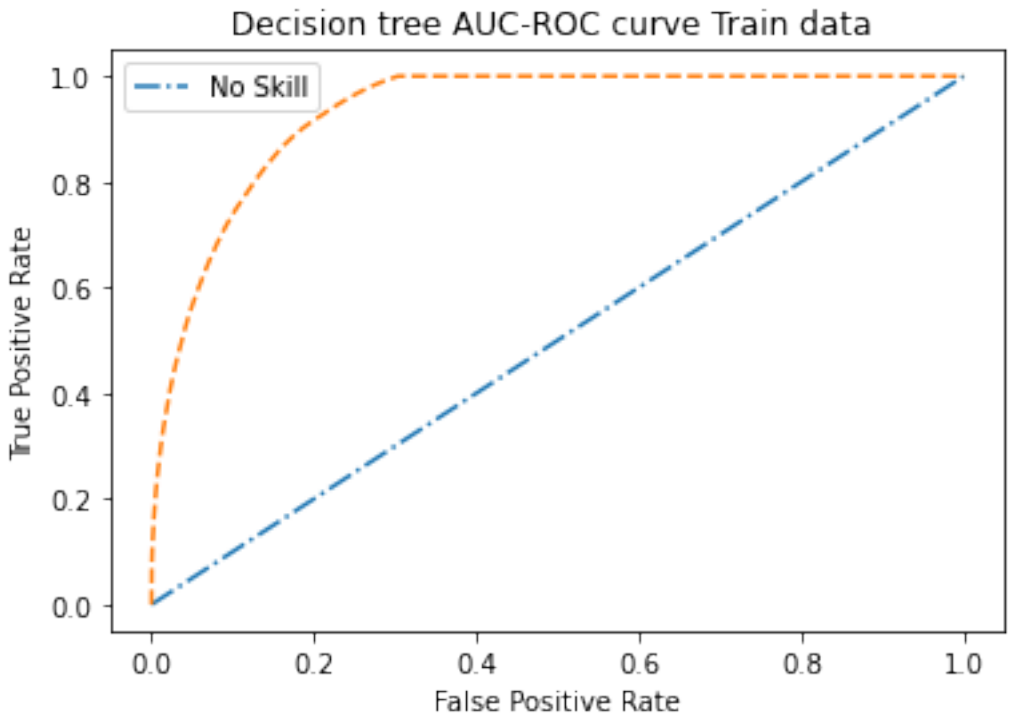
```
[ ]: DecisionTreeClassifier(max_depth=100, min_samples_leaf=10, min_samples_split=10)
```

```
[ ]: # making prediction
     y_train_pred=decisionTree.predict(X_train)
     y_test_pred=decisionTree.predict(X_test)
     y_train_proba=decisionTree.predict_proba(X_train)[:,1]
     y_test_proba=decisionTree.predict_proba(X_test)[:,1]
```

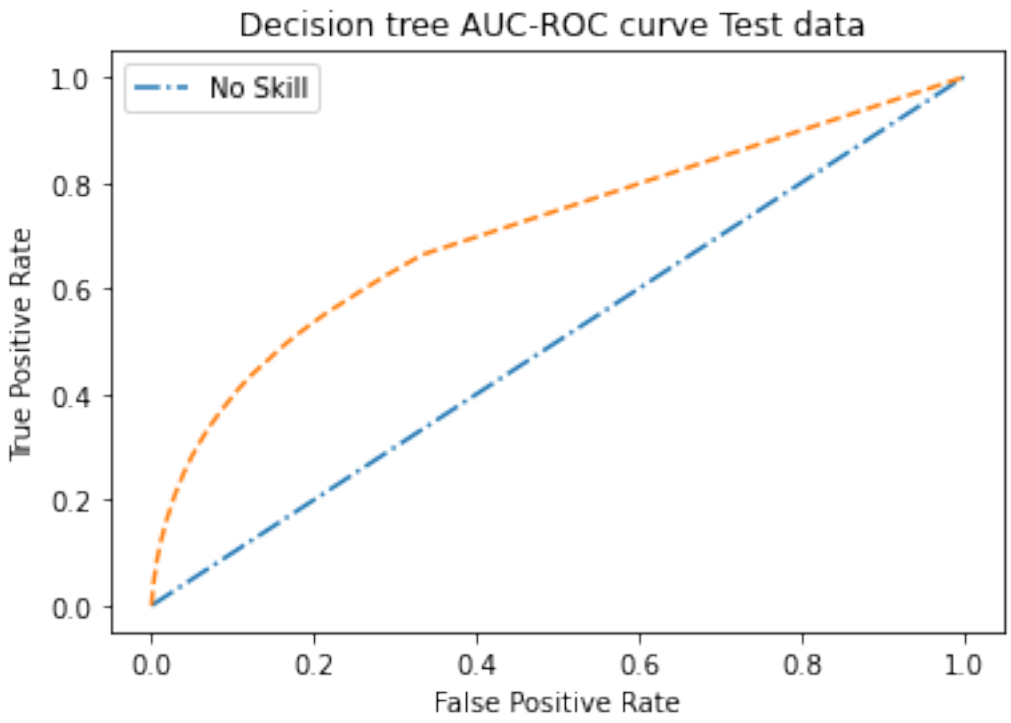
```
[ ]: # printing model performance
     printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  )

     plotAUC_ROC_Curve(y_train ,y_train_proba , title ="Decision tree AUC-ROC curve␣
     ↪Train data ")
     plotAUC_ROC_Curve(y_test ,y_test_proba , title ="Decision tree AUC-ROC curve␣
     ↪Test data ")

     plot_Precision_Recall(y_train ,y_train_proba ,title="Decision tree␣
     ↪Precision-Recall curve Train data")
     plot_Precision_Recall(y_test ,y_test_proba ,title="Decision tree␣
     ↪Precision-Recall curve Test data")
```

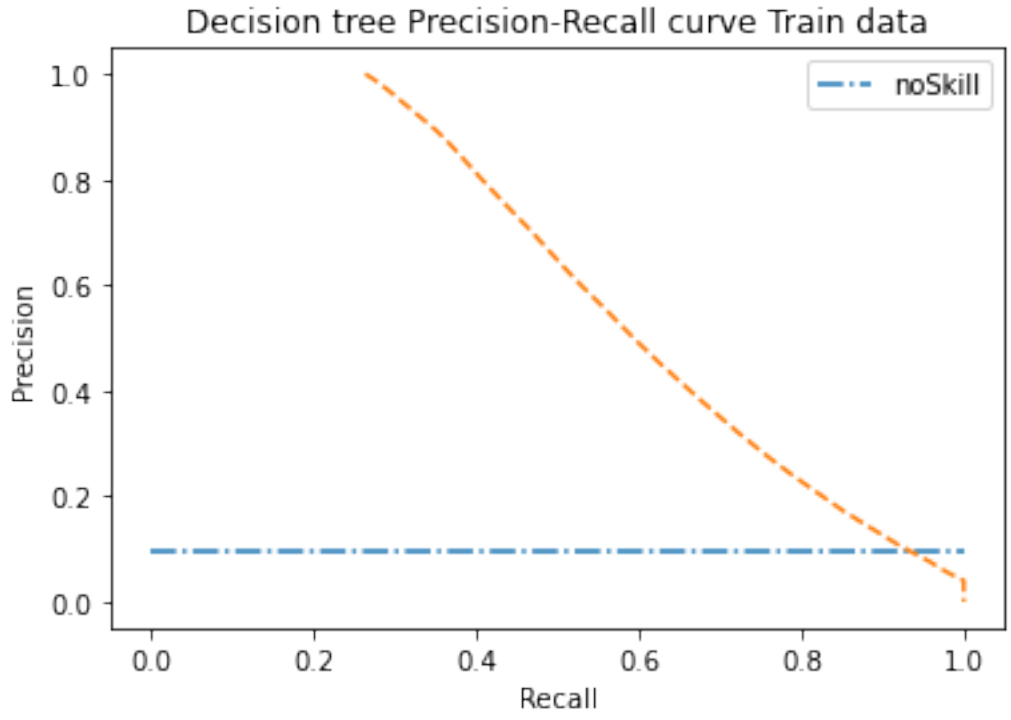
```
Model Performance
Train:
```

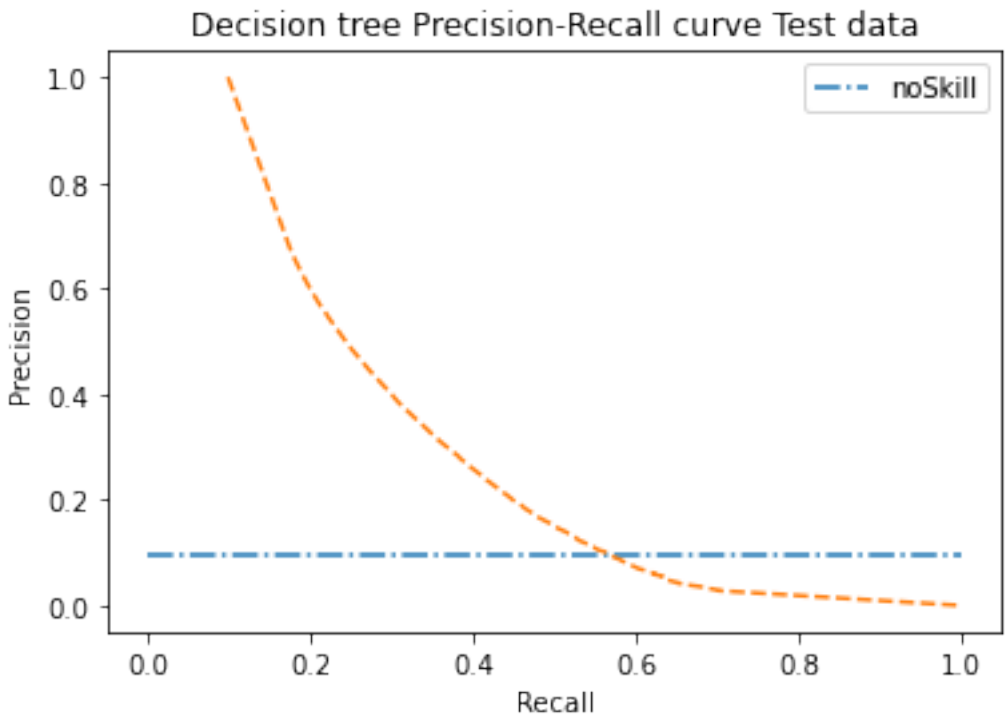
```
True positive rate  0.3330729241289237
False positive rate  0.01456037591702281
False negative  rate  0.6669270758710764
True negative  rate  0.9854396240829771
Precision  0.7126211649909006
Recall 0.3330729241289237
F1 rate 0.4539660646250078
Test:
True positive rate  0.2078424275329533
False positive rate  0.028365751833676876
False negative  rate  0.7921575724670468
True negative  rate  0.9716342481663232
Precision  0.44267560934870037
Recall 0.2078424275329533
F1 rate 0.2828723203362973

AUC score  0.9330946821769641
```




```
AUC score  0.7085283428959789
```


Decision tree AUC-ROC curve Test data
No Skill
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

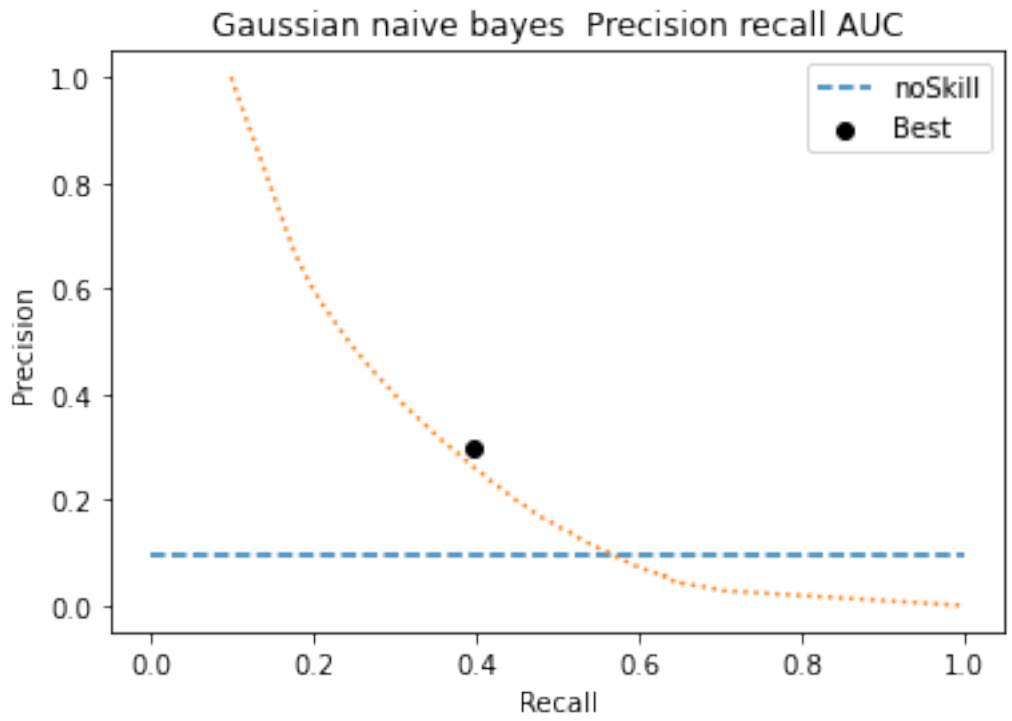

Decision tree Precision-Recall curve Train data
noSkill
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0



- Finding the optimal threshold from precision recall to maximize f1 score

[ ]:
```
thresh_dt=get_optimal_F1(y_test ,y_test_proba, label="Gaussian naive bayes ␣
↪Precision recall AUC")
Model_Perpormance_after_Thresholding(y_train,y_train_proba ,y_test,␣
↪y_test_proba , thresh_dt )
```

```
Best Threshold=0.277778, F-Score=0.342
```




```
Model Performance
Train:
True positive rate  0.6473390754065627
False positive rate  0.0702125617717748
False negative  rate  0.3526609245934374
True negative  rate  0.9297874382282252
Precision  0.49985850556253647
Recall 0.6473390754065627
F1 rate 0.5641189420075571
Test:
True positive rate  0.3939794287093174
False positive rate  0.09859086771368483
False negative  rate  0.6060205712906825
True negative  rate  0.9014091322863151
Precision  0.30225389919933354
Recall 0.3939794287093174
F1 rate 0.3420744562441892
```

5.1.2 Preparing for submission

[ ]:
```
# reading test data
test =pd.read_parquet('test.gzip')
test_temp = test[["order_id", "product_id"]]
```

```
# preparing test data
test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
 ↪,'department_id' ,'aisle_id' ,'ur_pr_count'] , inplace =True)
# standardizing test data
test , test_preprocessing_object=standardize(None, test␣
 ↪,test_preprocessing_object , flag ='test')
# One hot encoding test data
test =ohe_Test(test , test_preprocessing_object )
# making prediction on test data
predict_dt_test =decisionTree.predict_proba(test)[:,1]
```

```
ur_pr_reordered
order_number
ttl_cnt_product_user
Avg_no_prod_perOrder
days_since_prior_order
usr_ro_ratio
product_name_length
(4833292, 3)
(4833292, 24)
```

[ ]:
```
# generate submission csv
submission_log=generate_SubmissionCsv(test_temp ,predict_dt_test ,thresh_dt)

# save submission csv
submission_log.to_csv("./submission/decisionTree2.csv",index=False)
```

5.2 Score after submission

- public score :0.27510
- private score :0.27767

6 Random forest

6.0.1 HyperParameter tunning Random forest

[ ]:
```
distributions = {
'max_depth': [10, 20, 30, 40 ,50, None],
'max_features': ['auto', 'sqrt'],
'min_samples_leaf': [20 , 50,100,150 ,200],
'min_samples_split': [10 , 20 , 50,100,150 ,200],
'n_estimators': [10 ,20,30,40,50,60,70 ]
}
```

[ ]:
```
rf = RandomForestClassifier()
```

[ ]:
```
rf=RandomizedSearchCV(estimator = rf, param_distributions =␣
↪distributions,scoring ='f1', n_iter = 50, cv = 3, random_state=0, n_jobs =␣
↪-1)
```

[ ]:
```
rf.fit(X_train , y_train)
```

6.0.2 After hyper param tunning best param

- {'n_estimators': 20, 'min_samples_split': 10, 'min_samples_leaf': 20, 'max_features': 'sqrt', 'max_depth': 50}

[ ]:
```
rf = RandomForestClassifier(n_estimators= 20, min_samples_split= 10,␣
↪min_samples_leaf= 20, max_features= 'sqrt', max_depth= 50)
rf.fit(X_train, y_train)
```

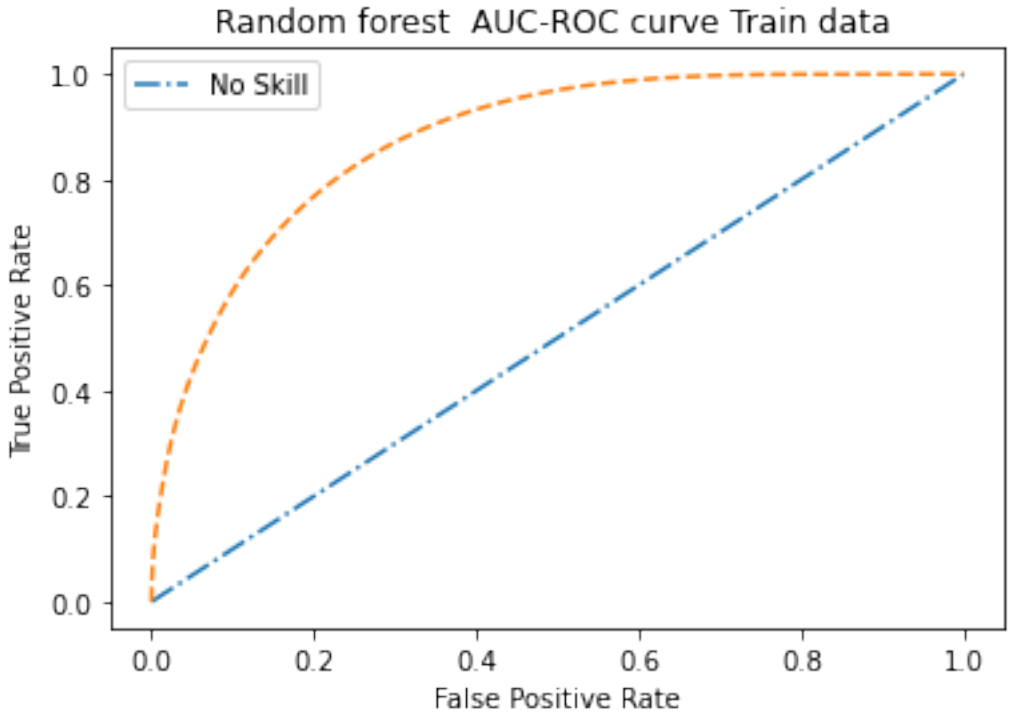
[ ]:
```
RandomForestClassifier(max_depth=50, max_features='sqrt', min_samples_leaf=20,
                       min_samples_split=10, n_estimators=20)
```

[ ]:
```
y_train_pred=rf.predict(X_train)
y_test_pred=rf.predict(X_test)
y_train_proba=rf.predict_proba(X_train)[:,1]
y_test_proba=rf.predict_proba(X_test)[:,1]
```

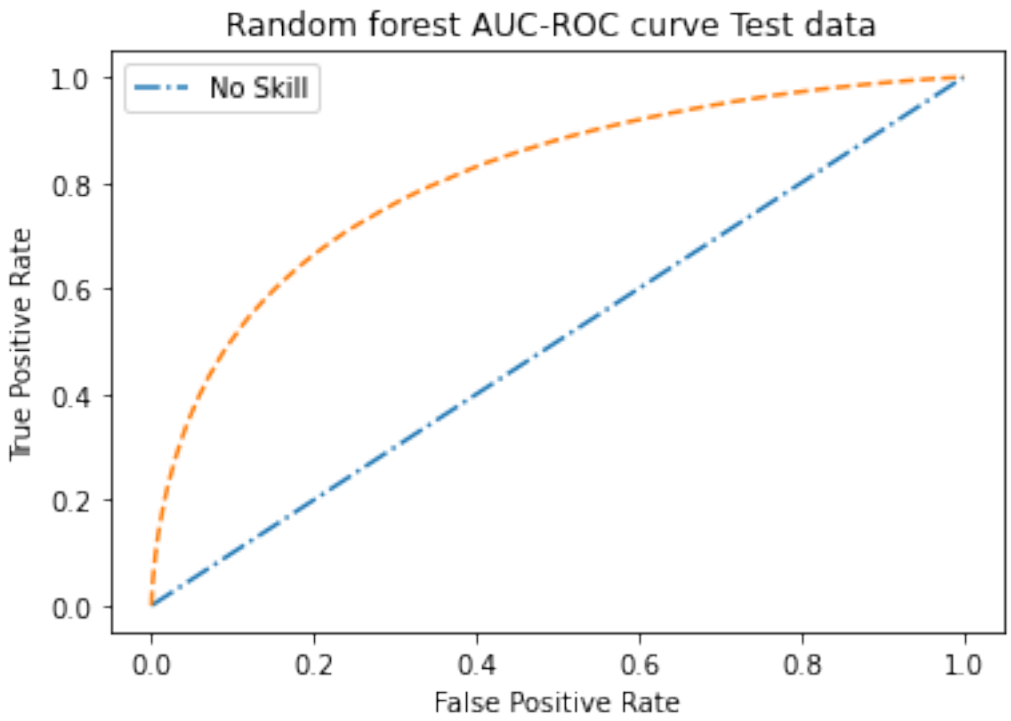
[ ]:
```
printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  )
plotAUC_ROC_Curve(y_train ,y_train_proba , title ="Random forest  AUC-ROC curve␣
↪Train data ")
plotAUC_ROC_Curve(y_test ,y_test_proba , title ="Random forest AUC-ROC curve␣
↪Test data ")
plot_Precision_Recall(y_train ,y_train_proba ,title="Random forest ␣
↪Precision-Recall curve Train data")
plot_Precision_Recall(y_test ,y_test_proba ,title="Random forest ␣
↪Precision-Recall curve Test data")
```

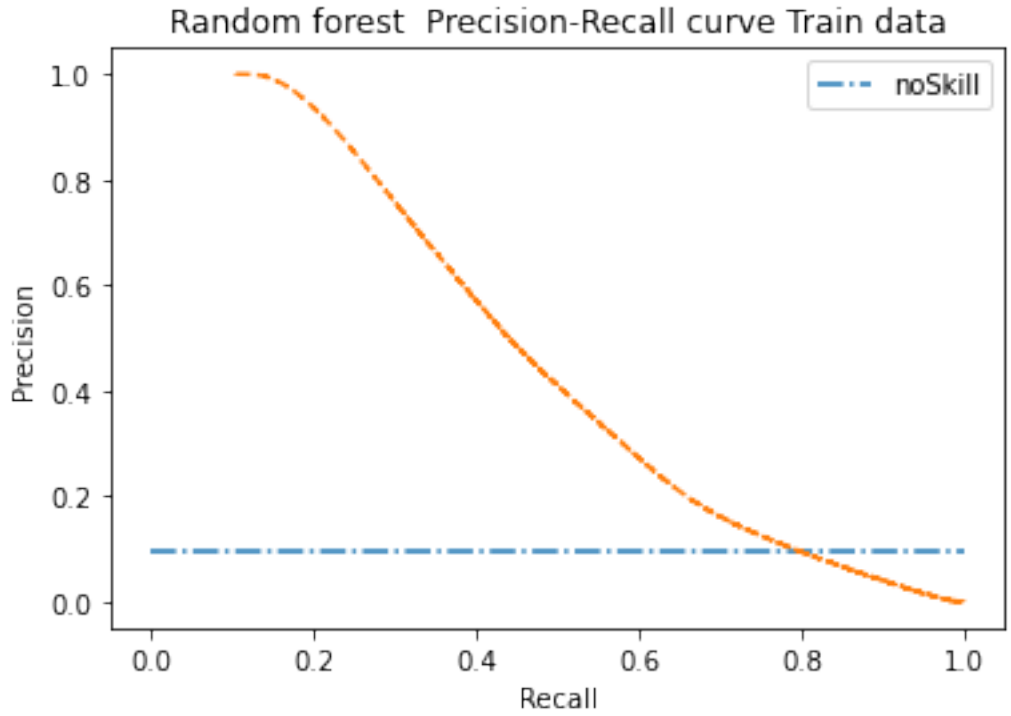
```
Model Performance
Train:
True positive rate  0.16024365855828818
False positive rate  0.007379179746361949
False negative  rate  0.8397563414417119
True negative  rate  0.9926208202536381
Precision  0.7018502249202375
Recall 0.16024365855828818
F1 rate 0.26091600916451185
Test:
True positive rate  0.1428829970138449
False positive rate  0.009075523421886935
False negative  rate  0.8571170029861551
True negative  rate  0.9909244765781131
```

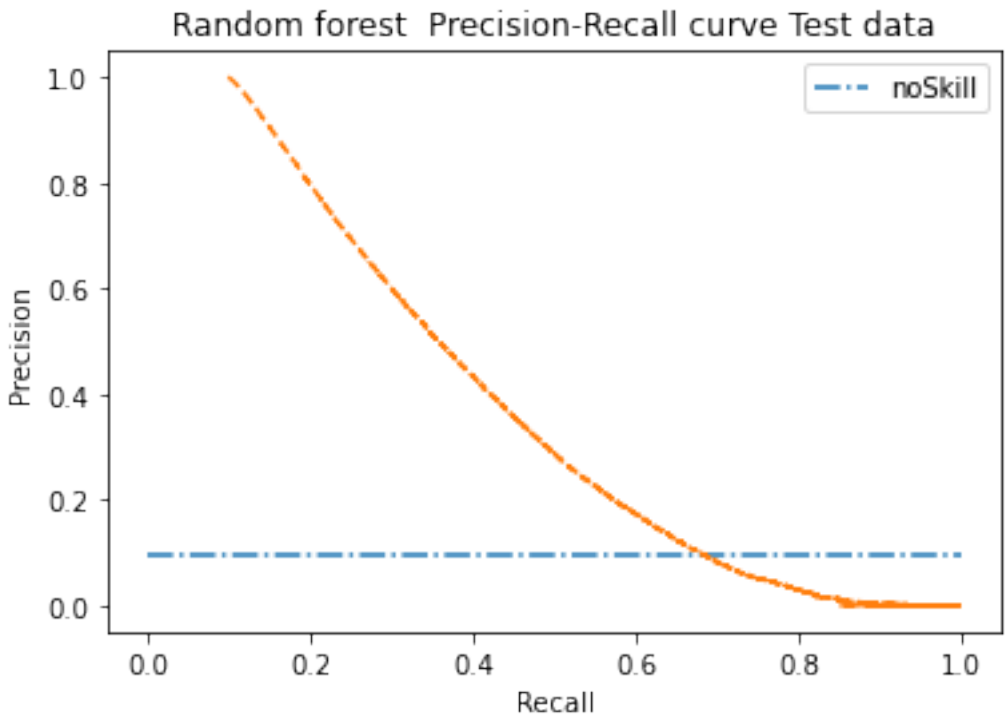
```
Precision  0.630540691638048
Recall 0.1428829970138449
F1 rate 0.23297332389046274

AUC score  0.8745674142279063
```




```
AUC score  0.8077957445905996
```


Random forest AUC-ROC curve Test data
No Skill
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

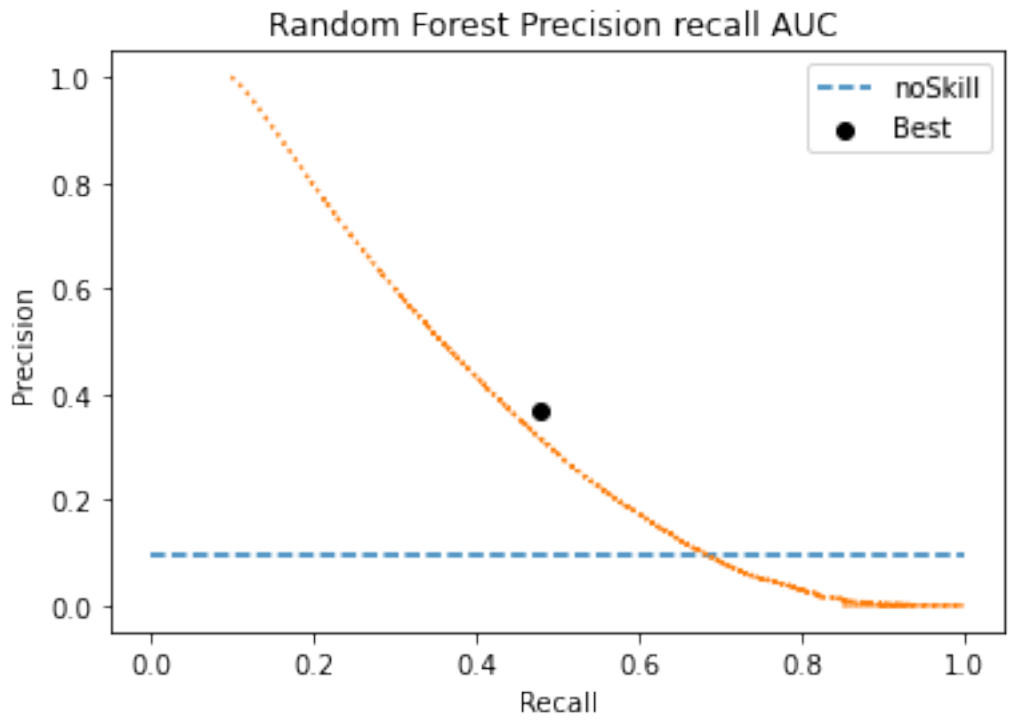

Random forest Precision-Recall curve Train data
noSkill
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0



- Finding the optimal threshold from precision recall to maximize f1 score

[ ]:
```
thresh_rf=get_optimal_F1(y_test ,y_test_proba, label="Random Forest Precision␣
↪recall AUC ")
Model_Perpormance_after_Thresholding(y_train,y_train_proba ,y_test,␣
↪y_test_proba , thresh_rf )
```

```
Best Threshold=0.208533, F-Score=0.418
```




```
Model Performance
Train:
True positive rate  0.5405793451261501
False positive rate  0.08196683521701109
False negative  rate  0.45942065487384987
True negative  rate  0.9180331647829889
Precision  0.416882996045592
Recall 0.5405793451261501
F1 rate 0.47074088934037883
Test:
True positive rate  0.4789430820740205
False positive rate  0.08808907850543564
False negative  rate  0.5210569179259795
True negative  rate  0.9119109214945643
Precision  0.37082603517130247
Recall 0.4789430820740205
F1 rate 0.41800663402306115
```

6.0.3 Prepare for submission

[ ]:
```
# reading test data
test =pd.read_parquet('test.gzip')
test_temp = test[["order_id", "product_id"]]
```

```
# preparing test data
test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
 ↪,'department_id' ,'aisle_id' ,'ur_pr_count'] , inplace =True)
# standardizing test data
test , test_preprocessing_object=standardize(None, test␣
 ↪,test_preprocessing_object , flag ='test')
# One hot encoding test data
test =ohe_Test(test , test_preprocessing_object )
# making prediction on test data
predict_rf_test =rf.predict_proba(test)[:,1]
```

```
ur_pr_reordered
order_number
ttl_cnt_product_user
Avg_no_prod_perOrder
days_since_prior_order
usr_ro_ratio
product_name_length
(4833292, 3)
(4833292, 24)
```

[ ]:
```
# generate submission csv
submission_rf=generate_SubmissionCsv(test_temp,predict_rf_test ,thresh_rf)
# save submission csv
submission_rf.to_csv("./submission/randomforest2.csv",index=False)
```

7 Submission score

- private score :0.34993
- public :0.35191

8 Xgboost

8.0.1 Hypertuning Xgboost

[26]:
```
def optimizeXg(params,X_train =X_train,y_train=y_train , X_test =X_test ,␣
 ↪y_test =y_test):
  kf = StratifiedKFold(n_splits =3)
  print(params)
  xgboost=xgb.XGBClassifier(**params)
  watchlist =[(X_test , y_test)]
  f1s =[]
  for idx in kf.split(X=X_train ,y =y_train):
    train_idx , test_idx=idx[0] ,idx[1]
    xtrain =X_train.iloc[train_idx ,:]
    ytrain =y_train[train_idx]
```

```
    xtest =X_train.iloc[test_idx ,:]
    ytest =y_train[test_idx]


    xgboost.fit(xtrain , ytrain ,eval_set=watchlist, eval_metric␣
 ↪='logloss',early_stopping_rounds=50)

    y_pred_test=xgboost.predict(xtest)

    f1=f1_score(ytest , y_pred_test)
    print(f1)
    f1s.append(f1)

  return -1.0 *np.mean(f1s)
```

[30]:
```
# define parameter space

param_space ={
    'max_depth' :scope.int(hp.quniform('max_depth' ,3 , 12,2)),
    'eta' :hp.uniform('eta' ,0.1 ,0.2 ),
    'subsample': hp.uniform('subsample' , 0.5 ,0.8 ),
    'colsample_bytree' :hp.uniform('colsample_bytree' , 0.5 ,0.8 ),
    'gamma' :scope.int(hp.quniform('gamma' ,0 ,2 ,1)),
    'n_estimators' :scope.int(hp.quniform('n_estimators' ,50 ,5000 ,50)),
    'scale_pos_weight' :scope.int(hp.quniform('scale_pos_weight' ,5,10 ,2)),
    'tree_method':hp.choice('tree_method',['gpu_hist']),
}

optimization_function = partial(optimizeXg ,X_train =X_train,y_train =y_train ,␣
 ↪X_test =X_test , y_test =y_test)
trail = Trials()
results =fmin(fn =optimization_function , space= param_space , algo =tpe.
 ↪suggest  , max_evals=100 ,trials=trail )
```

```
Streaming output truncated to the last 5000 lines.
[72]    validation_0-logloss:0.494626

[73]    validation_0-logloss:0.494594

[74]    validation_0-logloss:0.494542

[75]    validation_0-logloss:0.494437

[76]    validation_0-logloss:0.494527

[77]    validation_0-logloss:0.494367
```

```
[78]    validation_0-logloss:0.494341
[79]    validation_0-logloss:0.494352
[80]    validation_0-logloss:0.494324
[81]    validation_0-logloss:0.49424
[82]    validation_0-logloss:0.494204
[83]    validation_0-logloss:0.494205
[84]    validation_0-logloss:0.494011
[85]    validation_0-logloss:0.493939
[86]    validation_0-logloss:0.493869
[87]    validation_0-logloss:0.493687
[88]    validation_0-logloss:0.493616
[89]    validation_0-logloss:0.493549
[90]    validation_0-logloss:0.493542
[91]    validation_0-logloss:0.493548
[92]    validation_0-logloss:0.493537
[93]    validation_0-logloss:0.493489
[94]    validation_0-logloss:0.493513
[95]    validation_0-logloss:0.493473
[96]    validation_0-logloss:0.493456
[97]    validation_0-logloss:0.493496
[98]    validation_0-logloss:0.49354
[99]    validation_0-logloss:0.493498
[100]   validation_0-logloss:0.49343
[101]   validation_0-logloss:0.493394
```

```
[102]   validation_0-logloss:0.493278
[103]   validation_0-logloss:0.493169
[104]   validation_0-logloss:0.493077
[105]   validation_0-logloss:0.492982
[106]   validation_0-logloss:0.493053
[107]   validation_0-logloss:0.492977
[108]   validation_0-logloss:0.492996
[109]   validation_0-logloss:0.493012
[110]   validation_0-logloss:0.493047
[111]   validation_0-logloss:0.493025
[112]   validation_0-logloss:0.49297
[113]   validation_0-logloss:0.492919
[114]   validation_0-logloss:0.492928
[115]   validation_0-logloss:0.492877
[116]   validation_0-logloss:0.492759
[117]   validation_0-logloss:0.492737
[118]   validation_0-logloss:0.492742
[119]   validation_0-logloss:0.492817
[120]   validation_0-logloss:0.492781
[121]   validation_0-logloss:0.492796
[122]   validation_0-logloss:0.492799
[123]   validation_0-logloss:0.492761
[124]   validation_0-logloss:0.492785
[125]   validation_0-logloss:0.49275
```

```
[126]   validation_0-logloss:0.492736
[127]   validation_0-logloss:0.492815
[128]   validation_0-logloss:0.492823
[129]   validation_0-logloss:0.492787
[130]   validation_0-logloss:0.492791
[131]   validation_0-logloss:0.492804
[132]   validation_0-logloss:0.492743
[133]   validation_0-logloss:0.492766
[134]   validation_0-logloss:0.492748
[135]   validation_0-logloss:0.492658
[136]   validation_0-logloss:0.492594
[137]   validation_0-logloss:0.492529
[138]   validation_0-logloss:0.492501
[139]   validation_0-logloss:0.492448
[140]   validation_0-logloss:0.492428
[141]   validation_0-logloss:0.492335
[142]   validation_0-logloss:0.492333
[143]   validation_0-logloss:0.492343
[144]   validation_0-logloss:0.492321
[145]   validation_0-logloss:0.492307
[146]   validation_0-logloss:0.492335
[147]   validation_0-logloss:0.492384
[148]   validation_0-logloss:0.492367
[149]   validation_0-logloss:0.49245
```

```
[150]   validation_0-logloss:0.492475
[151]   validation_0-logloss:0.492465
[152]   validation_0-logloss:0.492418
[153]   validation_0-logloss:0.492361
[154]   validation_0-logloss:0.492312
[155]   validation_0-logloss:0.492259
[156]   validation_0-logloss:0.492267
[157]   validation_0-logloss:0.492319
[158]   validation_0-logloss:0.492348
[159]   validation_0-logloss:0.492268
[160]   validation_0-logloss:0.492281
[161]   validation_0-logloss:0.492324
[162]   validation_0-logloss:0.492302
[163]   validation_0-logloss:0.492309
[164]   validation_0-logloss:0.492273
[165]   validation_0-logloss:0.492216
[166]   validation_0-logloss:0.492178
[167]   validation_0-logloss:0.492213
[168]   validation_0-logloss:0.492181
[169]   validation_0-logloss:0.492184
[170]   validation_0-logloss:0.492197
[171]   validation_0-logloss:0.492178
[172]   validation_0-logloss:0.492162
[173]   validation_0-logloss:0.492179
```

```
[174]   validation_0-logloss:0.4922
[175]   validation_0-logloss:0.492206
[176]   validation_0-logloss:0.492188
[177]   validation_0-logloss:0.492197
[178]   validation_0-logloss:0.492206
[179]   validation_0-logloss:0.492165
[180]   validation_0-logloss:0.492175
[181]   validation_0-logloss:0.492137
[182]   validation_0-logloss:0.492128
[183]   validation_0-logloss:0.492129
[184]   validation_0-logloss:0.492108
[185]   validation_0-logloss:0.492071
[186]   validation_0-logloss:0.492046
[187]   validation_0-logloss:0.492025
[188]   validation_0-logloss:0.491947
[189]   validation_0-logloss:0.491953
[190]   validation_0-logloss:0.491931
[191]   validation_0-logloss:0.491978
[192]   validation_0-logloss:0.491942
[193]   validation_0-logloss:0.491962
[194]   validation_0-logloss:0.49195
[195]   validation_0-logloss:0.491928
[196]   validation_0-logloss:0.491907
[197]   validation_0-logloss:0.491889
```

```
[198]   validation_0-logloss:0.49187
[199]   validation_0-logloss:0.491862
[200]   validation_0-logloss:0.491895
[201]   validation_0-logloss:0.491871
[202]   validation_0-logloss:0.491837
[203]   validation_0-logloss:0.491801
[204]   validation_0-logloss:0.491795
[205]   validation_0-logloss:0.491874
[206]   validation_0-logloss:0.491828
[207]   validation_0-logloss:0.491878
[208]   validation_0-logloss:0.49187
[209]   validation_0-logloss:0.491858
[210]   validation_0-logloss:0.49183
[211]   validation_0-logloss:0.491925
[212]   validation_0-logloss:0.491878
[213]   validation_0-logloss:0.49193
[214]   validation_0-logloss:0.491925
[215]   validation_0-logloss:0.491959
[216]   validation_0-logloss:0.491954
[217]   validation_0-logloss:0.49196
[218]   validation_0-logloss:0.491988
[219]   validation_0-logloss:0.491977
[220]   validation_0-logloss:0.49196
[221]   validation_0-logloss:0.491952
```

```
[222]   validation_0-logloss:0.491929
[223]   validation_0-logloss:0.49188
[224]   validation_0-logloss:0.491834
[225]   validation_0-logloss:0.491851
[226]   validation_0-logloss:0.491897
[227]   validation_0-logloss:0.491887
[228]   validation_0-logloss:0.491922
[229]   validation_0-logloss:0.491923
[230]   validation_0-logloss:0.491921
[231]   validation_0-logloss:0.491871
[232]   validation_0-logloss:0.491866
[233]   validation_0-logloss:0.491869
[234]   validation_0-logloss:0.491856
[235]   validation_0-logloss:0.491794
[236]   validation_0-logloss:0.491774
[237]   validation_0-logloss:0.491718
[238]   validation_0-logloss:0.491719
[239]   validation_0-logloss:0.491717
[240]   validation_0-logloss:0.4917
[241]   validation_0-logloss:0.491698
[242]   validation_0-logloss:0.491708
[243]   validation_0-logloss:0.491779
[244]   validation_0-logloss:0.49176
[245]   validation_0-logloss:0.49174
```

```
[246]   validation_0-logloss:0.491679
[247]   validation_0-logloss:0.491642
[248]   validation_0-logloss:0.491574
[249]   validation_0-logloss:0.491578
[250]   validation_0-logloss:0.491586
[251]   validation_0-logloss:0.491651
[252]   validation_0-logloss:0.491674
[253]   validation_0-logloss:0.49166
[254]   validation_0-logloss:0.491604
[255]   validation_0-logloss:0.491597
[256]   validation_0-logloss:0.491624
[257]   validation_0-logloss:0.491631
[258]   validation_0-logloss:0.491599
[259]   validation_0-logloss:0.491547
[260]   validation_0-logloss:0.491481
[261]   validation_0-logloss:0.491439
[262]   validation_0-logloss:0.491336
[263]   validation_0-logloss:0.491255
[264]   validation_0-logloss:0.491318
[265]   validation_0-logloss:0.491299
[266]   validation_0-logloss:0.491297
[267]   validation_0-logloss:0.491329
[268]   validation_0-logloss:0.491356
[269]   validation_0-logloss:0.491325
```

```
[270]   validation_0-logloss:0.491344
[271]   validation_0-logloss:0.491377
[272]   validation_0-logloss:0.491458
[273]   validation_0-logloss:0.491472
[274]   validation_0-logloss:0.491496
[275]   validation_0-logloss:0.491559
[276]   validation_0-logloss:0.491654
[277]   validation_0-logloss:0.491628
[278]   validation_0-logloss:0.491599
[279]   validation_0-logloss:0.491614
[280]   validation_0-logloss:0.491643
[281]   validation_0-logloss:0.491614
[282]   validation_0-logloss:0.491638
[283]   validation_0-logloss:0.491573
[284]   validation_0-logloss:0.491568
[285]   validation_0-logloss:0.49158
[286]   validation_0-logloss:0.491471
[287]   validation_0-logloss:0.491519
[288]   validation_0-logloss:0.491515
[289]   validation_0-logloss:0.4915
[290]   validation_0-logloss:0.491547
[291]   validation_0-logloss:0.491534
[292]   validation_0-logloss:0.491537
[293]   validation_0-logloss:0.491509
```

```
[294]   validation_0-logloss:0.491504

[295]   validation_0-logloss:0.491513

[296]   validation_0-logloss:0.491529

[297]   validation_0-logloss:0.49152

[298]   validation_0-logloss:0.491556

[299]   validation_0-logloss:0.491564

[300]   validation_0-logloss:0.491512

[301]   validation_0-logloss:0.491528

[302]   validation_0-logloss:0.491517

[303]   validation_0-logloss:0.491529

[304]   validation_0-logloss:0.491578

[305]   validation_0-logloss:0.491651

[306]   validation_0-logloss:0.491636

[307]   validation_0-logloss:0.49161

[308]   validation_0-logloss:0.491571

[309]   validation_0-logloss:0.491594

[310]   validation_0-logloss:0.491532

[311]   validation_0-logloss:0.49153

[312]   validation_0-logloss:0.491526

[313]   validation_0-logloss:0.491519

Stopping. Best iteration:
[263]   validation_0-logloss:0.491255


0.3700813008130081
{'colsample_bytree': 0.5400075146780988, 'eta': 0.13987248831736387, 'gamma': 1,
'max_depth': 6, 'n_estimators': 750, 'scale_pos_weight': 6, 'subsample':
```

```
0.7504911360406542, 'tree_method': 'gpu_hist'}
[0]     validation_0-logloss:0.657309

Will train until validation_0-logloss hasn't improved in 50 rounds.

[1]     validation_0-logloss:0.627355

[2]     validation_0-logloss:0.60217

[3]     validation_0-logloss:0.586902

[4]     validation_0-logloss:0.567001

[5]     validation_0-logloss:0.555466

[6]     validation_0-logloss:0.541213

[7]     validation_0-logloss:0.526329

[8]     validation_0-logloss:0.519361

[9]     validation_0-logloss:0.507942

[10]    validation_0-logloss:0.499746

[11]    validation_0-logloss:0.490508

[12]    validation_0-logloss:0.48357

[13]    validation_0-logloss:0.477305

[14]    validation_0-logloss:0.471116

[15]    validation_0-logloss:0.466105

[16]    validation_0-logloss:0.461423

[17]    validation_0-logloss:0.459055

[18]    validation_0-logloss:0.456796

[19]    validation_0-logloss:0.453691

[20]    validation_0-logloss:0.450225

[21]    validation_0-logloss:0.447304

[22]    validation_0-logloss:0.44492
```

```
[23]    validation_0-logloss:0.443583
[24]    validation_0-logloss:0.441422
[25]    validation_0-logloss:0.439588
[26]    validation_0-logloss:0.437984
[27]    validation_0-logloss:0.43655
[28]    validation_0-logloss:0.435107
[29]    validation_0-logloss:0.433872
[30]    validation_0-logloss:0.432852
[31]    validation_0-logloss:0.43181
[32]    validation_0-logloss:0.430924
[33]    validation_0-logloss:0.430347
[34]    validation_0-logloss:0.429831
[35]    validation_0-logloss:0.42922
[36]    validation_0-logloss:0.428886
[37]    validation_0-logloss:0.428387
[38]    validation_0-logloss:0.42784
[39]    validation_0-logloss:0.427587
[40]    validation_0-logloss:0.42716
[41]    validation_0-logloss:0.426763
[42]    validation_0-logloss:0.426513
[43]    validation_0-logloss:0.426223
[44]    validation_0-logloss:0.426019
[45]    validation_0-logloss:0.425771
[46]    validation_0-logloss:0.425503
```

```
[47]    validation_0-logloss:0.425186
[48]    validation_0-logloss:0.424956
[49]    validation_0-logloss:0.42485
[50]    validation_0-logloss:0.424626
[51]    validation_0-logloss:0.424455
[52]    validation_0-logloss:0.424345
[53]    validation_0-logloss:0.424216
[54]    validation_0-logloss:0.424077
[55]    validation_0-logloss:0.424017
[56]    validation_0-logloss:0.423891
[57]    validation_0-logloss:0.423757
[58]    validation_0-logloss:0.423674
[59]    validation_0-logloss:0.4236
[60]    validation_0-logloss:0.423563
[61]    validation_0-logloss:0.423463
[62]    validation_0-logloss:0.423401
[63]    validation_0-logloss:0.423311
[64]    validation_0-logloss:0.423287
[65]    validation_0-logloss:0.423267
[66]    validation_0-logloss:0.423246
[67]    validation_0-logloss:0.423186
[68]    validation_0-logloss:0.423196
[69]    validation_0-logloss:0.42314
[70]    validation_0-logloss:0.423092
```

```
[71]    validation_0-logloss:0.423083
[72]    validation_0-logloss:0.423082
[73]    validation_0-logloss:0.423034
[74]    validation_0-logloss:0.422986
[75]    validation_0-logloss:0.422947
[76]    validation_0-logloss:0.422942
[77]    validation_0-logloss:0.422889
[78]    validation_0-logloss:0.42292
[79]    validation_0-logloss:0.422878
[80]    validation_0-logloss:0.422857
[81]    validation_0-logloss:0.422803
[82]    validation_0-logloss:0.422749
[83]    validation_0-logloss:0.422737
[84]    validation_0-logloss:0.422689
[85]    validation_0-logloss:0.422695
[86]    validation_0-logloss:0.422662
[87]    validation_0-logloss:0.422688
[88]    validation_0-logloss:0.42267
[89]    validation_0-logloss:0.422681
[90]    validation_0-logloss:0.422702
[91]    validation_0-logloss:0.422717
[92]    validation_0-logloss:0.422684
[93]    validation_0-logloss:0.42264
[94]    validation_0-logloss:0.422643
```

```
[95]    validation_0-logloss:0.422629
[96]    validation_0-logloss:0.42259
[97]    validation_0-logloss:0.42259
[98]    validation_0-logloss:0.422587
[99]    validation_0-logloss:0.422577
[100]   validation_0-logloss:0.422563
[101]   validation_0-logloss:0.422563
[102]   validation_0-logloss:0.422553
[103]   validation_0-logloss:0.422486
[104]   validation_0-logloss:0.422438
[105]   validation_0-logloss:0.422446
[106]   validation_0-logloss:0.422427
[107]   validation_0-logloss:0.422442
[108]   validation_0-logloss:0.422432
[109]   validation_0-logloss:0.422376
[110]   validation_0-logloss:0.422377
[111]   validation_0-logloss:0.422401
[112]   validation_0-logloss:0.422412
[113]   validation_0-logloss:0.422392
[114]   validation_0-logloss:0.422343
[115]   validation_0-logloss:0.422327
[116]   validation_0-logloss:0.42234
[117]   validation_0-logloss:0.422351
[118]   validation_0-logloss:0.422294
```

```
[119]   validation_0-logloss:0.422291
[120]   validation_0-logloss:0.422279
[121]   validation_0-logloss:0.422265
[122]   validation_0-logloss:0.422261
[123]   validation_0-logloss:0.422245
[124]   validation_0-logloss:0.422241
[125]   validation_0-logloss:0.42223
[126]   validation_0-logloss:0.42218
[127]   validation_0-logloss:0.422184
[128]   validation_0-logloss:0.422172
[129]   validation_0-logloss:0.422181
[130]   validation_0-logloss:0.422134
[131]   validation_0-logloss:0.422135
[132]   validation_0-logloss:0.422121
[133]   validation_0-logloss:0.42213
[134]   validation_0-logloss:0.422069
[135]   validation_0-logloss:0.422098
[136]   validation_0-logloss:0.422093
[137]   validation_0-logloss:0.422065
[138]   validation_0-logloss:0.422025
[139]   validation_0-logloss:0.421994
[140]   validation_0-logloss:0.421992
[141]   validation_0-logloss:0.421991
[142]   validation_0-logloss:0.421949
```

```
[143]   validation_0-logloss:0.42192
[144]   validation_0-logloss:0.421943
[145]   validation_0-logloss:0.421976
[146]   validation_0-logloss:0.422021
[147]   validation_0-logloss:0.422021
[148]   validation_0-logloss:0.422027
[149]   validation_0-logloss:0.422024
[150]   validation_0-logloss:0.422031
[151]   validation_0-logloss:0.422022
[152]   validation_0-logloss:0.422046
[153]   validation_0-logloss:0.422007
[154]   validation_0-logloss:0.421998
[155]   validation_0-logloss:0.421965
[156]   validation_0-logloss:0.421951
[157]   validation_0-logloss:0.421939
[158]   validation_0-logloss:0.421886
[159]   validation_0-logloss:0.421883
[160]   validation_0-logloss:0.421886
[161]   validation_0-logloss:0.421891
[162]   validation_0-logloss:0.421878
[163]   validation_0-logloss:0.421829
[164]   validation_0-logloss:0.42183
[165]   validation_0-logloss:0.421822
[166]   validation_0-logloss:0.421858
```

```
[167]   validation_0-logloss:0.421859
[168]   validation_0-logloss:0.421897
[169]   validation_0-logloss:0.421834
[170]   validation_0-logloss:0.421832
[171]   validation_0-logloss:0.421824
[172]   validation_0-logloss:0.421811
[173]   validation_0-logloss:0.421821
[174]   validation_0-logloss:0.421814
[175]   validation_0-logloss:0.421786
[176]   validation_0-logloss:0.421773
[177]   validation_0-logloss:0.421728
[178]   validation_0-logloss:0.421763
[179]   validation_0-logloss:0.421759
[180]   validation_0-logloss:0.421713
[181]   validation_0-logloss:0.421696
[182]   validation_0-logloss:0.421693
[183]   validation_0-logloss:0.42171
[184]   validation_0-logloss:0.421689
[185]   validation_0-logloss:0.421668
[186]   validation_0-logloss:0.421668
[187]   validation_0-logloss:0.421698
[188]   validation_0-logloss:0.42171
[189]   validation_0-logloss:0.421671
[190]   validation_0-logloss:0.421691
```

```
[191]   validation_0-logloss:0.421662
[192]   validation_0-logloss:0.421616
[193]   validation_0-logloss:0.421572
[194]   validation_0-logloss:0.421582
[195]   validation_0-logloss:0.421597
[196]   validation_0-logloss:0.421594
[197]   validation_0-logloss:0.421582
[198]   validation_0-logloss:0.421618
[199]   validation_0-logloss:0.421582
[200]   validation_0-logloss:0.421607
[201]   validation_0-logloss:0.421605
[202]   validation_0-logloss:0.421595
[203]   validation_0-logloss:0.421562
[204]   validation_0-logloss:0.42156
[205]   validation_0-logloss:0.421573
[206]   validation_0-logloss:0.421561
[207]   validation_0-logloss:0.421565
[208]   validation_0-logloss:0.421555
[209]   validation_0-logloss:0.421567
[210]   validation_0-logloss:0.421517
[211]   validation_0-logloss:0.421558
[212]   validation_0-logloss:0.421532
[213]   validation_0-logloss:0.421521
[214]   validation_0-logloss:0.421551
```

```
[215]   validation_0-logloss:0.421546
[216]   validation_0-logloss:0.421521
[217]   validation_0-logloss:0.421506
[218]   validation_0-logloss:0.42149
[219]   validation_0-logloss:0.421477
[220]   validation_0-logloss:0.421494
[221]   validation_0-logloss:0.42149
[222]   validation_0-logloss:0.421481
[223]   validation_0-logloss:0.421482
[224]   validation_0-logloss:0.421446
[225]   validation_0-logloss:0.42142
[226]   validation_0-logloss:0.421412
[227]   validation_0-logloss:0.421398
[228]   validation_0-logloss:0.42135
[229]   validation_0-logloss:0.421351
[230]   validation_0-logloss:0.421336
[231]   validation_0-logloss:0.421298
[232]   validation_0-logloss:0.421255
[233]   validation_0-logloss:0.421224
[234]   validation_0-logloss:0.421238
[235]   validation_0-logloss:0.421223
[236]   validation_0-logloss:0.421203
[237]   validation_0-logloss:0.421185
[238]   validation_0-logloss:0.421194
```

```
[239]   validation_0-logloss:0.421202
[240]   validation_0-logloss:0.421189
[241]   validation_0-logloss:0.421229
[242]   validation_0-logloss:0.421223
[243]   validation_0-logloss:0.421214
[244]   validation_0-logloss:0.42122
[245]   validation_0-logloss:0.421177
[246]   validation_0-logloss:0.421178
[247]   validation_0-logloss:0.421167
[248]   validation_0-logloss:0.421184
[249]   validation_0-logloss:0.421184
[250]   validation_0-logloss:0.421214
[251]   validation_0-logloss:0.421242
[252]   validation_0-logloss:0.421268
[253]   validation_0-logloss:0.421277
[254]   validation_0-logloss:0.421296
[255]   validation_0-logloss:0.421287
[256]   validation_0-logloss:0.421234
[257]   validation_0-logloss:0.421244
[258]   validation_0-logloss:0.421239
[259]   validation_0-logloss:0.421242
[260]   validation_0-logloss:0.421206
[261]   validation_0-logloss:0.421204
[262]   validation_0-logloss:0.421182
```

```
[263]   validation_0-logloss:0.4212
[264]   validation_0-logloss:0.421166
[265]   validation_0-logloss:0.421127
[266]   validation_0-logloss:0.421107
[267]   validation_0-logloss:0.421127
[268]   validation_0-logloss:0.421121
[269]   validation_0-logloss:0.421083
[270]   validation_0-logloss:0.421029
[271]   validation_0-logloss:0.421
[272]   validation_0-logloss:0.421044
[273]   validation_0-logloss:0.421039
[274]   validation_0-logloss:0.421039
[275]   validation_0-logloss:0.420996
[276]   validation_0-logloss:0.421009
[277]   validation_0-logloss:0.420985
[278]   validation_0-logloss:0.420995
[279]   validation_0-logloss:0.420993
[280]   validation_0-logloss:0.421024
[281]   validation_0-logloss:0.420986
[282]   validation_0-logloss:0.420982
[283]   validation_0-logloss:0.420967
[284]   validation_0-logloss:0.420914
[285]   validation_0-logloss:0.420864
[286]   validation_0-logloss:0.420841
```

```
[287]   validation_0-logloss:0.420856
[288]   validation_0-logloss:0.420883
[289]   validation_0-logloss:0.420829
[290]   validation_0-logloss:0.42082
[291]   validation_0-logloss:0.420843
[292]   validation_0-logloss:0.4209
[293]   validation_0-logloss:0.420934
[294]   validation_0-logloss:0.420877
[295]   validation_0-logloss:0.420855
[296]   validation_0-logloss:0.420852
[297]   validation_0-logloss:0.420839
[298]   validation_0-logloss:0.420782
[299]   validation_0-logloss:0.420787
[300]   validation_0-logloss:0.420749
[301]   validation_0-logloss:0.420767
[302]   validation_0-logloss:0.420749
[303]   validation_0-logloss:0.420751
[304]   validation_0-logloss:0.420774
[305]   validation_0-logloss:0.42074
[306]   validation_0-logloss:0.420722
[307]   validation_0-logloss:0.420721
[308]   validation_0-logloss:0.420723
[309]   validation_0-logloss:0.42076
[310]   validation_0-logloss:0.420766
```

```
[311]   validation_0-logloss:0.420762
[312]   validation_0-logloss:0.420773
[313]   validation_0-logloss:0.420768
[314]   validation_0-logloss:0.420786
[315]   validation_0-logloss:0.420774
[316]   validation_0-logloss:0.420775
[317]   validation_0-logloss:0.420763
[318]   validation_0-logloss:0.420745
[319]   validation_0-logloss:0.42074
[320]   validation_0-logloss:0.420707
[321]   validation_0-logloss:0.420698
[322]   validation_0-logloss:0.420693
[323]   validation_0-logloss:0.420663
[324]   validation_0-logloss:0.420672
[325]   validation_0-logloss:0.420643
[326]   validation_0-logloss:0.420649
[327]   validation_0-logloss:0.420665
[328]   validation_0-logloss:0.420644
[329]   validation_0-logloss:0.420652
[330]   validation_0-logloss:0.420635
[331]   validation_0-logloss:0.42062
[332]   validation_0-logloss:0.420626
[333]   validation_0-logloss:0.420598
[334]   validation_0-logloss:0.420583
```

```
[335]   validation_0-logloss:0.420566
[336]   validation_0-logloss:0.420561
[337]   validation_0-logloss:0.420519
[338]   validation_0-logloss:0.42055
[339]   validation_0-logloss:0.420534
[340]   validation_0-logloss:0.420527
[341]   validation_0-logloss:0.420497
[342]   validation_0-logloss:0.420499
[343]   validation_0-logloss:0.420511
[344]   validation_0-logloss:0.420495
[345]   validation_0-logloss:0.420479
[346]   validation_0-logloss:0.420471
[347]   validation_0-logloss:0.420501
[348]   validation_0-logloss:0.420505
[349]   validation_0-logloss:0.420468
[350]   validation_0-logloss:0.420474
[351]   validation_0-logloss:0.420436
[352]   validation_0-logloss:0.420442
[353]   validation_0-logloss:0.420431
[354]   validation_0-logloss:0.420436
[355]   validation_0-logloss:0.420395
[356]   validation_0-logloss:0.420346
[357]   validation_0-logloss:0.420331
[358]   validation_0-logloss:0.420333
```

```
[359]   validation_0-logloss:0.42033
[360]   validation_0-logloss:0.420322
[361]   validation_0-logloss:0.420318
[362]   validation_0-logloss:0.420313
[363]   validation_0-logloss:0.420373
[364]   validation_0-logloss:0.420377
[365]   validation_0-logloss:0.420373
[366]   validation_0-logloss:0.420378
[367]   validation_0-logloss:0.420354
[368]   validation_0-logloss:0.420396
[369]   validation_0-logloss:0.420365
[370]   validation_0-logloss:0.420402
[371]   validation_0-logloss:0.420394
[372]   validation_0-logloss:0.420371
[373]   validation_0-logloss:0.420353
[374]   validation_0-logloss:0.420344
[375]   validation_0-logloss:0.420311
[376]   validation_0-logloss:0.42029
[377]   validation_0-logloss:0.420298
[378]   validation_0-logloss:0.42026
[379]   validation_0-logloss:0.420266
[380]   validation_0-logloss:0.42026
[381]   validation_0-logloss:0.420255
[382]   validation_0-logloss:0.420262
```

```
[383]   validation_0-logloss:0.420246
[384]   validation_0-logloss:0.420237
[385]   validation_0-logloss:0.420223
[386]   validation_0-logloss:0.420239
[387]   validation_0-logloss:0.420236
[388]   validation_0-logloss:0.420229
[389]   validation_0-logloss:0.420221
[390]   validation_0-logloss:0.420188
[391]   validation_0-logloss:0.420124
[392]   validation_0-logloss:0.420092
[393]   validation_0-logloss:0.420093
[394]   validation_0-logloss:0.420088
[395]   validation_0-logloss:0.420119
[396]   validation_0-logloss:0.420143
[397]   validation_0-logloss:0.420115
[398]   validation_0-logloss:0.420162
[399]   validation_0-logloss:0.420174
[400]   validation_0-logloss:0.420191
[401]   validation_0-logloss:0.420174
[402]   validation_0-logloss:0.420179
[403]   validation_0-logloss:0.420168
[404]   validation_0-logloss:0.420152
[405]   validation_0-logloss:0.420114
[406]   validation_0-logloss:0.420125
```

```
[407]   validation_0-logloss:0.420131
[408]   validation_0-logloss:0.420088
[409]   validation_0-logloss:0.420063
[410]   validation_0-logloss:0.420039
[411]   validation_0-logloss:0.420038
[412]   validation_0-logloss:0.420025
[413]   validation_0-logloss:0.420022
[414]   validation_0-logloss:0.42001
[415]   validation_0-logloss:0.420014
[416]   validation_0-logloss:0.420016
[417]   validation_0-logloss:0.420006
[418]   validation_0-logloss:0.419972
[419]   validation_0-logloss:0.419992
[420]   validation_0-logloss:0.419965
[421]   validation_0-logloss:0.419974
[422]   validation_0-logloss:0.419998
[423]   validation_0-logloss:0.419988
[424]   validation_0-logloss:0.419975
[425]   validation_0-logloss:0.419981
[426]   validation_0-logloss:0.419977
[427]   validation_0-logloss:0.41997
[428]   validation_0-logloss:0.419953
[429]   validation_0-logloss:0.419902
[430]   validation_0-logloss:0.419916
```

```
[431]   validation_0-logloss:0.419936
[432]   validation_0-logloss:0.419929
[433]   validation_0-logloss:0.419933
[434]   validation_0-logloss:0.41991
[435]   validation_0-logloss:0.419932
[436]   validation_0-logloss:0.419926
[437]   validation_0-logloss:0.419917
[438]   validation_0-logloss:0.419919
[439]   validation_0-logloss:0.419891
[440]   validation_0-logloss:0.419867
[441]   validation_0-logloss:0.419851
[442]   validation_0-logloss:0.419818
[443]   validation_0-logloss:0.419844
[444]   validation_0-logloss:0.419856
[445]   validation_0-logloss:0.419897
[446]   validation_0-logloss:0.419865
[447]   validation_0-logloss:0.41986
[448]   validation_0-logloss:0.41982
[449]   validation_0-logloss:0.419824
[450]   validation_0-logloss:0.419784
[451]   validation_0-logloss:0.419796
[452]   validation_0-logloss:0.419735
[453]   validation_0-logloss:0.419728
[454]   validation_0-logloss:0.419725
```

```
[455]   validation_0-logloss:0.419727
[456]   validation_0-logloss:0.419714
[457]   validation_0-logloss:0.419741
[458]   validation_0-logloss:0.41975
[459]   validation_0-logloss:0.41973
[460]   validation_0-logloss:0.419724
[461]   validation_0-logloss:0.419724
[462]   validation_0-logloss:0.419722
[463]   validation_0-logloss:0.41972
[464]   validation_0-logloss:0.419698
[465]   validation_0-logloss:0.419693
[466]   validation_0-logloss:0.419685
[467]   validation_0-logloss:0.419663
[468]   validation_0-logloss:0.419666
[469]   validation_0-logloss:0.419656
[470]   validation_0-logloss:0.419649
[471]   validation_0-logloss:0.41962
[472]   validation_0-logloss:0.419653
[473]   validation_0-logloss:0.419617
[474]   validation_0-logloss:0.419606
[475]   validation_0-logloss:0.419597
[476]   validation_0-logloss:0.419587
[477]   validation_0-logloss:0.419595
[478]   validation_0-logloss:0.419608
```

```
[479]   validation_0-logloss:0.419581
[480]   validation_0-logloss:0.419557
[481]   validation_0-logloss:0.419563
[482]   validation_0-logloss:0.419599
[483]   validation_0-logloss:0.419618
[484]   validation_0-logloss:0.419577
[485]   validation_0-logloss:0.419548
[486]   validation_0-logloss:0.419535
[487]   validation_0-logloss:0.419503
[488]   validation_0-logloss:0.419505
[489]   validation_0-logloss:0.419432
[490]   validation_0-logloss:0.419465
[491]   validation_0-logloss:0.419498
[492]   validation_0-logloss:0.419496
[493]   validation_0-logloss:0.419493
[494]   validation_0-logloss:0.419476
[495]   validation_0-logloss:0.419473
[496]   validation_0-logloss:0.419478
[497]   validation_0-logloss:0.419498
[498]   validation_0-logloss:0.419479
[499]   validation_0-logloss:0.419442
[500]   validation_0-logloss:0.419452
[501]   validation_0-logloss:0.419478
[502]   validation_0-logloss:0.419462
```

```
[503]   validation_0-logloss:0.419491
[504]   validation_0-logloss:0.419464
[505]   validation_0-logloss:0.419448
[506]   validation_0-logloss:0.419443
[507]   validation_0-logloss:0.419449
[508]   validation_0-logloss:0.419431
[509]   validation_0-logloss:0.419402
[510]   validation_0-logloss:0.419359
[511]   validation_0-logloss:0.419352
[512]   validation_0-logloss:0.419387
[513]   validation_0-logloss:0.419388
[514]   validation_0-logloss:0.419383
[515]   validation_0-logloss:0.419403
[516]   validation_0-logloss:0.41932
[517]   validation_0-logloss:0.419308
[518]   validation_0-logloss:0.419288
[519]   validation_0-logloss:0.419274
[520]   validation_0-logloss:0.419294
[521]   validation_0-logloss:0.419318
[522]   validation_0-logloss:0.419301
[523]   validation_0-logloss:0.419308
[524]   validation_0-logloss:0.419254
[525]   validation_0-logloss:0.419241
[526]   validation_0-logloss:0.419214
```

```
[527]   validation_0-logloss:0.419215
[528]   validation_0-logloss:0.419227
[529]   validation_0-logloss:0.419248
[530]   validation_0-logloss:0.419241
[531]   validation_0-logloss:0.419236
[532]   validation_0-logloss:0.419258
[533]   validation_0-logloss:0.419273
[534]   validation_0-logloss:0.41928
[535]   validation_0-logloss:0.419226
[536]   validation_0-logloss:0.419233
[537]   validation_0-logloss:0.419226
[538]   validation_0-logloss:0.419206
[539]   validation_0-logloss:0.419192
[540]   validation_0-logloss:0.419202
[541]   validation_0-logloss:0.41918
[542]   validation_0-logloss:0.419231
[543]   validation_0-logloss:0.419204
[544]   validation_0-logloss:0.419206
[545]   validation_0-logloss:0.419208
[546]   validation_0-logloss:0.419168
[547]   validation_0-logloss:0.419134
[548]   validation_0-logloss:0.419098
[549]   validation_0-logloss:0.419116
[550]   validation_0-logloss:0.419144
```

```
[551]   validation_0-logloss:0.419117
[552]   validation_0-logloss:0.419104
[553]   validation_0-logloss:0.419107
[554]   validation_0-logloss:0.419094
[555]   validation_0-logloss:0.419108
[556]   validation_0-logloss:0.419067
[557]   validation_0-logloss:0.41907
[558]   validation_0-logloss:0.419082
[559]   validation_0-logloss:0.419121
[560]   validation_0-logloss:0.419169
[561]   validation_0-logloss:0.419187
[562]   validation_0-logloss:0.419169
[563]   validation_0-logloss:0.419188
[564]   validation_0-logloss:0.419178
[565]   validation_0-logloss:0.419191
[566]   validation_0-logloss:0.419194
[567]   validation_0-logloss:0.419169
[568]   validation_0-logloss:0.419127
[569]   validation_0-logloss:0.41909
[570]   validation_0-logloss:0.419049
[571]   validation_0-logloss:0.419046
[572]   validation_0-logloss:0.419055
[573]   validation_0-logloss:0.419063
[574]   validation_0-logloss:0.419043
```

```
[575]   validation_0-logloss:0.419036
[576]   validation_0-logloss:0.419054
[577]   validation_0-logloss:0.419014
[578]   validation_0-logloss:0.418996
[579]   validation_0-logloss:0.418967
[580]   validation_0-logloss:0.418962
[581]   validation_0-logloss:0.418987
[582]   validation_0-logloss:0.419001
[583]   validation_0-logloss:0.419007
[584]   validation_0-logloss:0.418981
[585]   validation_0-logloss:0.418989
[586]   validation_0-logloss:0.41898
[587]   validation_0-logloss:0.418916
[588]   validation_0-logloss:0.418926
[589]   validation_0-logloss:0.418891
[590]   validation_0-logloss:0.418906
[591]   validation_0-logloss:0.418884
[592]   validation_0-logloss:0.418899
[593]   validation_0-logloss:0.418932
[594]   validation_0-logloss:0.418938
[595]   validation_0-logloss:0.418908
[596]   validation_0-logloss:0.418883
[597]   validation_0-logloss:0.418904
[598]   validation_0-logloss:0.418873
```

```
[599]   validation_0-logloss:0.418899
[600]   validation_0-logloss:0.418881
[601]   validation_0-logloss:0.418878
[602]   validation_0-logloss:0.418878
[603]   validation_0-logloss:0.418887
[604]   validation_0-logloss:0.418888
[605]   validation_0-logloss:0.418891
[606]   validation_0-logloss:0.41887
[607]   validation_0-logloss:0.418842
[608]   validation_0-logloss:0.418839
[609]   validation_0-logloss:0.418813
[610]   validation_0-logloss:0.418811
[611]   validation_0-logloss:0.41881
[612]   validation_0-logloss:0.418804
[613]   validation_0-logloss:0.418832
[614]   validation_0-logloss:0.418806
[615]   validation_0-logloss:0.418809
[616]   validation_0-logloss:0.418776
[617]   validation_0-logloss:0.418773
[618]   validation_0-logloss:0.418769
[619]   validation_0-logloss:0.418823
[620]   validation_0-logloss:0.418817
[621]   validation_0-logloss:0.418806
[622]   validation_0-logloss:0.418752
```

```
[623]   validation_0-logloss:0.418736
[624]   validation_0-logloss:0.418738
[625]   validation_0-logloss:0.418724
[626]   validation_0-logloss:0.418704
[627]   validation_0-logloss:0.41864
[628]   validation_0-logloss:0.41866
[629]   validation_0-logloss:0.418633
[630]   validation_0-logloss:0.418666
[631]   validation_0-logloss:0.418686
[632]   validation_0-logloss:0.418677
[633]   validation_0-logloss:0.418673
[634]   validation_0-logloss:0.418649
[635]   validation_0-logloss:0.418618
[636]   validation_0-logloss:0.418641
[637]   validation_0-logloss:0.418639
[638]   validation_0-logloss:0.418648
[639]   validation_0-logloss:0.418644
[640]   validation_0-logloss:0.418649
[641]   validation_0-logloss:0.418644
[642]   validation_0-logloss:0.418659
[643]   validation_0-logloss:0.418614
[644]   validation_0-logloss:0.418629
[645]   validation_0-logloss:0.418615
[646]   validation_0-logloss:0.418613
```

```
[647]   validation_0-logloss:0.418633
[648]   validation_0-logloss:0.418675
[649]   validation_0-logloss:0.418679
[650]   validation_0-logloss:0.41867
[651]   validation_0-logloss:0.418673
[652]   validation_0-logloss:0.418667
[653]   validation_0-logloss:0.418706
[654]   validation_0-logloss:0.418698
[655]   validation_0-logloss:0.418721
[656]   validation_0-logloss:0.418696
[657]   validation_0-logloss:0.418716
[658]   validation_0-logloss:0.418661
[659]   validation_0-logloss:0.418633
[660]   validation_0-logloss:0.418607
[661]   validation_0-logloss:0.418609
[662]   validation_0-logloss:0.418557
[663]   validation_0-logloss:0.418552
[664]   validation_0-logloss:0.418495
[665]   validation_0-logloss:0.418505
[666]   validation_0-logloss:0.418517
[667]   validation_0-logloss:0.418516
[668]   validation_0-logloss:0.41851
[669]   validation_0-logloss:0.41854
[670]   validation_0-logloss:0.4185
```

```
[671]   validation_0-logloss:0.418503
[672]   validation_0-logloss:0.418501
[673]   validation_0-logloss:0.418453
[674]   validation_0-logloss:0.418464
[675]   validation_0-logloss:0.418421
[676]   validation_0-logloss:0.418413
[677]   validation_0-logloss:0.418387
[678]   validation_0-logloss:0.418387
[679]   validation_0-logloss:0.418389
[680]   validation_0-logloss:0.418358
[681]   validation_0-logloss:0.418369
[682]   validation_0-logloss:0.418386
[683]   validation_0-logloss:0.418401
[684]   validation_0-logloss:0.4184
[685]   validation_0-logloss:0.418387
[686]   validation_0-logloss:0.418356
[687]   validation_0-logloss:0.418329
[688]   validation_0-logloss:0.418293
[689]   validation_0-logloss:0.418297
[690]   validation_0-logloss:0.418319
[691]   validation_0-logloss:0.4183
[692]   validation_0-logloss:0.418307
[693]   validation_0-logloss:0.418289
[694]   validation_0-logloss:0.418317
```

```
[695]   validation_0-logloss:0.418318
[696]   validation_0-logloss:0.418313
[697]   validation_0-logloss:0.418282
[698]   validation_0-logloss:0.418268
[699]   validation_0-logloss:0.418275
[700]   validation_0-logloss:0.418285
[701]   validation_0-logloss:0.418284
[702]   validation_0-logloss:0.418294
[703]   validation_0-logloss:0.418313
[704]   validation_0-logloss:0.418306
[705]   validation_0-logloss:0.418309
[706]   validation_0-logloss:0.418314
[707]   validation_0-logloss:0.418306
[708]   validation_0-logloss:0.418252
[709]   validation_0-logloss:0.418278
[710]   validation_0-logloss:0.41827
[711]   validation_0-logloss:0.418292
[712]   validation_0-logloss:0.418269
[713]   validation_0-logloss:0.418276
[714]   validation_0-logloss:0.418266
[715]   validation_0-logloss:0.418276
[716]   validation_0-logloss:0.418266
[717]   validation_0-logloss:0.418234
[718]   validation_0-logloss:0.418239
```

```
[719]   validation_0-logloss:0.418248
[720]   validation_0-logloss:0.418277
[721]   validation_0-logloss:0.418315
[722]   validation_0-logloss:0.418295
[723]   validation_0-logloss:0.418248
[724]   validation_0-logloss:0.418216
[725]   validation_0-logloss:0.418197
[726]   validation_0-logloss:0.418189
[727]   validation_0-logloss:0.418184
[728]   validation_0-logloss:0.418191
[729]   validation_0-logloss:0.41815
[730]   validation_0-logloss:0.418139
[731]   validation_0-logloss:0.418167
[732]   validation_0-logloss:0.418184
[733]   validation_0-logloss:0.418189
[734]   validation_0-logloss:0.418179
[735]   validation_0-logloss:0.418176
[736]   validation_0-logloss:0.418184
[737]   validation_0-logloss:0.418212
[738]   validation_0-logloss:0.418208
[739]   validation_0-logloss:0.418191
[740]   validation_0-logloss:0.418154
[741]   validation_0-logloss:0.41816
[742]   validation_0-logloss:0.418189
```

```
[743]   validation_0-logloss:0.418222

[744]   validation_0-logloss:0.418241

[745]   validation_0-logloss:0.418222

[746]   validation_0-logloss:0.418186

[747]   validation_0-logloss:0.41817

[748]   validation_0-logloss:0.418174

[749]   validation_0-logloss:0.418174

0.3979921860850216
[0]     validation_0-logloss:0.657177

Will train until validation_0-logloss hasn't improved in 50 rounds.

[1]     validation_0-logloss:0.627275

[2]     validation_0-logloss:0.602044

[3]     validation_0-logloss:0.586849

[4]     validation_0-logloss:0.566908

[5]     validation_0-logloss:0.555417

[6]     validation_0-logloss:0.541179

[7]     validation_0-logloss:0.526263

[8]     validation_0-logloss:0.519572

[9]     validation_0-logloss:0.50816

[10]    validation_0-logloss:0.499932

[11]    validation_0-logloss:0.490596

[12]    validation_0-logloss:0.483592

[13]    validation_0-logloss:0.477318

[14]    validation_0-logloss:0.471288
```

```
[15]    validation_0-logloss:0.466205
[16]    validation_0-logloss:0.461435
[17]    validation_0-logloss:0.459083
[18]    validation_0-logloss:0.456837
[19]    validation_0-logloss:0.453727
[20]    validation_0-logloss:0.450264
[21]    validation_0-logloss:0.447346
[22]    validation_0-logloss:0.444994
[23]    validation_0-logloss:0.443602
[24]    validation_0-logloss:0.441491
[25]    validation_0-logloss:0.439603
[26]    validation_0-logloss:0.437982
[27]    validation_0-logloss:0.436517
[28]    validation_0-logloss:0.435106
[29]    validation_0-logloss:0.433863
[30]    validation_0-logloss:0.432817
[31]    validation_0-logloss:0.431638
[32]    validation_0-logloss:0.43081
[33]    validation_0-logloss:0.430169
[34]    validation_0-logloss:0.429616
[35]    validation_0-logloss:0.429005
[36]    validation_0-logloss:0.428713
[37]    validation_0-logloss:0.428275
[38]    validation_0-logloss:0.427731
```

```
[39]    validation_0-logloss:0.427408

[40]    validation_0-logloss:0.426941

[41]    validation_0-logloss:0.426469

[42]    validation_0-logloss:0.426205

[43]    validation_0-logloss:0.425938

[44]    validation_0-logloss:0.425681

[45]    validation_0-logloss:0.425441

[46]    validation_0-logloss:0.425176

[47]    validation_0-logloss:0.424914

[48]    validation_0-logloss:0.424715

[49]    validation_0-logloss:0.424615

[50]    validation_0-logloss:0.424453

[51]    validation_0-logloss:0.424263

[52]    validation_0-logloss:0.424121

[53]    validation_0-logloss:0.423976

[54]    validation_0-logloss:0.423855

[55]    validation_0-logloss:0.423762

[56]    validation_0-logloss:0.423641

[57]    validation_0-logloss:0.423591

[58]    validation_0-logloss:0.423519

[59]    validation_0-logloss:0.423477

[60]    validation_0-logloss:0.423361

[61]    validation_0-logloss:0.423338

[62]    validation_0-logloss:0.423314
```

```
[63]    validation_0-logloss:0.423305

[64]    validation_0-logloss:0.423287

[65]    validation_0-logloss:0.423256

[66]    validation_0-logloss:0.423177

[67]    validation_0-logloss:0.423128

[68]    validation_0-logloss:0.423039

[69]    validation_0-logloss:0.423009

[70]    validation_0-logloss:0.422926

[71]    validation_0-logloss:0.422869

[72]    validation_0-logloss:0.422864

[73]    validation_0-logloss:0.422826

[74]    validation_0-logloss:0.422875

[75]    validation_0-logloss:0.422807

[76]    validation_0-logloss:0.422831

[77]    validation_0-logloss:0.422835

[78]    validation_0-logloss:0.422813

[79]    validation_0-logloss:0.422765

[80]    validation_0-logloss:0.422733

[81]    validation_0-logloss:0.422668

[82]    validation_0-logloss:0.422649

[83]    validation_0-logloss:0.422655

[84]    validation_0-logloss:0.422631

[85]    validation_0-logloss:0.422632

[86]    validation_0-logloss:0.422596
```

```
[87]    validation_0-logloss:0.422562
[88]    validation_0-logloss:0.422502
[89]    validation_0-logloss:0.422498
[90]    validation_0-logloss:0.422463
[91]    validation_0-logloss:0.422479
[92]    validation_0-logloss:0.422391
[93]    validation_0-logloss:0.422385
[94]    validation_0-logloss:0.422435
[95]    validation_0-logloss:0.422404
[96]    validation_0-logloss:0.422405
[97]    validation_0-logloss:0.422431
[98]    validation_0-logloss:0.42245
[99]    validation_0-logloss:0.422444
[100]   validation_0-logloss:0.422368
[101]   validation_0-logloss:0.422326
[102]   validation_0-logloss:0.422307
[103]   validation_0-logloss:0.4223
[104]   validation_0-logloss:0.422329
[105]   validation_0-logloss:0.422337
[106]   validation_0-logloss:0.422315
[107]   validation_0-logloss:0.422277
[108]   validation_0-logloss:0.422274
[109]   validation_0-logloss:0.422263
[110]   validation_0-logloss:0.422269
```

```
[111]   validation_0-logloss:0.422201

[112]   validation_0-logloss:0.422221

[113]   validation_0-logloss:0.422229

[114]   validation_0-logloss:0.422257

[115]   validation_0-logloss:0.422243

[116]   validation_0-logloss:0.42228

[117]   validation_0-logloss:0.42225

[118]   validation_0-logloss:0.422216

[119]   validation_0-logloss:0.422205

[120]   validation_0-logloss:0.422181

[121]   validation_0-logloss:0.422156

[122]   validation_0-logloss:0.422144

[123]   validation_0-logloss:0.422094

[124]   validation_0-logloss:0.422081

[125]   validation_0-logloss:0.422083

[126]   validation_0-logloss:0.422117

[127]   validation_0-logloss:0.422085

[128]   validation_0-logloss:0.422064

[129]   validation_0-logloss:0.422025

[130]   validation_0-logloss:0.42203

[131]   validation_0-logloss:0.422018

[132]   validation_0-logloss:0.421991

[133]   validation_0-logloss:0.421966

[134]   validation_0-logloss:0.421988
```

```
[135]   validation_0-logloss:0.421983

[136]   validation_0-logloss:0.421998

[137]   validation_0-logloss:0.422004

[138]   validation_0-logloss:0.421965

[139]   validation_0-logloss:0.421977

[140]   validation_0-logloss:0.421983

[141]   validation_0-logloss:0.421973

[142]   validation_0-logloss:0.421972

[143]   validation_0-logloss:0.42198

[144]   validation_0-logloss:0.421957

[145]   validation_0-logloss:0.421936

[146]   validation_0-logloss:0.421921

[147]   validation_0-logloss:0.421942

[148]   validation_0-logloss:0.421935

[149]   validation_0-logloss:0.421921

[150]   validation_0-logloss:0.421914

[151]   validation_0-logloss:0.421944

[152]   validation_0-logloss:0.421953

[153]   validation_0-logloss:0.421896

[154]   validation_0-logloss:0.42187

[155]   validation_0-logloss:0.421815

[156]   validation_0-logloss:0.421764

[157]   validation_0-logloss:0.421784

[158]   validation_0-logloss:0.421751
```

```
[159]   validation_0-logloss:0.421698

[160]   validation_0-logloss:0.421703

[161]   validation_0-logloss:0.421702

[162]   validation_0-logloss:0.42168

[163]   validation_0-logloss:0.421706

[164]   validation_0-logloss:0.421675

[165]   validation_0-logloss:0.421672

[166]   validation_0-logloss:0.421717

[167]   validation_0-logloss:0.421682

[168]   validation_0-logloss:0.421683

[169]   validation_0-logloss:0.421639

[170]   validation_0-logloss:0.421648

[171]   validation_0-logloss:0.421582

[172]   validation_0-logloss:0.421565

[173]   validation_0-logloss:0.421581

[174]   validation_0-logloss:0.421557

[175]   validation_0-logloss:0.421554

[176]   validation_0-logloss:0.421544

[177]   validation_0-logloss:0.421531

[178]   validation_0-logloss:0.421561

[179]   validation_0-logloss:0.421515

[180]   validation_0-logloss:0.421508

[181]   validation_0-logloss:0.421505

[182]   validation_0-logloss:0.421485
```

```
[183]   validation_0-logloss:0.421514

[184]   validation_0-logloss:0.421523

[185]   validation_0-logloss:0.421481

[186]   validation_0-logloss:0.421514

[187]   validation_0-logloss:0.421515

[188]   validation_0-logloss:0.421515

[189]   validation_0-logloss:0.42149

[190]   validation_0-logloss:0.421465

[191]   validation_0-logloss:0.421464

[192]   validation_0-logloss:0.421423

[193]   validation_0-logloss:0.421416

[194]   validation_0-logloss:0.421407

[195]   validation_0-logloss:0.421406

[196]   validation_0-logloss:0.421334

[197]   validation_0-logloss:0.421333

[198]   validation_0-logloss:0.421338

[199]   validation_0-logloss:0.421345

[200]   validation_0-logloss:0.421335

[201]   validation_0-logloss:0.421316

[202]   validation_0-logloss:0.421329

[203]   validation_0-logloss:0.421318

[204]   validation_0-logloss:0.421317

[205]   validation_0-logloss:0.421329

[206]   validation_0-logloss:0.421309
```

```
[207]   validation_0-logloss:0.421257

[208]   validation_0-logloss:0.421261

[209]   validation_0-logloss:0.421273

[210]   validation_0-logloss:0.421295

[211]   validation_0-logloss:0.421308

[212]   validation_0-logloss:0.421312

[213]   validation_0-logloss:0.421282

[214]   validation_0-logloss:0.42128

[215]   validation_0-logloss:0.421265

[216]   validation_0-logloss:0.421248

[217]   validation_0-logloss:0.421277

[218]   validation_0-logloss:0.421323

[219]   validation_0-logloss:0.421289

[220]   validation_0-logloss:0.421311

[221]   validation_0-logloss:0.421284

[222]   validation_0-logloss:0.421291

[223]   validation_0-logloss:0.42128

[224]   validation_0-logloss:0.421206

[225]   validation_0-logloss:0.421246

[226]   validation_0-logloss:0.421262

[227]   validation_0-logloss:0.421233

[228]   validation_0-logloss:0.421194

[229]   validation_0-logloss:0.421193

[230]   validation_0-logloss:0.421221
```

```
[231]   validation_0-logloss:0.42117
[232]   validation_0-logloss:0.421163
[233]   validation_0-logloss:0.421158
[234]   validation_0-logloss:0.421151
[235]   validation_0-logloss:0.421163
[236]   validation_0-logloss:0.421164
[237]   validation_0-logloss:0.421162
[238]   validation_0-logloss:0.421215
[239]   validation_0-logloss:0.421213
[240]   validation_0-logloss:0.421216
[241]   validation_0-logloss:0.421195
[242]   validation_0-logloss:0.421175
[243]   validation_0-logloss:0.421163
[244]   validation_0-logloss:0.421151
[245]   validation_0-logloss:0.421124
[246]   validation_0-logloss:0.421111
[247]   validation_0-logloss:0.421103
[248]   validation_0-logloss:0.421062
[249]   validation_0-logloss:0.421065
[250]   validation_0-logloss:0.421053
[251]   validation_0-logloss:0.421039
[252]   validation_0-logloss:0.421035
[253]   validation_0-logloss:0.420997
[254]   validation_0-logloss:0.420988
```

```
[255]   validation_0-logloss:0.420953
[256]   validation_0-logloss:0.420956
[257]   validation_0-logloss:0.420962
[258]   validation_0-logloss:0.420944
[259]   validation_0-logloss:0.420895
[260]   validation_0-logloss:0.420933
[261]   validation_0-logloss:0.420913
[262]   validation_0-logloss:0.420867
[263]   validation_0-logloss:0.420843
[264]   validation_0-logloss:0.420807
[265]   validation_0-logloss:0.420822
[266]   validation_0-logloss:0.420833
[267]   validation_0-logloss:0.420821
[268]   validation_0-logloss:0.420797
[269]   validation_0-logloss:0.420781
[270]   validation_0-logloss:0.420773
[271]   validation_0-logloss:0.420786
[272]   validation_0-logloss:0.420789
[273]   validation_0-logloss:0.420792
[274]   validation_0-logloss:0.42079
[275]   validation_0-logloss:0.420777
[276]   validation_0-logloss:0.420835
[277]   validation_0-logloss:0.420831
[278]   validation_0-logloss:0.420855
```

```
[279]   validation_0-logloss:0.420859
[280]   validation_0-logloss:0.420866
[281]   validation_0-logloss:0.420845
[282]   validation_0-logloss:0.420851
[283]   validation_0-logloss:0.420848
[284]   validation_0-logloss:0.420844
[285]   validation_0-logloss:0.420858
[286]   validation_0-logloss:0.420804
[287]   validation_0-logloss:0.420817
[288]   validation_0-logloss:0.420794
[289]   validation_0-logloss:0.420781
[290]   validation_0-logloss:0.420771
[291]   validation_0-logloss:0.420731
[292]   validation_0-logloss:0.420741
[293]   validation_0-logloss:0.420682
[294]   validation_0-logloss:0.420667
[295]   validation_0-logloss:0.420655
[296]   validation_0-logloss:0.420677
[297]   validation_0-logloss:0.420695
[298]   validation_0-logloss:0.420663
[299]   validation_0-logloss:0.420647
[300]   validation_0-logloss:0.420639
[301]   validation_0-logloss:0.420659
[302]   validation_0-logloss:0.420647
```

```
[303]   validation_0-logloss:0.420641
[304]   validation_0-logloss:0.420631
[305]   validation_0-logloss:0.420625
[306]   validation_0-logloss:0.420614
[307]   validation_0-logloss:0.420606
[308]   validation_0-logloss:0.420556
[309]   validation_0-logloss:0.420542
[310]   validation_0-logloss:0.420545
[311]   validation_0-logloss:0.420547
[312]   validation_0-logloss:0.420537
[313]   validation_0-logloss:0.420545
[314]   validation_0-logloss:0.420545
[315]   validation_0-logloss:0.420568
[316]   validation_0-logloss:0.42053
[317]   validation_0-logloss:0.420516
[318]   validation_0-logloss:0.420505
[319]   validation_0-logloss:0.420564
[320]   validation_0-logloss:0.42059
[321]   validation_0-logloss:0.420614
[322]   validation_0-logloss:0.420604
[323]   validation_0-logloss:0.420594
[324]   validation_0-logloss:0.420598
[325]   validation_0-logloss:0.42058
[326]   validation_0-logloss:0.420555
```

```
[327]   validation_0-logloss:0.42056
[328]   validation_0-logloss:0.420541
[329]   validation_0-logloss:0.420496
[330]   validation_0-logloss:0.420481
[331]   validation_0-logloss:0.420465
[332]   validation_0-logloss:0.420444
[333]   validation_0-logloss:0.420464
[334]   validation_0-logloss:0.420451
[335]   validation_0-logloss:0.420445
[336]   validation_0-logloss:0.420486
[337]   validation_0-logloss:0.420462
[338]   validation_0-logloss:0.420481
[339]   validation_0-logloss:0.420465
[340]   validation_0-logloss:0.420467
[341]   validation_0-logloss:0.420449
[342]   validation_0-logloss:0.420438
[343]   validation_0-logloss:0.420408
[344]   validation_0-logloss:0.420398
[345]   validation_0-logloss:0.4204
[346]   validation_0-logloss:0.420404
[347]   validation_0-logloss:0.420397
[348]   validation_0-logloss:0.420378
[349]   validation_0-logloss:0.420369
[350]   validation_0-logloss:0.42037
```

```
[351]   validation_0-logloss:0.420364
[352]   validation_0-logloss:0.420423
[353]   validation_0-logloss:0.420421
[354]   validation_0-logloss:0.420404
[355]   validation_0-logloss:0.420368
[356]   validation_0-logloss:0.420354
[357]   validation_0-logloss:0.420339
[358]   validation_0-logloss:0.420334
[359]   validation_0-logloss:0.420321
[360]   validation_0-logloss:0.42031
[361]   validation_0-logloss:0.420275
[362]   validation_0-logloss:0.420258
[363]   validation_0-logloss:0.420254
[364]   validation_0-logloss:0.420225
[365]   validation_0-logloss:0.420213
[366]   validation_0-logloss:0.420214
[367]   validation_0-logloss:0.420208
[368]   validation_0-logloss:0.420201
[369]   validation_0-logloss:0.420209
[370]   validation_0-logloss:0.420208
[371]   validation_0-logloss:0.420176
[372]   validation_0-logloss:0.420159
[373]   validation_0-logloss:0.420115
[374]   validation_0-logloss:0.420097
```

```
[375]   validation_0-logloss:0.420065
[376]   validation_0-logloss:0.420078
[377]   validation_0-logloss:0.420034
[378]   validation_0-logloss:0.42003
[379]   validation_0-logloss:0.420079
[380]   validation_0-logloss:0.420031
[381]   validation_0-logloss:0.420027
[382]   validation_0-logloss:0.420042
[383]   validation_0-logloss:0.420057
[384]   validation_0-logloss:0.420041
[385]   validation_0-logloss:0.420059
[386]   validation_0-logloss:0.42005
[387]   validation_0-logloss:0.420032
[388]   validation_0-logloss:0.420047
[389]   validation_0-logloss:0.420006
[390]   validation_0-logloss:0.41998
[391]   validation_0-logloss:0.419993
[392]   validation_0-logloss:0.419984
[393]   validation_0-logloss:0.41998
[394]   validation_0-logloss:0.419961
[395]   validation_0-logloss:0.419963
[396]   validation_0-logloss:0.419937
[397]   validation_0-logloss:0.419966
[398]   validation_0-logloss:0.420009
```

```
[399]   validation_0-logloss:0.420007
[400]   validation_0-logloss:0.419991
[401]   validation_0-logloss:0.419981
[402]   validation_0-logloss:0.419984
[403]   validation_0-logloss:0.419967
[404]   validation_0-logloss:0.41993
[405]   validation_0-logloss:0.419929
[406]   validation_0-logloss:0.419922
[407]   validation_0-logloss:0.419941
[408]   validation_0-logloss:0.419938
[409]   validation_0-logloss:0.419956
[410]   validation_0-logloss:0.419947
[411]   validation_0-logloss:0.419938
[412]   validation_0-logloss:0.419938
[413]   validation_0-logloss:0.419938
[414]   validation_0-logloss:0.419921
[415]   validation_0-logloss:0.419925
[416]   validation_0-logloss:0.41994
[417]   validation_0-logloss:0.419953
[418]   validation_0-logloss:0.419925
[419]   validation_0-logloss:0.419886
[420]   validation_0-logloss:0.4199
[421]   validation_0-logloss:0.419902
[422]   validation_0-logloss:0.419864
```

```
[423]   validation_0-logloss:0.419872
[424]   validation_0-logloss:0.419882
[425]   validation_0-logloss:0.41988
[426]   validation_0-logloss:0.419882
[427]   validation_0-logloss:0.419881
[428]   validation_0-logloss:0.419847
[429]   validation_0-logloss:0.419833
[430]   validation_0-logloss:0.419835
[431]   validation_0-logloss:0.419781
[432]   validation_0-logloss:0.419802
[433]   validation_0-logloss:0.419783
[434]   validation_0-logloss:0.419779
[435]   validation_0-logloss:0.419778
[436]   validation_0-logloss:0.419805
[437]   validation_0-logloss:0.419746
[438]   validation_0-logloss:0.419775
[439]   validation_0-logloss:0.419737
[440]   validation_0-logloss:0.419759
[441]   validation_0-logloss:0.419782
[442]   validation_0-logloss:0.419773
[443]   validation_0-logloss:0.419752
[444]   validation_0-logloss:0.419789
[445]   validation_0-logloss:0.419787
[446]   validation_0-logloss:0.419785
```

```
[447]   validation_0-logloss:0.419797

[448]   validation_0-logloss:0.41979

[449]   validation_0-logloss:0.419769

[450]   validation_0-logloss:0.419766

[451]   validation_0-logloss:0.419737

[452]   validation_0-logloss:0.419719

[453]   validation_0-logloss:0.419711

[454]   validation_0-logloss:0.419696

[455]   validation_0-logloss:0.419644

[456]   validation_0-logloss:0.41963

[457]   validation_0-logloss:0.419655

[458]   validation_0-logloss:0.419615

[459]   validation_0-logloss:0.419634

[460]   validation_0-logloss:0.419659

[461]   validation_0-logloss:0.419656

[462]   validation_0-logloss:0.419649

[463]   validation_0-logloss:0.419582

[464]   validation_0-logloss:0.419584

[465]   validation_0-logloss:0.419557

[466]   validation_0-logloss:0.419541

[467]   validation_0-logloss:0.41955

[468]   validation_0-logloss:0.419524

[469]   validation_0-logloss:0.419526

[470]   validation_0-logloss:0.41949
```

```
[471]   validation_0-logloss:0.419504

[472]   validation_0-logloss:0.419505

[473]   validation_0-logloss:0.419531

[474]   validation_0-logloss:0.419514

[475]   validation_0-logloss:0.419511

[476]   validation_0-logloss:0.419505

[477]   validation_0-logloss:0.419501

[478]   validation_0-logloss:0.419442

[479]   validation_0-logloss:0.419418

[480]   validation_0-logloss:0.419408

[481]   validation_0-logloss:0.419357

[482]   validation_0-logloss:0.41933

[483]   validation_0-logloss:0.419351

[484]   validation_0-logloss:0.419356

[485]   validation_0-logloss:0.419338

[486]   validation_0-logloss:0.419309

[487]   validation_0-logloss:0.419321

[488]   validation_0-logloss:0.419317

[489]   validation_0-logloss:0.419303

[490]   validation_0-logloss:0.419304

[491]   validation_0-logloss:0.419306

[492]   validation_0-logloss:0.419297

[493]   validation_0-logloss:0.419299

[494]   validation_0-logloss:0.419314
```

```
[495]   validation_0-logloss:0.419288
[496]   validation_0-logloss:0.419308
[497]   validation_0-logloss:0.419274
[498]   validation_0-logloss:0.419286
[499]   validation_0-logloss:0.419285
[500]   validation_0-logloss:0.419299
[501]   validation_0-logloss:0.41929
[502]   validation_0-logloss:0.419241
[503]   validation_0-logloss:0.419212
[504]   validation_0-logloss:0.419229
[505]   validation_0-logloss:0.419241
[506]   validation_0-logloss:0.419235
[507]   validation_0-logloss:0.419215
[508]   validation_0-logloss:0.419163
[509]   validation_0-logloss:0.419159
[510]   validation_0-logloss:0.419162
[511]   validation_0-logloss:0.419191
[512]   validation_0-logloss:0.419211
[513]   validation_0-logloss:0.419198
[514]   validation_0-logloss:0.419227
[515]   validation_0-logloss:0.419237
[516]   validation_0-logloss:0.419245
[517]   validation_0-logloss:0.419241
[518]   validation_0-logloss:0.419183
```

```
[519]   validation_0-logloss:0.419163
[520]   validation_0-logloss:0.419195
[521]   validation_0-logloss:0.41919
[522]   validation_0-logloss:0.419173
[523]   validation_0-logloss:0.41917
[524]   validation_0-logloss:0.419129
[525]   validation_0-logloss:0.419128
[526]   validation_0-logloss:0.419108
[527]   validation_0-logloss:0.419124
[528]   validation_0-logloss:0.419109
[529]   validation_0-logloss:0.419111
[530]   validation_0-logloss:0.419088
[531]   validation_0-logloss:0.419078
[532]   validation_0-logloss:0.419085
[533]   validation_0-logloss:0.419083
[534]   validation_0-logloss:0.419108
[535]   validation_0-logloss:0.419046
[536]   validation_0-logloss:0.419041
[537]   validation_0-logloss:0.419024
[538]   validation_0-logloss:0.419001
[539]   validation_0-logloss:0.418987
[540]   validation_0-logloss:0.418968
[541]   validation_0-logloss:0.41899
[542]   validation_0-logloss:0.419004
```

```
[543]   validation_0-logloss:0.419016
[544]   validation_0-logloss:0.419003
[545]   validation_0-logloss:0.419052
[546]   validation_0-logloss:0.419021
[547]   validation_0-logloss:0.419021
[548]   validation_0-logloss:0.419018
[549]   validation_0-logloss:0.419028
[550]   validation_0-logloss:0.418993
[551]   validation_0-logloss:0.418967
[552]   validation_0-logloss:0.418968
[553]   validation_0-logloss:0.418982
[554]   validation_0-logloss:0.41899
[555]   validation_0-logloss:0.418998
[556]   validation_0-logloss:0.418971
[557]   validation_0-logloss:0.418961
[558]   validation_0-logloss:0.418975
[559]   validation_0-logloss:0.419001
[560]   validation_0-logloss:0.419008
[561]   validation_0-logloss:0.419009
[562]   validation_0-logloss:0.418995
[563]   validation_0-logloss:0.418952
[564]   validation_0-logloss:0.418908
[565]   validation_0-logloss:0.418911
[566]   validation_0-logloss:0.418944
```

```
[567]   validation_0-logloss:0.418945

[568]   validation_0-logloss:0.418897

[569]   validation_0-logloss:0.418919

[570]   validation_0-logloss:0.4189

[571]   validation_0-logloss:0.418908

[572]   validation_0-logloss:0.418917

[573]   validation_0-logloss:0.41892

[574]   validation_0-logloss:0.418938

[575]   validation_0-logloss:0.418942

[576]   validation_0-logloss:0.418921

[577]   validation_0-logloss:0.418939

[578]   validation_0-logloss:0.418969

[579]   validation_0-logloss:0.41896

[580]   validation_0-logloss:0.418942

[581]   validation_0-logloss:0.418923

[582]   validation_0-logloss:0.418921

[583]   validation_0-logloss:0.418895

[584]   validation_0-logloss:0.418887

[585]   validation_0-logloss:0.418865

[586]   validation_0-logloss:0.418825

[587]   validation_0-logloss:0.418825

[588]   validation_0-logloss:0.418833

[589]   validation_0-logloss:0.418792

[590]   validation_0-logloss:0.418759
```

```
[591]   validation_0-logloss:0.418764
[592]   validation_0-logloss:0.418789
[593]   validation_0-logloss:0.418772
[594]   validation_0-logloss:0.418727
[595]   validation_0-logloss:0.418745
[596]   validation_0-logloss:0.418744
[597]   validation_0-logloss:0.418752
[598]   validation_0-logloss:0.418712
[599]   validation_0-logloss:0.418735
[600]   validation_0-logloss:0.418774
[601]   validation_0-logloss:0.41878
[602]   validation_0-logloss:0.418772
[603]   validation_0-logloss:0.418802
[604]   validation_0-logloss:0.418796
[605]   validation_0-logloss:0.418738
[606]   validation_0-logloss:0.418688
[607]   validation_0-logloss:0.418684
[608]   validation_0-logloss:0.41869
[609]   validation_0-logloss:0.41871
[610]   validation_0-logloss:0.418722
[611]   validation_0-logloss:0.418732
[612]   validation_0-logloss:0.418683
[613]   validation_0-logloss:0.418695
[614]   validation_0-logloss:0.418673
```

```
[615]   validation_0-logloss:0.418652
[616]   validation_0-logloss:0.418641
[617]   validation_0-logloss:0.41868
[618]   validation_0-logloss:0.418644
[619]   validation_0-logloss:0.418656
[620]   validation_0-logloss:0.41865
[621]   validation_0-logloss:0.418691
[622]   validation_0-logloss:0.41866
[623]   validation_0-logloss:0.418653
[624]   validation_0-logloss:0.418665
[625]   validation_0-logloss:0.418658
[626]   validation_0-logloss:0.418656
[627]   validation_0-logloss:0.418642
[628]   validation_0-logloss:0.418657
[629]   validation_0-logloss:0.418644
[630]   validation_0-logloss:0.41862
[631]   validation_0-logloss:0.418611
[632]   validation_0-logloss:0.418608
[633]   validation_0-logloss:0.418574
[634]   validation_0-logloss:0.418543
[635]   validation_0-logloss:0.41855
[636]   validation_0-logloss:0.418567
[637]   validation_0-logloss:0.418599
[638]   validation_0-logloss:0.418581
```

```
[639]   validation_0-logloss:0.418556
[640]   validation_0-logloss:0.418541
[641]   validation_0-logloss:0.418554
[642]   validation_0-logloss:0.418543
[643]   validation_0-logloss:0.418536
[644]   validation_0-logloss:0.418497
[645]   validation_0-logloss:0.418492
[646]   validation_0-logloss:0.418463
[647]   validation_0-logloss:0.418412
[648]   validation_0-logloss:0.418398
[649]   validation_0-logloss:0.418376
[650]   validation_0-logloss:0.41837
[651]   validation_0-logloss:0.418397
[652]   validation_0-logloss:0.418404
[653]   validation_0-logloss:0.418408
[654]   validation_0-logloss:0.41842
[655]   validation_0-logloss:0.418394
[656]   validation_0-logloss:0.418393
[657]   validation_0-logloss:0.418463
[658]   validation_0-logloss:0.418496
[659]   validation_0-logloss:0.418493
[660]   validation_0-logloss:0.418494
[661]   validation_0-logloss:0.418489
[662]   validation_0-logloss:0.418465
```

```
[663]   validation_0-logloss:0.418448
[664]   validation_0-logloss:0.418434
[665]   validation_0-logloss:0.418361
[666]   validation_0-logloss:0.418364
[667]   validation_0-logloss:0.418343
[668]   validation_0-logloss:0.418323
[669]   validation_0-logloss:0.41833
[670]   validation_0-logloss:0.418318
[671]   validation_0-logloss:0.418319
[672]   validation_0-logloss:0.418289
[673]   validation_0-logloss:0.418267
[674]   validation_0-logloss:0.418227
[675]   validation_0-logloss:0.418237
[676]   validation_0-logloss:0.418275
[677]   validation_0-logloss:0.418271
[678]   validation_0-logloss:0.418282
[679]   validation_0-logloss:0.418268
[680]   validation_0-logloss:0.418238
[681]   validation_0-logloss:0.418267
[682]   validation_0-logloss:0.41824
[683]   validation_0-logloss:0.418259
[684]   validation_0-logloss:0.418248
[685]   validation_0-logloss:0.418242
[686]   validation_0-logloss:0.418209
```

```
[687]   validation_0-logloss:0.418184
[688]   validation_0-logloss:0.418163
[689]   validation_0-logloss:0.41818
[690]   validation_0-logloss:0.418174
[691]   validation_0-logloss:0.418174
[692]   validation_0-logloss:0.41821
[693]   validation_0-logloss:0.418203
[694]   validation_0-logloss:0.418245
[695]   validation_0-logloss:0.41823
[696]   validation_0-logloss:0.418236
[697]   validation_0-logloss:0.418247
[698]   validation_0-logloss:0.418206
[699]   validation_0-logloss:0.418153
[700]   validation_0-logloss:0.418153
[701]   validation_0-logloss:0.418153
[702]   validation_0-logloss:0.418159
[703]   validation_0-logloss:0.418184
[704]   validation_0-logloss:0.418191
[705]   validation_0-logloss:0.418164
[706]   validation_0-logloss:0.418153
[707]   validation_0-logloss:0.418169
[708]   validation_0-logloss:0.41819
[709]   validation_0-logloss:0.418201
[710]   validation_0-logloss:0.418153
```

```
[711]   validation_0-logloss:0.418135
[712]   validation_0-logloss:0.418143
[713]   validation_0-logloss:0.418128
[714]   validation_0-logloss:0.41811
[715]   validation_0-logloss:0.418098
[716]   validation_0-logloss:0.418093
[717]   validation_0-logloss:0.418117
[718]   validation_0-logloss:0.418115
[719]   validation_0-logloss:0.418132
[720]   validation_0-logloss:0.418129
[721]   validation_0-logloss:0.418114
[722]   validation_0-logloss:0.41814
[723]   validation_0-logloss:0.418141
[724]   validation_0-logloss:0.418129
[725]   validation_0-logloss:0.418105
[726]   validation_0-logloss:0.418123
[727]   validation_0-logloss:0.418131
[728]   validation_0-logloss:0.418095
[729]   validation_0-logloss:0.418092
[730]   validation_0-logloss:0.418039
[731]   validation_0-logloss:0.418022
[732]   validation_0-logloss:0.418036
[733]   validation_0-logloss:0.418011
[734]   validation_0-logloss:0.417992
```

```
[735]   validation_0-logloss:0.418032

[736]   validation_0-logloss:0.418029

[737]   validation_0-logloss:0.418095

[738]   validation_0-logloss:0.418049

[739]   validation_0-logloss:0.41802

[740]   validation_0-logloss:0.418029

[741]   validation_0-logloss:0.41804

[742]   validation_0-logloss:0.418003

[743]   validation_0-logloss:0.418012

[744]   validation_0-logloss:0.418017

[745]   validation_0-logloss:0.41797

[746]   validation_0-logloss:0.417959

[747]   validation_0-logloss:0.417967

[748]   validation_0-logloss:0.417959

[749]   validation_0-logloss:0.417966

0.39841220352387396
[0]     validation_0-logloss:0.657214

Will train until validation_0-logloss hasn't improved in 50 rounds.

[1]     validation_0-logloss:0.627323

[2]     validation_0-logloss:0.602144

[3]     validation_0-logloss:0.586846

[4]     validation_0-logloss:0.56694

[5]     validation_0-logloss:0.555473

[6]     validation_0-logloss:0.541272

[7]     validation_0-logloss:0.526587
```

```
[8]     validation_0-logloss:0.519702

[9]     validation_0-logloss:0.508284

[10]    validation_0-logloss:0.500049

[11]    validation_0-logloss:0.490668

[12]    validation_0-logloss:0.483678

[13]    validation_0-logloss:0.477458

[14]    validation_0-logloss:0.471333

[15]    validation_0-logloss:0.466301

[16]    validation_0-logloss:0.461591

[17]    validation_0-logloss:0.45923

[18]    validation_0-logloss:0.45704

[19]    validation_0-logloss:0.45391

[20]    validation_0-logloss:0.450428

[21]    validation_0-logloss:0.447648

[22]    validation_0-logloss:0.445263

[23]    validation_0-logloss:0.443897

[24]    validation_0-logloss:0.441778

[25]    validation_0-logloss:0.43991

[26]    validation_0-logloss:0.438289

[27]    validation_0-logloss:0.436829

[28]    validation_0-logloss:0.435375

[29]    validation_0-logloss:0.434123

[30]    validation_0-logloss:0.433062

[31]    validation_0-logloss:0.432026
```

```
[32]    validation_0-logloss:0.431234
[33]    validation_0-logloss:0.430599
[34]    validation_0-logloss:0.430033
[35]    validation_0-logloss:0.429381
[36]    validation_0-logloss:0.429053
[37]    validation_0-logloss:0.4286
[38]    validation_0-logloss:0.428049
[39]    validation_0-logloss:0.427759
[40]    validation_0-logloss:0.427263
[41]    validation_0-logloss:0.426821
[42]    validation_0-logloss:0.426585
[43]    validation_0-logloss:0.426372
[44]    validation_0-logloss:0.426152
[45]    validation_0-logloss:0.425859
[46]    validation_0-logloss:0.425687
[47]    validation_0-logloss:0.425414
[48]    validation_0-logloss:0.425189
[49]    validation_0-logloss:0.425072
[50]    validation_0-logloss:0.424854
[51]    validation_0-logloss:0.424685
[52]    validation_0-logloss:0.424536
[53]    validation_0-logloss:0.42436
[54]    validation_0-logloss:0.42423
[55]    validation_0-logloss:0.424097
```

```
[56]    validation_0-logloss:0.423985
[57]    validation_0-logloss:0.423895
[58]    validation_0-logloss:0.423849
[59]    validation_0-logloss:0.423786
[60]    validation_0-logloss:0.423671
[61]    validation_0-logloss:0.423612
[62]    validation_0-logloss:0.423551
[63]    validation_0-logloss:0.423523
[64]    validation_0-logloss:0.423505
[65]    validation_0-logloss:0.423485
[66]    validation_0-logloss:0.423451
[67]    validation_0-logloss:0.423451
[68]    validation_0-logloss:0.423389
[69]    validation_0-logloss:0.423389
[70]    validation_0-logloss:0.423346
[71]    validation_0-logloss:0.423294
[72]    validation_0-logloss:0.423263
[73]    validation_0-logloss:0.423217
[74]    validation_0-logloss:0.423252
[75]    validation_0-logloss:0.423172
[76]    validation_0-logloss:0.423198
[77]    validation_0-logloss:0.423161
[78]    validation_0-logloss:0.423128
[79]    validation_0-logloss:0.423125
```

```
[80]    validation_0-logloss:0.42307
[81]    validation_0-logloss:0.423029
[82]    validation_0-logloss:0.423015
[83]    validation_0-logloss:0.422984
[84]    validation_0-logloss:0.422933
[85]    validation_0-logloss:0.422902
[86]    validation_0-logloss:0.422898
[87]    validation_0-logloss:0.42287
[88]    validation_0-logloss:0.422875
[89]    validation_0-logloss:0.42286
[90]    validation_0-logloss:0.422878
[91]    validation_0-logloss:0.422887
[92]    validation_0-logloss:0.422904
[93]    validation_0-logloss:0.422884
[94]    validation_0-logloss:0.42292
[95]    validation_0-logloss:0.422904
[96]    validation_0-logloss:0.422885
[97]    validation_0-logloss:0.422889
[98]    validation_0-logloss:0.422913
[99]    validation_0-logloss:0.422856
[100]   validation_0-logloss:0.422839
[101]   validation_0-logloss:0.422837
[102]   validation_0-logloss:0.422821
[103]   validation_0-logloss:0.422814
```

```
[104]   validation_0-logloss:0.422805
[105]   validation_0-logloss:0.422751
[106]   validation_0-logloss:0.422739
[107]   validation_0-logloss:0.42269
[108]   validation_0-logloss:0.422654
[109]   validation_0-logloss:0.422655
[110]   validation_0-logloss:0.422653
[111]   validation_0-logloss:0.422637
[112]   validation_0-logloss:0.422634
[113]   validation_0-logloss:0.422624
[114]   validation_0-logloss:0.422635
[115]   validation_0-logloss:0.422584
[116]   validation_0-logloss:0.422584
[117]   validation_0-logloss:0.422555
[118]   validation_0-logloss:0.422553
[119]   validation_0-logloss:0.422561
[120]   validation_0-logloss:0.422565
[121]   validation_0-logloss:0.422524
[122]   validation_0-logloss:0.422506
[123]   validation_0-logloss:0.422452
[124]   validation_0-logloss:0.422492
[125]   validation_0-logloss:0.42248
[126]   validation_0-logloss:0.422467
[127]   validation_0-logloss:0.422444
```

```
[128]   validation_0-logloss:0.422426
[129]   validation_0-logloss:0.422425
[130]   validation_0-logloss:0.422439
[131]   validation_0-logloss:0.422456
[132]   validation_0-logloss:0.422454
[133]   validation_0-logloss:0.422439
[134]   validation_0-logloss:0.422432
[135]   validation_0-logloss:0.422391
[136]   validation_0-logloss:0.422399
[137]   validation_0-logloss:0.422389
[138]   validation_0-logloss:0.422381
[139]   validation_0-logloss:0.422361
[140]   validation_0-logloss:0.422352
[141]   validation_0-logloss:0.422288
[142]   validation_0-logloss:0.422258
[143]   validation_0-logloss:0.422247
[144]   validation_0-logloss:0.422256
[145]   validation_0-logloss:0.422259
[146]   validation_0-logloss:0.422257
[147]   validation_0-logloss:0.422266
[148]   validation_0-logloss:0.422219
[149]   validation_0-logloss:0.422216
[150]   validation_0-logloss:0.422214
[151]   validation_0-logloss:0.422209
```

```
[152]   validation_0-logloss:0.422173
[153]   validation_0-logloss:0.422161
[154]   validation_0-logloss:0.42217
[155]   validation_0-logloss:0.422137
[156]   validation_0-logloss:0.4221
[157]   validation_0-logloss:0.422131
[158]   validation_0-logloss:0.422144
[159]   validation_0-logloss:0.422102
[160]   validation_0-logloss:0.422086
[161]   validation_0-logloss:0.422089
[162]   validation_0-logloss:0.422074
[163]   validation_0-logloss:0.422087
[164]   validation_0-logloss:0.422041
[165]   validation_0-logloss:0.422035
[166]   validation_0-logloss:0.422042
[167]   validation_0-logloss:0.422061
[168]   validation_0-logloss:0.422039
[169]   validation_0-logloss:0.422019
[170]   validation_0-logloss:0.421998
[171]   validation_0-logloss:0.421964
[172]   validation_0-logloss:0.421932
[173]   validation_0-logloss:0.421924
[174]   validation_0-logloss:0.421929
[175]   validation_0-logloss:0.421912
```

```
[176]   validation_0-logloss:0.421926

[177]   validation_0-logloss:0.42192

[178]   validation_0-logloss:0.421935

[179]   validation_0-logloss:0.421899

[180]   validation_0-logloss:0.421899

[181]   validation_0-logloss:0.421895

[182]   validation_0-logloss:0.42187

[183]   validation_0-logloss:0.421875

[184]   validation_0-logloss:0.42187

[185]   validation_0-logloss:0.421863

[186]   validation_0-logloss:0.42184

[187]   validation_0-logloss:0.421818

[188]   validation_0-logloss:0.421814

[189]   validation_0-logloss:0.421792

[190]   validation_0-logloss:0.421782

[191]   validation_0-logloss:0.421817

[192]   validation_0-logloss:0.42178

[193]   validation_0-logloss:0.421781

[194]   validation_0-logloss:0.421793

[195]   validation_0-logloss:0.42179

[196]   validation_0-logloss:0.421772

[197]   validation_0-logloss:0.421775

[198]   validation_0-logloss:0.421728

[199]   validation_0-logloss:0.421722
```

```
[200]   validation_0-logloss:0.421732
[201]   validation_0-logloss:0.421724
[202]   validation_0-logloss:0.42175
[203]   validation_0-logloss:0.421688
[204]   validation_0-logloss:0.421661
[205]   validation_0-logloss:0.421687
[206]   validation_0-logloss:0.421656
[207]   validation_0-logloss:0.421661
[208]   validation_0-logloss:0.421674
[209]   validation_0-logloss:0.421672
[210]   validation_0-logloss:0.421665
[211]   validation_0-logloss:0.421664
[212]   validation_0-logloss:0.421654
[213]   validation_0-logloss:0.421659
[214]   validation_0-logloss:0.421629
[215]   validation_0-logloss:0.421612
[216]   validation_0-logloss:0.421607
[217]   validation_0-logloss:0.421597
[218]   validation_0-logloss:0.421653
[219]   validation_0-logloss:0.421647
[220]   validation_0-logloss:0.421654
[221]   validation_0-logloss:0.421666
[222]   validation_0-logloss:0.421647
[223]   validation_0-logloss:0.421633
```

```
[224]   validation_0-logloss:0.421582
[225]   validation_0-logloss:0.421614
[226]   validation_0-logloss:0.421635
[227]   validation_0-logloss:0.421622
[228]   validation_0-logloss:0.421599
[229]   validation_0-logloss:0.421601
[230]   validation_0-logloss:0.421559
[231]   validation_0-logloss:0.421542
[232]   validation_0-logloss:0.421548
[233]   validation_0-logloss:0.421557
[234]   validation_0-logloss:0.421521
[235]   validation_0-logloss:0.421501
[236]   validation_0-logloss:0.421506
[237]   validation_0-logloss:0.42146
[238]   validation_0-logloss:0.421464
[239]   validation_0-logloss:0.421459
[240]   validation_0-logloss:0.421457
[241]   validation_0-logloss:0.421457
[242]   validation_0-logloss:0.421423
[243]   validation_0-logloss:0.42145
[244]   validation_0-logloss:0.421448
[245]   validation_0-logloss:0.42144
[246]   validation_0-logloss:0.421421
[247]   validation_0-logloss:0.421402
```

```
[248]   validation_0-logloss:0.421375
[249]   validation_0-logloss:0.421365
[250]   validation_0-logloss:0.421389
[251]   validation_0-logloss:0.421409
[252]   validation_0-logloss:0.421418
[253]   validation_0-logloss:0.421417
[254]   validation_0-logloss:0.421396
[255]   validation_0-logloss:0.421399
[256]   validation_0-logloss:0.421377
[257]   validation_0-logloss:0.421367
[258]   validation_0-logloss:0.42132
[259]   validation_0-logloss:0.421306
[260]   validation_0-logloss:0.421278
[261]   validation_0-logloss:0.421259
[262]   validation_0-logloss:0.421223
[263]   validation_0-logloss:0.421168
[264]   validation_0-logloss:0.421211
[265]   validation_0-logloss:0.421203
[266]   validation_0-logloss:0.421181
[267]   validation_0-logloss:0.421186
[268]   validation_0-logloss:0.421207
[269]   validation_0-logloss:0.421204
[270]   validation_0-logloss:0.421168
[271]   validation_0-logloss:0.4212
```

```
[272]   validation_0-logloss:0.421213
[273]   validation_0-logloss:0.421227
[274]   validation_0-logloss:0.421212
[275]   validation_0-logloss:0.42124
[276]   validation_0-logloss:0.421257
[277]   validation_0-logloss:0.421248
[278]   validation_0-logloss:0.421243
[279]   validation_0-logloss:0.421206
[280]   validation_0-logloss:0.421231
[281]   validation_0-logloss:0.421191
[282]   validation_0-logloss:0.421177
[283]   validation_0-logloss:0.421156
[284]   validation_0-logloss:0.421157
[285]   validation_0-logloss:0.421187
[286]   validation_0-logloss:0.421119
[287]   validation_0-logloss:0.421137
[288]   validation_0-logloss:0.421097
[289]   validation_0-logloss:0.421102
[290]   validation_0-logloss:0.421121
[291]   validation_0-logloss:0.421104
[292]   validation_0-logloss:0.421116
[293]   validation_0-logloss:0.421073
[294]   validation_0-logloss:0.421084
[295]   validation_0-logloss:0.421078
```

```
[296]   validation_0-logloss:0.421086
[297]   validation_0-logloss:0.421111
[298]   validation_0-logloss:0.421097
[299]   validation_0-logloss:0.421082
[300]   validation_0-logloss:0.42106
[301]   validation_0-logloss:0.421053
[302]   validation_0-logloss:0.421045
[303]   validation_0-logloss:0.421053
[304]   validation_0-logloss:0.421044
[305]   validation_0-logloss:0.421057
[306]   validation_0-logloss:0.421053
[307]   validation_0-logloss:0.421067
[308]   validation_0-logloss:0.421032
[309]   validation_0-logloss:0.42104
[310]   validation_0-logloss:0.420987
[311]   validation_0-logloss:0.420984
[312]   validation_0-logloss:0.421022
[313]   validation_0-logloss:0.420999
[314]   validation_0-logloss:0.420981
[315]   validation_0-logloss:0.420989
[316]   validation_0-logloss:0.42096
[317]   validation_0-logloss:0.420916
[318]   validation_0-logloss:0.420911
[319]   validation_0-logloss:0.420938
```

```
[320]   validation_0-logloss:0.420938
[321]   validation_0-logloss:0.420958
[322]   validation_0-logloss:0.420909
[323]   validation_0-logloss:0.420894
[324]   validation_0-logloss:0.420888
[325]   validation_0-logloss:0.420895
[326]   validation_0-logloss:0.420861
[327]   validation_0-logloss:0.420873
[328]   validation_0-logloss:0.420857
[329]   validation_0-logloss:0.420846
[330]   validation_0-logloss:0.420802
[331]   validation_0-logloss:0.420768
[332]   validation_0-logloss:0.420772
[333]   validation_0-logloss:0.420783
[334]   validation_0-logloss:0.420804
[335]   validation_0-logloss:0.420798
[336]   validation_0-logloss:0.420816
[337]   validation_0-logloss:0.420796
[338]   validation_0-logloss:0.420792
[339]   validation_0-logloss:0.420788
[340]   validation_0-logloss:0.420776
[341]   validation_0-logloss:0.420765
[342]   validation_0-logloss:0.420752
[343]   validation_0-logloss:0.42072
```

```
[344]   validation_0-logloss:0.420712
[345]   validation_0-logloss:0.420727
[346]   validation_0-logloss:0.420737
[347]   validation_0-logloss:0.420743
[348]   validation_0-logloss:0.420739
[349]   validation_0-logloss:0.420727
[350]   validation_0-logloss:0.42073
[351]   validation_0-logloss:0.420706
[352]   validation_0-logloss:0.420754
[353]   validation_0-logloss:0.420779
[354]   validation_0-logloss:0.420758
[355]   validation_0-logloss:0.420754
[356]   validation_0-logloss:0.420744
[357]   validation_0-logloss:0.420724
[358]   validation_0-logloss:0.420713
[359]   validation_0-logloss:0.420699
[360]   validation_0-logloss:0.420676
[361]   validation_0-logloss:0.420642
[362]   validation_0-logloss:0.420626
[363]   validation_0-logloss:0.420602
[364]   validation_0-logloss:0.420576
[365]   validation_0-logloss:0.420568
[366]   validation_0-logloss:0.420562
[367]   validation_0-logloss:0.420546
```

```
[368]   validation_0-logloss:0.420537
[369]   validation_0-logloss:0.42057
[370]   validation_0-logloss:0.420582
[371]   validation_0-logloss:0.420542
[372]   validation_0-logloss:0.420523
[373]   validation_0-logloss:0.420524
[374]   validation_0-logloss:0.42047
[375]   validation_0-logloss:0.420422
[376]   validation_0-logloss:0.420447
[377]   validation_0-logloss:0.420434
[378]   validation_0-logloss:0.420442
[379]   validation_0-logloss:0.420485
[380]   validation_0-logloss:0.420469
[381]   validation_0-logloss:0.420448
[382]   validation_0-logloss:0.420435
[383]   validation_0-logloss:0.420409
[384]   validation_0-logloss:0.420374
[385]   validation_0-logloss:0.420372
[386]   validation_0-logloss:0.42037
[387]   validation_0-logloss:0.420346
[388]   validation_0-logloss:0.420307
[389]   validation_0-logloss:0.420315
[390]   validation_0-logloss:0.420299
[391]   validation_0-logloss:0.420312
```

```
[392]   validation_0-logloss:0.420303
[393]   validation_0-logloss:0.420307
[394]   validation_0-logloss:0.420287
[395]   validation_0-logloss:0.42029
[396]   validation_0-logloss:0.420291
[397]   validation_0-logloss:0.420305
[398]   validation_0-logloss:0.420349
[399]   validation_0-logloss:0.420395
[400]   validation_0-logloss:0.420396
[401]   validation_0-logloss:0.42039
[402]   validation_0-logloss:0.42043
[403]   validation_0-logloss:0.420411
[404]   validation_0-logloss:0.420363
[405]   validation_0-logloss:0.420324
[406]   validation_0-logloss:0.42032
[407]   validation_0-logloss:0.420346
[408]   validation_0-logloss:0.420306
[409]   validation_0-logloss:0.42032
[410]   validation_0-logloss:0.42028
[411]   validation_0-logloss:0.420311
[412]   validation_0-logloss:0.420331
[413]   validation_0-logloss:0.420347
[414]   validation_0-logloss:0.420326
[415]   validation_0-logloss:0.42032
```

```
[416]   validation_0-logloss:0.420331
[417]   validation_0-logloss:0.420363
[418]   validation_0-logloss:0.420319
[419]   validation_0-logloss:0.420294
[420]   validation_0-logloss:0.420281
[421]   validation_0-logloss:0.420284
[422]   validation_0-logloss:0.420265
[423]   validation_0-logloss:0.420265
[424]   validation_0-logloss:0.420253
[425]   validation_0-logloss:0.420227
[426]   validation_0-logloss:0.420253
[427]   validation_0-logloss:0.420242
[428]   validation_0-logloss:0.420242
[429]   validation_0-logloss:0.420234
[430]   validation_0-logloss:0.420207
[431]   validation_0-logloss:0.420166
[432]   validation_0-logloss:0.420181
[433]   validation_0-logloss:0.420157
[434]   validation_0-logloss:0.420153
[435]   validation_0-logloss:0.420181
[436]   validation_0-logloss:0.420195
[437]   validation_0-logloss:0.42016
[438]   validation_0-logloss:0.420144
[439]   validation_0-logloss:0.420112
```

```
[440]   validation_0-logloss:0.42009
[441]   validation_0-logloss:0.420078
[442]   validation_0-logloss:0.42007
[443]   validation_0-logloss:0.420058
[444]   validation_0-logloss:0.420046
[445]   validation_0-logloss:0.420055
[446]   validation_0-logloss:0.420037
[447]   validation_0-logloss:0.420063
[448]   validation_0-logloss:0.420072
[449]   validation_0-logloss:0.420072
[450]   validation_0-logloss:0.420053
[451]   validation_0-logloss:0.420045
[452]   validation_0-logloss:0.420012
[453]   validation_0-logloss:0.419979
[454]   validation_0-logloss:0.420009
[455]   validation_0-logloss:0.419997
[456]   validation_0-logloss:0.41998
[457]   validation_0-logloss:0.419997
[458]   validation_0-logloss:0.419975
[459]   validation_0-logloss:0.419941
[460]   validation_0-logloss:0.419973
[461]   validation_0-logloss:0.419986
[462]   validation_0-logloss:0.420004
[463]   validation_0-logloss:0.419974
```

```
[464]   validation_0-logloss:0.41997
[465]   validation_0-logloss:0.419951
[466]   validation_0-logloss:0.419937
[467]   validation_0-logloss:0.419967
[468]   validation_0-logloss:0.419952
[469]   validation_0-logloss:0.419917
[470]   validation_0-logloss:0.419877
[471]   validation_0-logloss:0.419906
[472]   validation_0-logloss:0.419903
[473]   validation_0-logloss:0.419933
[474]   validation_0-logloss:0.419894
[475]   validation_0-logloss:0.419855
[476]   validation_0-logloss:0.41987
[477]   validation_0-logloss:0.419858
[478]   validation_0-logloss:0.419813
[479]   validation_0-logloss:0.41982
[480]   validation_0-logloss:0.419811
[481]   validation_0-logloss:0.419786
[482]   validation_0-logloss:0.419756
[483]   validation_0-logloss:0.419776
[484]   validation_0-logloss:0.419791
[485]   validation_0-logloss:0.419769
[486]   validation_0-logloss:0.41978
[487]   validation_0-logloss:0.419792
```

```
[488]   validation_0-logloss:0.4198
[489]   validation_0-logloss:0.419775
[490]   validation_0-logloss:0.419772
[491]   validation_0-logloss:0.419714
[492]   validation_0-logloss:0.419698
[493]   validation_0-logloss:0.41969
[494]   validation_0-logloss:0.419674
[495]   validation_0-logloss:0.419648
[496]   validation_0-logloss:0.419676
[497]   validation_0-logloss:0.419659
[498]   validation_0-logloss:0.419672
[499]   validation_0-logloss:0.419677
[500]   validation_0-logloss:0.419677
[501]   validation_0-logloss:0.41968
[502]   validation_0-logloss:0.419673
[503]   validation_0-logloss:0.419653
[504]   validation_0-logloss:0.419634
[505]   validation_0-logloss:0.419604
[506]   validation_0-logloss:0.419555
[507]   validation_0-logloss:0.419527
[508]   validation_0-logloss:0.419512
[509]   validation_0-logloss:0.419552
[510]   validation_0-logloss:0.419574
[511]   validation_0-logloss:0.419565
```

```
[512]   validation_0-logloss:0.419608
[513]   validation_0-logloss:0.419609
[514]   validation_0-logloss:0.419592
[515]   validation_0-logloss:0.419594
[516]   validation_0-logloss:0.419632
[517]   validation_0-logloss:0.419613
[518]   validation_0-logloss:0.419595
[519]   validation_0-logloss:0.419583
[520]   validation_0-logloss:0.419644
[521]   validation_0-logloss:0.419612
[522]   validation_0-logloss:0.419588
[523]   validation_0-logloss:0.41958
[524]   validation_0-logloss:0.419516
[525]   validation_0-logloss:0.419518
[526]   validation_0-logloss:0.419503
[527]   validation_0-logloss:0.419533
[528]   validation_0-logloss:0.419519
[529]   validation_0-logloss:0.419525
[530]   validation_0-logloss:0.419514
[531]   validation_0-logloss:0.419462
[532]   validation_0-logloss:0.419451
[533]   validation_0-logloss:0.41948
[534]   validation_0-logloss:0.419459
[535]   validation_0-logloss:0.419416
```

```
[536]   validation_0-logloss:0.419409
[537]   validation_0-logloss:0.419412
[538]   validation_0-logloss:0.419377
[539]   validation_0-logloss:0.419362
[540]   validation_0-logloss:0.419335
[541]   validation_0-logloss:0.419352
[542]   validation_0-logloss:0.419348
[543]   validation_0-logloss:0.41933
[544]   validation_0-logloss:0.419335
[545]   validation_0-logloss:0.419396
[546]   validation_0-logloss:0.419394
[547]   validation_0-logloss:0.419388
[548]   validation_0-logloss:0.419366
[549]   validation_0-logloss:0.419365
[550]   validation_0-logloss:0.419354
[551]   validation_0-logloss:0.419347
[552]   validation_0-logloss:0.419332
[553]   validation_0-logloss:0.419318
[554]   validation_0-logloss:0.419313
[555]   validation_0-logloss:0.419346
[556]   validation_0-logloss:0.419333
[557]   validation_0-logloss:0.419306
[558]   validation_0-logloss:0.419311
[559]   validation_0-logloss:0.419329
```

```
[560]   validation_0-logloss:0.419316
[561]   validation_0-logloss:0.419292
[562]   validation_0-logloss:0.419315
[563]   validation_0-logloss:0.419264
[564]   validation_0-logloss:0.41926
[565]   validation_0-logloss:0.419266
[566]   validation_0-logloss:0.419267
[567]   validation_0-logloss:0.419269
[568]   validation_0-logloss:0.419244
[569]   validation_0-logloss:0.419275
[570]   validation_0-logloss:0.419249
[571]   validation_0-logloss:0.419276
[572]   validation_0-logloss:0.41928
[573]   validation_0-logloss:0.419274
[574]   validation_0-logloss:0.419257
[575]   validation_0-logloss:0.419261
[576]   validation_0-logloss:0.419252
[577]   validation_0-logloss:0.419229
[578]   validation_0-logloss:0.419253
[579]   validation_0-logloss:0.41924
[580]   validation_0-logloss:0.419226
[581]   validation_0-logloss:0.419178
[582]   validation_0-logloss:0.419196
[583]   validation_0-logloss:0.41919
```

```
[584]   validation_0-logloss:0.419199
[585]   validation_0-logloss:0.419208
[586]   validation_0-logloss:0.419172
[587]   validation_0-logloss:0.419182
[588]   validation_0-logloss:0.419182
[589]   validation_0-logloss:0.419152
[590]   validation_0-logloss:0.419148
[591]   validation_0-logloss:0.419163
[592]   validation_0-logloss:0.419192
[593]   validation_0-logloss:0.41917
[594]   validation_0-logloss:0.419145
[595]   validation_0-logloss:0.419166
[596]   validation_0-logloss:0.419191
[597]   validation_0-logloss:0.419188
[598]   validation_0-logloss:0.419121
[599]   validation_0-logloss:0.419132
[600]   validation_0-logloss:0.419147
[601]   validation_0-logloss:0.419138
[602]   validation_0-logloss:0.419121
[603]   validation_0-logloss:0.419124
[604]   validation_0-logloss:0.419124
[605]   validation_0-logloss:0.419044
[606]   validation_0-logloss:0.419027
[607]   validation_0-logloss:0.419026
```

```
[608]   validation_0-logloss:0.41905
[609]   validation_0-logloss:0.419049
[610]   validation_0-logloss:0.41905
[611]   validation_0-logloss:0.419041
[612]   validation_0-logloss:0.419065
[613]   validation_0-logloss:0.419081
[614]   validation_0-logloss:0.419088
[615]   validation_0-logloss:0.419073
[616]   validation_0-logloss:0.41904
[617]   validation_0-logloss:0.419068
[618]   validation_0-logloss:0.41906
[619]   validation_0-logloss:0.419032
[620]   validation_0-logloss:0.419031
[621]   validation_0-logloss:0.419075
[622]   validation_0-logloss:0.419047
[623]   validation_0-logloss:0.419036
[624]   validation_0-logloss:0.419109
[625]   validation_0-logloss:0.419118
[626]   validation_0-logloss:0.419109
[627]   validation_0-logloss:0.419085
[628]   validation_0-logloss:0.419075
[629]   validation_0-logloss:0.419025
[630]   validation_0-logloss:0.419027
[631]   validation_0-logloss:0.418969
```

```
[632]   validation_0-logloss:0.418943
[633]   validation_0-logloss:0.418947
[634]   validation_0-logloss:0.418897
[635]   validation_0-logloss:0.41892
[636]   validation_0-logloss:0.418919
[637]   validation_0-logloss:0.418913
[638]   validation_0-logloss:0.418862
[639]   validation_0-logloss:0.418832
[640]   validation_0-logloss:0.41881
[641]   validation_0-logloss:0.418841
[642]   validation_0-logloss:0.418849
[643]   validation_0-logloss:0.418891
[644]   validation_0-logloss:0.418919
[645]   validation_0-logloss:0.418907
[646]   validation_0-logloss:0.418889
[647]   validation_0-logloss:0.418873
[648]   validation_0-logloss:0.418879
[649]   validation_0-logloss:0.418876
[650]   validation_0-logloss:0.418882
[651]   validation_0-logloss:0.418891
[652]   validation_0-logloss:0.418874
[653]   validation_0-logloss:0.418844
[654]   validation_0-logloss:0.418838
[655]   validation_0-logloss:0.418818
```

```
[656]   validation_0-logloss:0.418875
[657]   validation_0-logloss:0.418878
[658]   validation_0-logloss:0.418874
[659]   validation_0-logloss:0.418912
[660]   validation_0-logloss:0.418924
[661]   validation_0-logloss:0.418933
[662]   validation_0-logloss:0.418924
[663]   validation_0-logloss:0.418912
[664]   validation_0-logloss:0.41891
[665]   validation_0-logloss:0.418878
[666]   validation_0-logloss:0.41889
[667]   validation_0-logloss:0.418898
[668]   validation_0-logloss:0.418868
[669]   validation_0-logloss:0.418879
[670]   validation_0-logloss:0.418861
[671]   validation_0-logloss:0.418886
[672]   validation_0-logloss:0.418881
[673]   validation_0-logloss:0.418825
[674]   validation_0-logloss:0.418793
[675]   validation_0-logloss:0.418787
[676]   validation_0-logloss:0.418775
[677]   validation_0-logloss:0.418758
[678]   validation_0-logloss:0.418776
[679]   validation_0-logloss:0.418747
```

```
[680]   validation_0-logloss:0.418737
[681]   validation_0-logloss:0.418744
[682]   validation_0-logloss:0.41871
[683]   validation_0-logloss:0.418723
[684]   validation_0-logloss:0.418702
[685]   validation_0-logloss:0.418688
[686]   validation_0-logloss:0.418697
[687]   validation_0-logloss:0.41866
[688]   validation_0-logloss:0.418651
[689]   validation_0-logloss:0.418645
[690]   validation_0-logloss:0.41863
[691]   validation_0-logloss:0.418625
[692]   validation_0-logloss:0.418629
[693]   validation_0-logloss:0.418616
[694]   validation_0-logloss:0.418679
[695]   validation_0-logloss:0.418653
[696]   validation_0-logloss:0.418658
[697]   validation_0-logloss:0.418654
[698]   validation_0-logloss:0.41864
[699]   validation_0-logloss:0.418577
[700]   validation_0-logloss:0.418574
[701]   validation_0-logloss:0.418534
[702]   validation_0-logloss:0.418554
[703]   validation_0-logloss:0.418552
```

```
[704]   validation_0-logloss:0.418569
[705]   validation_0-logloss:0.418546
[706]   validation_0-logloss:0.4185
[707]   validation_0-logloss:0.418507
[708]   validation_0-logloss:0.418496
[709]   validation_0-logloss:0.418524
[710]   validation_0-logloss:0.418483
[711]   validation_0-logloss:0.418523
[712]   validation_0-logloss:0.418525
[713]   validation_0-logloss:0.418481
[714]   validation_0-logloss:0.418478
[715]   validation_0-logloss:0.418469
[716]   validation_0-logloss:0.418453
[717]   validation_0-logloss:0.418424
[718]   validation_0-logloss:0.418424
[719]   validation_0-logloss:0.418403
[720]   validation_0-logloss:0.418388
[721]   validation_0-logloss:0.418394
[722]   validation_0-logloss:0.418422
[723]   validation_0-logloss:0.418416
[724]   validation_0-logloss:0.418427
[725]   validation_0-logloss:0.418424
[726]   validation_0-logloss:0.418434
[727]   validation_0-logloss:0.418448
```

```
[728]   validation_0-logloss:0.418423

[729]   validation_0-logloss:0.418422

[730]   validation_0-logloss:0.4184

[731]   validation_0-logloss:0.418403

[732]   validation_0-logloss:0.418425

[733]   validation_0-logloss:0.418415

[734]   validation_0-logloss:0.418369

[735]   validation_0-logloss:0.418392

[736]   validation_0-logloss:0.418388

[737]   validation_0-logloss:0.418432

[738]   validation_0-logloss:0.418426

[739]   validation_0-logloss:0.418384

[740]   validation_0-logloss:0.418411

[741]   validation_0-logloss:0.418412

[742]   validation_0-logloss:0.418403

[743]   validation_0-logloss:0.418434

[744]   validation_0-logloss:0.418481

[745]   validation_0-logloss:0.418476

[746]   validation_0-logloss:0.418446

[747]   validation_0-logloss:0.41846

[748]   validation_0-logloss:0.418426

[749]   validation_0-logloss:0.418412

0.39909128321318854
100%|      | 100/100 [13:41:41<00:00, 493.01s/it, best loss:
-0.4094476497261533]
```

8.0.2 Best pramameter after different trails:

-best_params ={'colsample_bytree': 0.7736610394413841, 'eta': 0.1, 'gamma': 1, 'max_depth': 5, 'scale_pos_weight':9,'n_estimators': 1800, 'subsample': 0.598690048807158, 'tree_method': 'gpu_hist'}

8.1 Training Xgboost with best hyperparams

[58]:
```
best_params ={'colsample_bytree': 0.7736610394413841, 'eta': 0.1, 'gamma': 1,␣
↪'max_depth': 5, 'scale_pos_weight':9,'n_estimators': 1800, 'subsample': 0.
↪598690048807158, 'tree_method': 'gpu_hist'}
```

[59]:
```
#Train xgboost
xgboost=xgb.XGBClassifier(**best_params)
watchlist =[(X_test , y_test)]
```

[60]:
```
xgboost.fit(X_train , y_train ,eval_set=watchlist, eval_metric␣
↪='logloss',early_stopping_rounds=50)
```

```
[0]     validation_0-logloss:0.669561
Will train until validation_0-logloss hasn't improved in 50 rounds.
[1]     validation_0-logloss:0.650047
[2]     validation_0-logloss:0.631839
[3]     validation_0-logloss:0.616543
[4]     validation_0-logloss:0.605007
[5]     validation_0-logloss:0.600454
[6]     validation_0-logloss:0.590555
[7]     validation_0-logloss:0.581559
[8]     validation_0-logloss:0.575187
[9]     validation_0-logloss:0.568418
[10]    validation_0-logloss:0.56374
[11]    validation_0-logloss:0.558403
[12]    validation_0-logloss:0.554813
[13]    validation_0-logloss:0.551007
[14]    validation_0-logloss:0.548128
[15]    validation_0-logloss:0.545182
[16]    validation_0-logloss:0.543092
[17]    validation_0-logloss:0.541911
[18]    validation_0-logloss:0.540089
[19]    validation_0-logloss:0.538176
[20]    validation_0-logloss:0.53661
[21]    validation_0-logloss:0.535262
[22]    validation_0-logloss:0.534441
[23]    validation_0-logloss:0.533528
[24]    validation_0-logloss:0.532684
[25]    validation_0-logloss:0.532054
[26]    validation_0-logloss:0.531385
[27]    validation_0-logloss:0.530683
```

```
[28]    validation_0-logloss:0.530135
[29]    validation_0-logloss:0.529685
[30]    validation_0-logloss:0.529383
[31]    validation_0-logloss:0.52901
[32]    validation_0-logloss:0.52863
[33]    validation_0-logloss:0.528322
[34]    validation_0-logloss:0.528027
[35]    validation_0-logloss:0.527884
[36]    validation_0-logloss:0.527673
[37]    validation_0-logloss:0.527483
[38]    validation_0-logloss:0.527231
[39]    validation_0-logloss:0.527037
[40]    validation_0-logloss:0.526917
[41]    validation_0-logloss:0.526778
[42]    validation_0-logloss:0.526544
[43]    validation_0-logloss:0.526466
[44]    validation_0-logloss:0.52638
[45]    validation_0-logloss:0.526253
[46]    validation_0-logloss:0.526173
[47]    validation_0-logloss:0.526073
[48]    validation_0-logloss:0.525885
[49]    validation_0-logloss:0.525881
[50]    validation_0-logloss:0.525731
[51]    validation_0-logloss:0.525782
[52]    validation_0-logloss:0.525726
[53]    validation_0-logloss:0.525699
[54]    validation_0-logloss:0.525571
[55]    validation_0-logloss:0.525469
[56]    validation_0-logloss:0.52547
[57]    validation_0-logloss:0.525339
[58]    validation_0-logloss:0.525257
[59]    validation_0-logloss:0.525221
[60]    validation_0-logloss:0.525213
[61]    validation_0-logloss:0.525147
[62]    validation_0-logloss:0.525106
[63]    validation_0-logloss:0.525087
[64]    validation_0-logloss:0.525002
[65]    validation_0-logloss:0.524954
[66]    validation_0-logloss:0.524939
[67]    validation_0-logloss:0.524925
[68]    validation_0-logloss:0.524805
[69]    validation_0-logloss:0.524696
[70]    validation_0-logloss:0.524714
[71]    validation_0-logloss:0.524684
[72]    validation_0-logloss:0.524761
[73]    validation_0-logloss:0.524753
[74]    validation_0-logloss:0.524704
[75]    validation_0-logloss:0.524771
```

```
[76]    validation_0-logloss:0.524668
[77]    validation_0-logloss:0.524585
[78]    validation_0-logloss:0.524529
[79]    validation_0-logloss:0.524551
[80]    validation_0-logloss:0.524544
[81]    validation_0-logloss:0.524613
[82]    validation_0-logloss:0.524613
[83]    validation_0-logloss:0.524509
[84]    validation_0-logloss:0.524453
[85]    validation_0-logloss:0.524469
[86]    validation_0-logloss:0.524457
[87]    validation_0-logloss:0.524396
[88]    validation_0-logloss:0.524423
[89]    validation_0-logloss:0.52443
[90]    validation_0-logloss:0.524404
[91]    validation_0-logloss:0.524334
[92]    validation_0-logloss:0.524275
[93]    validation_0-logloss:0.524281
[94]    validation_0-logloss:0.524174
[95]    validation_0-logloss:0.524208
[96]    validation_0-logloss:0.524261
[97]    validation_0-logloss:0.524284
[98]    validation_0-logloss:0.524207
[99]    validation_0-logloss:0.524223
[100]   validation_0-logloss:0.524248
[101]   validation_0-logloss:0.524176
[102]   validation_0-logloss:0.52417
[103]   validation_0-logloss:0.524229
[104]   validation_0-logloss:0.524247
[105]   validation_0-logloss:0.524264
[106]   validation_0-logloss:0.524194
[107]   validation_0-logloss:0.524156
[108]   validation_0-logloss:0.524193
[109]   validation_0-logloss:0.524222
[110]   validation_0-logloss:0.524276
[111]   validation_0-logloss:0.524163
[112]   validation_0-logloss:0.524145
[113]   validation_0-logloss:0.524177
[114]   validation_0-logloss:0.524134
[115]   validation_0-logloss:0.524138
[116]   validation_0-logloss:0.524111
[117]   validation_0-logloss:0.524105
[118]   validation_0-logloss:0.524109
[119]   validation_0-logloss:0.52409
[120]   validation_0-logloss:0.524031
[121]   validation_0-logloss:0.524091
[122]   validation_0-logloss:0.524133
[123]   validation_0-logloss:0.52412
```

```
[124]   validation_0-logloss:0.524088
[125]   validation_0-logloss:0.524116
[126]   validation_0-logloss:0.524142
[127]   validation_0-logloss:0.524087
[128]   validation_0-logloss:0.524027
[129]   validation_0-logloss:0.523992
[130]   validation_0-logloss:0.523961
[131]   validation_0-logloss:0.523974
[132]   validation_0-logloss:0.524048
[133]   validation_0-logloss:0.52404
[134]   validation_0-logloss:0.524024
[135]   validation_0-logloss:0.524108
[136]   validation_0-logloss:0.52401
[137]   validation_0-logloss:0.524009
[138]   validation_0-logloss:0.523918
[139]   validation_0-logloss:0.523912
[140]   validation_0-logloss:0.52387
[141]   validation_0-logloss:0.523875
[142]   validation_0-logloss:0.523817
[143]   validation_0-logloss:0.523781
[144]   validation_0-logloss:0.523781
[145]   validation_0-logloss:0.523745
[146]   validation_0-logloss:0.523742
[147]   validation_0-logloss:0.523783
[148]   validation_0-logloss:0.52374
[149]   validation_0-logloss:0.523742
[150]   validation_0-logloss:0.523784
[151]   validation_0-logloss:0.523746
[152]   validation_0-logloss:0.523761
[153]   validation_0-logloss:0.523795
[154]   validation_0-logloss:0.523814
[155]   validation_0-logloss:0.523799
[156]   validation_0-logloss:0.523801
[157]   validation_0-logloss:0.523845
[158]   validation_0-logloss:0.523807
[159]   validation_0-logloss:0.523786
[160]   validation_0-logloss:0.523836
[161]   validation_0-logloss:0.523797
[162]   validation_0-logloss:0.523717
[163]   validation_0-logloss:0.523743
[164]   validation_0-logloss:0.523708
[165]   validation_0-logloss:0.523638
[166]   validation_0-logloss:0.523662
[167]   validation_0-logloss:0.523558
[168]   validation_0-logloss:0.523583
[169]   validation_0-logloss:0.523613
[170]   validation_0-logloss:0.523633
[171]   validation_0-logloss:0.523592
```

```
[172]   validation_0-logloss:0.523562
[173]   validation_0-logloss:0.523575
[174]   validation_0-logloss:0.52356
[175]   validation_0-logloss:0.523613
[176]   validation_0-logloss:0.523613
[177]   validation_0-logloss:0.523587
[178]   validation_0-logloss:0.523572
[179]   validation_0-logloss:0.523578
[180]   validation_0-logloss:0.523569
[181]   validation_0-logloss:0.523562
[182]   validation_0-logloss:0.523621
[183]   validation_0-logloss:0.523592
[184]   validation_0-logloss:0.523581
[185]   validation_0-logloss:0.523643
[186]   validation_0-logloss:0.523633
[187]   validation_0-logloss:0.523557
[188]   validation_0-logloss:0.52355
[189]   validation_0-logloss:0.523479
[190]   validation_0-logloss:0.523517
[191]   validation_0-logloss:0.52349
[192]   validation_0-logloss:0.523475
[193]   validation_0-logloss:0.523481
[194]   validation_0-logloss:0.523457
[195]   validation_0-logloss:0.523468
[196]   validation_0-logloss:0.523424
[197]   validation_0-logloss:0.523464
[198]   validation_0-logloss:0.52348
[199]   validation_0-logloss:0.523518
[200]   validation_0-logloss:0.523545
[201]   validation_0-logloss:0.523528
[202]   validation_0-logloss:0.523458
[203]   validation_0-logloss:0.523472
[204]   validation_0-logloss:0.523409
[205]   validation_0-logloss:0.523386
[206]   validation_0-logloss:0.52338
[207]   validation_0-logloss:0.523357
[208]   validation_0-logloss:0.523352
[209]   validation_0-logloss:0.523334
[210]   validation_0-logloss:0.523326
[211]   validation_0-logloss:0.523302
[212]   validation_0-logloss:0.523285
[213]   validation_0-logloss:0.523358
[214]   validation_0-logloss:0.523281
[215]   validation_0-logloss:0.523196
[216]   validation_0-logloss:0.523187
[217]   validation_0-logloss:0.523183
[218]   validation_0-logloss:0.523227
[219]   validation_0-logloss:0.523221
```

```
[220]   validation_0-logloss:0.523247
[221]   validation_0-logloss:0.52327
[222]   validation_0-logloss:0.523257
[223]   validation_0-logloss:0.523176
[224]   validation_0-logloss:0.523187
[225]   validation_0-logloss:0.523231
[226]   validation_0-logloss:0.523205
[227]   validation_0-logloss:0.523211
[228]   validation_0-logloss:0.523185
[229]   validation_0-logloss:0.523169
[230]   validation_0-logloss:0.523174
[231]   validation_0-logloss:0.523196
[232]   validation_0-logloss:0.523207
[233]   validation_0-logloss:0.523166
[234]   validation_0-logloss:0.523174
[235]   validation_0-logloss:0.523193
[236]   validation_0-logloss:0.523189
[237]   validation_0-logloss:0.523152
[238]   validation_0-logloss:0.523156
[239]   validation_0-logloss:0.523189
[240]   validation_0-logloss:0.523147
[241]   validation_0-logloss:0.52318
[242]   validation_0-logloss:0.523167
[243]   validation_0-logloss:0.523135
[244]   validation_0-logloss:0.523147
[245]   validation_0-logloss:0.523133
[246]   validation_0-logloss:0.523214
[247]   validation_0-logloss:0.523138
[248]   validation_0-logloss:0.52315
[249]   validation_0-logloss:0.523113
[250]   validation_0-logloss:0.523111
[251]   validation_0-logloss:0.523016
[252]   validation_0-logloss:0.523078
[253]   validation_0-logloss:0.523039
[254]   validation_0-logloss:0.522993
[255]   validation_0-logloss:0.522966
[256]   validation_0-logloss:0.522963
[257]   validation_0-logloss:0.522983
[258]   validation_0-logloss:0.523019
[259]   validation_0-logloss:0.523057
[260]   validation_0-logloss:0.523042
[261]   validation_0-logloss:0.522994
[262]   validation_0-logloss:0.523014
[263]   validation_0-logloss:0.523032
[264]   validation_0-logloss:0.523008
[265]   validation_0-logloss:0.522995
[266]   validation_0-logloss:0.523041
[267]   validation_0-logloss:0.523036
```

```
[268]   validation_0-logloss:0.523055
[269]   validation_0-logloss:0.523125
[270]   validation_0-logloss:0.523122
[271]   validation_0-logloss:0.523181
[272]   validation_0-logloss:0.523125
[273]   validation_0-logloss:0.523118
[274]   validation_0-logloss:0.523064
[275]   validation_0-logloss:0.523081
[276]   validation_0-logloss:0.523057
[277]   validation_0-logloss:0.523009
[278]   validation_0-logloss:0.523013
[279]   validation_0-logloss:0.522962
[280]   validation_0-logloss:0.522897
[281]   validation_0-logloss:0.522945
[282]   validation_0-logloss:0.522953
[283]   validation_0-logloss:0.522933
[284]   validation_0-logloss:0.522956
[285]   validation_0-logloss:0.522909
[286]   validation_0-logloss:0.522906
[287]   validation_0-logloss:0.522869
[288]   validation_0-logloss:0.522891
[289]   validation_0-logloss:0.522901
[290]   validation_0-logloss:0.522924
[291]   validation_0-logloss:0.522899
[292]   validation_0-logloss:0.522856
[293]   validation_0-logloss:0.522861
[294]   validation_0-logloss:0.522865
[295]   validation_0-logloss:0.522868
[296]   validation_0-logloss:0.522872
[297]   validation_0-logloss:0.52285
[298]   validation_0-logloss:0.522838
[299]   validation_0-logloss:0.522868
[300]   validation_0-logloss:0.522779
[301]   validation_0-logloss:0.522748
[302]   validation_0-logloss:0.522775
[303]   validation_0-logloss:0.522733
[304]   validation_0-logloss:0.522801
[305]   validation_0-logloss:0.522866
[306]   validation_0-logloss:0.522859
[307]   validation_0-logloss:0.522825
[308]   validation_0-logloss:0.522775
[309]   validation_0-logloss:0.522742
[310]   validation_0-logloss:0.522662
[311]   validation_0-logloss:0.522707
[312]   validation_0-logloss:0.522746
[313]   validation_0-logloss:0.522679
[314]   validation_0-logloss:0.522658
[315]   validation_0-logloss:0.522656
```

```
[316]   validation_0-logloss:0.522688
[317]   validation_0-logloss:0.522681
[318]   validation_0-logloss:0.52265
[319]   validation_0-logloss:0.522609
[320]   validation_0-logloss:0.522618
[321]   validation_0-logloss:0.522703
[322]   validation_0-logloss:0.522757
[323]   validation_0-logloss:0.52272
[324]   validation_0-logloss:0.522716
[325]   validation_0-logloss:0.522784
[326]   validation_0-logloss:0.522734
[327]   validation_0-logloss:0.522729
[328]   validation_0-logloss:0.52268
[329]   validation_0-logloss:0.522606
[330]   validation_0-logloss:0.522615
[331]   validation_0-logloss:0.522593
[332]   validation_0-logloss:0.522635
[333]   validation_0-logloss:0.522639
[334]   validation_0-logloss:0.522601
[335]   validation_0-logloss:0.522629
[336]   validation_0-logloss:0.522632
[337]   validation_0-logloss:0.522578
[338]   validation_0-logloss:0.522593
[339]   validation_0-logloss:0.522625
[340]   validation_0-logloss:0.522602
[341]   validation_0-logloss:0.522597
[342]   validation_0-logloss:0.522519
[343]   validation_0-logloss:0.522471
[344]   validation_0-logloss:0.522503
[345]   validation_0-logloss:0.52245
[346]   validation_0-logloss:0.522503
[347]   validation_0-logloss:0.52249
[348]   validation_0-logloss:0.522527
[349]   validation_0-logloss:0.5225
[350]   validation_0-logloss:0.522579
[351]   validation_0-logloss:0.522593
[352]   validation_0-logloss:0.522588
[353]   validation_0-logloss:0.522544
[354]   validation_0-logloss:0.522515
[355]   validation_0-logloss:0.522535
[356]   validation_0-logloss:0.522488
[357]   validation_0-logloss:0.522488
[358]   validation_0-logloss:0.522447
[359]   validation_0-logloss:0.52246
[360]   validation_0-logloss:0.522441
[361]   validation_0-logloss:0.522508
[362]   validation_0-logloss:0.52251
[363]   validation_0-logloss:0.52251
```

```
[364]   validation_0-logloss:0.522511
[365]   validation_0-logloss:0.522515
[366]   validation_0-logloss:0.522528
[367]   validation_0-logloss:0.522456
[368]   validation_0-logloss:0.522377
[369]   validation_0-logloss:0.522388
[370]   validation_0-logloss:0.522378
[371]   validation_0-logloss:0.522346
[372]   validation_0-logloss:0.522366
[373]   validation_0-logloss:0.522358
[374]   validation_0-logloss:0.522399
[375]   validation_0-logloss:0.522407
[376]   validation_0-logloss:0.522362
[377]   validation_0-logloss:0.522353
[378]   validation_0-logloss:0.522344
[379]   validation_0-logloss:0.522356
[380]   validation_0-logloss:0.522349
[381]   validation_0-logloss:0.522386
[382]   validation_0-logloss:0.522445
[383]   validation_0-logloss:0.522349
[384]   validation_0-logloss:0.522286
[385]   validation_0-logloss:0.522248
[386]   validation_0-logloss:0.522255
[387]   validation_0-logloss:0.52227
[388]   validation_0-logloss:0.522245
[389]   validation_0-logloss:0.522224
[390]   validation_0-logloss:0.522241
[391]   validation_0-logloss:0.522239
[392]   validation_0-logloss:0.522255
[393]   validation_0-logloss:0.522274
[394]   validation_0-logloss:0.522278
[395]   validation_0-logloss:0.52226
[396]   validation_0-logloss:0.522274
[397]   validation_0-logloss:0.52228
[398]   validation_0-logloss:0.522297
[399]   validation_0-logloss:0.522253
[400]   validation_0-logloss:0.522235
[401]   validation_0-logloss:0.522233
[402]   validation_0-logloss:0.522214
[403]   validation_0-logloss:0.522245
[404]   validation_0-logloss:0.522327
[405]   validation_0-logloss:0.522327
[406]   validation_0-logloss:0.52229
[407]   validation_0-logloss:0.522217
[408]   validation_0-logloss:0.522202
[409]   validation_0-logloss:0.522178
[410]   validation_0-logloss:0.522241
[411]   validation_0-logloss:0.522241
```

```
[412]   validation_0-logloss:0.522179
[413]   validation_0-logloss:0.522197
[414]   validation_0-logloss:0.522152
[415]   validation_0-logloss:0.522076
[416]   validation_0-logloss:0.522089
[417]   validation_0-logloss:0.522048
[418]   validation_0-logloss:0.522138
[419]   validation_0-logloss:0.522119
[420]   validation_0-logloss:0.522143
[421]   validation_0-logloss:0.522143
[422]   validation_0-logloss:0.522061
[423]   validation_0-logloss:0.522088
[424]   validation_0-logloss:0.522087
[425]   validation_0-logloss:0.522071
[426]   validation_0-logloss:0.52207
[427]   validation_0-logloss:0.522009
[428]   validation_0-logloss:0.522006
[429]   validation_0-logloss:0.52202
[430]   validation_0-logloss:0.522026
[431]   validation_0-logloss:0.521983
[432]   validation_0-logloss:0.522028
[433]   validation_0-logloss:0.522019
[434]   validation_0-logloss:0.52206
[435]   validation_0-logloss:0.522014
[436]   validation_0-logloss:0.522005
[437]   validation_0-logloss:0.52199
[438]   validation_0-logloss:0.522029
[439]   validation_0-logloss:0.522087
[440]   validation_0-logloss:0.522074
[441]   validation_0-logloss:0.522082
[442]   validation_0-logloss:0.522139
[443]   validation_0-logloss:0.522142
[444]   validation_0-logloss:0.522154
[445]   validation_0-logloss:0.522154
[446]   validation_0-logloss:0.522174
[447]   validation_0-logloss:0.52221
[448]   validation_0-logloss:0.522184
[449]   validation_0-logloss:0.522167
[450]   validation_0-logloss:0.522092
[451]   validation_0-logloss:0.522131
[452]   validation_0-logloss:0.522121
[453]   validation_0-logloss:0.522068
[454]   validation_0-logloss:0.522026
[455]   validation_0-logloss:0.522044
[456]   validation_0-logloss:0.52199
[457]   validation_0-logloss:0.522001
[458]   validation_0-logloss:0.5219
[459]   validation_0-logloss:0.521873
```

```
[460]   validation_0-logloss:0.521888
[461]   validation_0-logloss:0.521855
[462]   validation_0-logloss:0.521842
[463]   validation_0-logloss:0.521887
[464]   validation_0-logloss:0.521898
[465]   validation_0-logloss:0.521965
[466]   validation_0-logloss:0.52196
[467]   validation_0-logloss:0.522018
[468]   validation_0-logloss:0.522015
[469]   validation_0-logloss:0.522007
[470]   validation_0-logloss:0.521966
[471]   validation_0-logloss:0.521951
[472]   validation_0-logloss:0.521901
[473]   validation_0-logloss:0.521903
[474]   validation_0-logloss:0.521889
[475]   validation_0-logloss:0.521899
[476]   validation_0-logloss:0.521901
[477]   validation_0-logloss:0.521892
[478]   validation_0-logloss:0.521944
[479]   validation_0-logloss:0.521909
[480]   validation_0-logloss:0.521874
[481]   validation_0-logloss:0.521888
[482]   validation_0-logloss:0.521867
[483]   validation_0-logloss:0.521808
[484]   validation_0-logloss:0.521816
[485]   validation_0-logloss:0.521833
[486]   validation_0-logloss:0.521848
[487]   validation_0-logloss:0.521874
[488]   validation_0-logloss:0.521803
[489]   validation_0-logloss:0.52183
[490]   validation_0-logloss:0.521812
[491]   validation_0-logloss:0.521863
[492]   validation_0-logloss:0.521837
[493]   validation_0-logloss:0.521824
[494]   validation_0-logloss:0.5219
[495]   validation_0-logloss:0.521873
[496]   validation_0-logloss:0.521842
[497]   validation_0-logloss:0.52188
[498]   validation_0-logloss:0.521879
[499]   validation_0-logloss:0.521851
[500]   validation_0-logloss:0.521801
[501]   validation_0-logloss:0.521787
[502]   validation_0-logloss:0.521784
[503]   validation_0-logloss:0.521791
[504]   validation_0-logloss:0.521794
[505]   validation_0-logloss:0.521807
[506]   validation_0-logloss:0.521715
[507]   validation_0-logloss:0.52166
```

```
[508]   validation_0-logloss:0.521644
[509]   validation_0-logloss:0.521716
[510]   validation_0-logloss:0.521711
[511]   validation_0-logloss:0.521714
[512]   validation_0-logloss:0.521625
[513]   validation_0-logloss:0.521607
[514]   validation_0-logloss:0.521597
[515]   validation_0-logloss:0.521549
[516]   validation_0-logloss:0.521595
[517]   validation_0-logloss:0.521605
[518]   validation_0-logloss:0.521631
[519]   validation_0-logloss:0.52164
[520]   validation_0-logloss:0.521637
[521]   validation_0-logloss:0.521669
[522]   validation_0-logloss:0.521645
[523]   validation_0-logloss:0.521669
[524]   validation_0-logloss:0.521668
[525]   validation_0-logloss:0.521679
[526]   validation_0-logloss:0.521667
[527]   validation_0-logloss:0.521637
[528]   validation_0-logloss:0.521716
[529]   validation_0-logloss:0.521752
[530]   validation_0-logloss:0.521709
[531]   validation_0-logloss:0.521667
[532]   validation_0-logloss:0.521691
[533]   validation_0-logloss:0.521654
[534]   validation_0-logloss:0.521711
[535]   validation_0-logloss:0.521638
[536]   validation_0-logloss:0.521613
[537]   validation_0-logloss:0.521588
[538]   validation_0-logloss:0.521529
[539]   validation_0-logloss:0.52154
[540]   validation_0-logloss:0.521576
[541]   validation_0-logloss:0.52151
[542]   validation_0-logloss:0.521546
[543]   validation_0-logloss:0.521556
[544]   validation_0-logloss:0.521523
[545]   validation_0-logloss:0.521559
[546]   validation_0-logloss:0.521602
[547]   validation_0-logloss:0.521658
[548]   validation_0-logloss:0.521615
[549]   validation_0-logloss:0.521589
[550]   validation_0-logloss:0.521608
[551]   validation_0-logloss:0.521592
[552]   validation_0-logloss:0.521596
[553]   validation_0-logloss:0.521663
[554]   validation_0-logloss:0.521624
[555]   validation_0-logloss:0.521654
```

```
[556]   validation_0-logloss:0.521639
[557]   validation_0-logloss:0.521636
[558]   validation_0-logloss:0.521594
[559]   validation_0-logloss:0.521582
[560]   validation_0-logloss:0.521501
[561]   validation_0-logloss:0.521512
[562]   validation_0-logloss:0.521507
[563]   validation_0-logloss:0.521468
[564]   validation_0-logloss:0.52146
[565]   validation_0-logloss:0.521412
[566]   validation_0-logloss:0.521395
[567]   validation_0-logloss:0.521379
[568]   validation_0-logloss:0.521381
[569]   validation_0-logloss:0.521334
[570]   validation_0-logloss:0.521322
[571]   validation_0-logloss:0.521375
[572]   validation_0-logloss:0.521386
[573]   validation_0-logloss:0.521472
[574]   validation_0-logloss:0.521493
[575]   validation_0-logloss:0.521432
[576]   validation_0-logloss:0.521422
[577]   validation_0-logloss:0.521414
[578]   validation_0-logloss:0.52136
[579]   validation_0-logloss:0.521353
[580]   validation_0-logloss:0.521363
[581]   validation_0-logloss:0.521351
[582]   validation_0-logloss:0.521379
[583]   validation_0-logloss:0.521392
[584]   validation_0-logloss:0.521362
[585]   validation_0-logloss:0.521301
[586]   validation_0-logloss:0.521364
[587]   validation_0-logloss:0.521336
[588]   validation_0-logloss:0.52128
[589]   validation_0-logloss:0.52128
[590]   validation_0-logloss:0.521304
[591]   validation_0-logloss:0.52127
[592]   validation_0-logloss:0.521266
[593]   validation_0-logloss:0.521357
[594]   validation_0-logloss:0.52134
[595]   validation_0-logloss:0.52132
[596]   validation_0-logloss:0.52136
[597]   validation_0-logloss:0.52137
[598]   validation_0-logloss:0.521367
[599]   validation_0-logloss:0.521332
[600]   validation_0-logloss:0.521304
[601]   validation_0-logloss:0.521322
[602]   validation_0-logloss:0.521286
[603]   validation_0-logloss:0.521207
```

```
[604]   validation_0-logloss:0.521225
[605]   validation_0-logloss:0.521219
[606]   validation_0-logloss:0.521215
[607]   validation_0-logloss:0.521162
[608]   validation_0-logloss:0.521216
[609]   validation_0-logloss:0.5212
[610]   validation_0-logloss:0.521249
[611]   validation_0-logloss:0.521184
[612]   validation_0-logloss:0.521214
[613]   validation_0-logloss:0.521245
[614]   validation_0-logloss:0.521225
[615]   validation_0-logloss:0.521215
[616]   validation_0-logloss:0.521223
[617]   validation_0-logloss:0.521223
[618]   validation_0-logloss:0.52118
[619]   validation_0-logloss:0.521165
[620]   validation_0-logloss:0.521162
[621]   validation_0-logloss:0.521148
[622]   validation_0-logloss:0.52115
[623]   validation_0-logloss:0.521114
[624]   validation_0-logloss:0.521094
[625]   validation_0-logloss:0.521124
[626]   validation_0-logloss:0.521127
[627]   validation_0-logloss:0.521152
[628]   validation_0-logloss:0.521146
[629]   validation_0-logloss:0.52115
[630]   validation_0-logloss:0.521146
[631]   validation_0-logloss:0.52111
[632]   validation_0-logloss:0.521137
[633]   validation_0-logloss:0.521116
[634]   validation_0-logloss:0.521173
[635]   validation_0-logloss:0.521186
[636]   validation_0-logloss:0.521183
[637]   validation_0-logloss:0.521194
[638]   validation_0-logloss:0.521085
[639]   validation_0-logloss:0.52106
[640]   validation_0-logloss:0.521083
[641]   validation_0-logloss:0.521109
[642]   validation_0-logloss:0.521132
[643]   validation_0-logloss:0.521141
[644]   validation_0-logloss:0.521179
[645]   validation_0-logloss:0.521142
[646]   validation_0-logloss:0.521115
[647]   validation_0-logloss:0.521067
[648]   validation_0-logloss:0.521077
[649]   validation_0-logloss:0.521056
[650]   validation_0-logloss:0.521114
[651]   validation_0-logloss:0.521054
```

```
[652]   validation_0-logloss:0.520998
[653]   validation_0-logloss:0.52096
[654]   validation_0-logloss:0.521043
[655]   validation_0-logloss:0.521035
[656]   validation_0-logloss:0.521031
[657]   validation_0-logloss:0.520974
[658]   validation_0-logloss:0.520982
[659]   validation_0-logloss:0.52098
[660]   validation_0-logloss:0.521077
[661]   validation_0-logloss:0.520976
[662]   validation_0-logloss:0.520992
[663]   validation_0-logloss:0.521011
[664]   validation_0-logloss:0.52095
[665]   validation_0-logloss:0.520915
[666]   validation_0-logloss:0.520875
[667]   validation_0-logloss:0.520907
[668]   validation_0-logloss:0.520922
[669]   validation_0-logloss:0.520869
[670]   validation_0-logloss:0.520829
[671]   validation_0-logloss:0.520821
[672]   validation_0-logloss:0.520848
[673]   validation_0-logloss:0.520873
[674]   validation_0-logloss:0.520792
[675]   validation_0-logloss:0.520819
[676]   validation_0-logloss:0.520849
[677]   validation_0-logloss:0.520865
[678]   validation_0-logloss:0.520926
[679]   validation_0-logloss:0.520938
[680]   validation_0-logloss:0.520898
[681]   validation_0-logloss:0.520948
[682]   validation_0-logloss:0.520875
[683]   validation_0-logloss:0.520847
[684]   validation_0-logloss:0.520862
[685]   validation_0-logloss:0.520881
[686]   validation_0-logloss:0.520942
[687]   validation_0-logloss:0.52094
[688]   validation_0-logloss:0.520865
[689]   validation_0-logloss:0.52085
[690]   validation_0-logloss:0.520845
[691]   validation_0-logloss:0.520827
[692]   validation_0-logloss:0.520746
[693]   validation_0-logloss:0.520728
[694]   validation_0-logloss:0.52073
[695]   validation_0-logloss:0.520708
[696]   validation_0-logloss:0.520748
[697]   validation_0-logloss:0.520745
[698]   validation_0-logloss:0.520778
[699]   validation_0-logloss:0.520729
```

```
[700]   validation_0-logloss:0.520701
[701]   validation_0-logloss:0.52074
[702]   validation_0-logloss:0.520772
[703]   validation_0-logloss:0.520788
[704]   validation_0-logloss:0.520779
[705]   validation_0-logloss:0.520775
[706]   validation_0-logloss:0.520786
[707]   validation_0-logloss:0.52078
[708]   validation_0-logloss:0.520754
[709]   validation_0-logloss:0.520728
[710]   validation_0-logloss:0.52084
[711]   validation_0-logloss:0.520852
[712]   validation_0-logloss:0.520858
[713]   validation_0-logloss:0.520873
[714]   validation_0-logloss:0.520914
[715]   validation_0-logloss:0.520877
[716]   validation_0-logloss:0.520879
[717]   validation_0-logloss:0.520843
[718]   validation_0-logloss:0.520836
[719]   validation_0-logloss:0.520815
[720]   validation_0-logloss:0.520838
[721]   validation_0-logloss:0.520854
[722]   validation_0-logloss:0.520838
[723]   validation_0-logloss:0.520802
[724]   validation_0-logloss:0.52076
[725]   validation_0-logloss:0.520724
[726]   validation_0-logloss:0.52072
[727]   validation_0-logloss:0.520763
[728]   validation_0-logloss:0.520748
[729]   validation_0-logloss:0.520705
[730]   validation_0-logloss:0.520685
[731]   validation_0-logloss:0.520722
[732]   validation_0-logloss:0.520753
[733]   validation_0-logloss:0.520775
[734]   validation_0-logloss:0.52076
[735]   validation_0-logloss:0.520671
[736]   validation_0-logloss:0.520741
[737]   validation_0-logloss:0.520758
[738]   validation_0-logloss:0.520764
[739]   validation_0-logloss:0.520763
[740]   validation_0-logloss:0.520778
[741]   validation_0-logloss:0.520744
[742]   validation_0-logloss:0.520728
[743]   validation_0-logloss:0.520743
[744]   validation_0-logloss:0.520684
[745]   validation_0-logloss:0.520708
[746]   validation_0-logloss:0.520763
[747]   validation_0-logloss:0.520741
```

```
[748]   validation_0-logloss:0.520758
[749]   validation_0-logloss:0.520733
[750]   validation_0-logloss:0.520741
[751]   validation_0-logloss:0.520703
[752]   validation_0-logloss:0.520738
[753]   validation_0-logloss:0.520698
[754]   validation_0-logloss:0.520745
[755]   validation_0-logloss:0.520796
[756]   validation_0-logloss:0.520754
[757]   validation_0-logloss:0.520773
[758]   validation_0-logloss:0.520739
[759]   validation_0-logloss:0.520721
[760]   validation_0-logloss:0.520711
[761]   validation_0-logloss:0.520669
[762]   validation_0-logloss:0.520621
[763]   validation_0-logloss:0.520571
[764]   validation_0-logloss:0.520589
[765]   validation_0-logloss:0.520605
[766]   validation_0-logloss:0.520554
[767]   validation_0-logloss:0.520526
[768]   validation_0-logloss:0.52051
[769]   validation_0-logloss:0.520471
[770]   validation_0-logloss:0.520478
[771]   validation_0-logloss:0.520453
[772]   validation_0-logloss:0.52048
[773]   validation_0-logloss:0.520567
[774]   validation_0-logloss:0.52058
[775]   validation_0-logloss:0.52054
[776]   validation_0-logloss:0.520564
[777]   validation_0-logloss:0.520525
[778]   validation_0-logloss:0.520495
[779]   validation_0-logloss:0.520542
[780]   validation_0-logloss:0.52049
[781]   validation_0-logloss:0.520506
[782]   validation_0-logloss:0.520534
[783]   validation_0-logloss:0.520547
[784]   validation_0-logloss:0.520576
[785]   validation_0-logloss:0.520623
[786]   validation_0-logloss:0.520652
[787]   validation_0-logloss:0.520623
[788]   validation_0-logloss:0.520621
[789]   validation_0-logloss:0.520573
[790]   validation_0-logloss:0.520552
[791]   validation_0-logloss:0.520563
[792]   validation_0-logloss:0.520567
[793]   validation_0-logloss:0.520557
[794]   validation_0-logloss:0.520516
[795]   validation_0-logloss:0.520521
```

```
[796]   validation_0-logloss:0.52054
[797]   validation_0-logloss:0.520485
[798]   validation_0-logloss:0.520543
[799]   validation_0-logloss:0.520589
[800]   validation_0-logloss:0.52058
[801]   validation_0-logloss:0.520576
[802]   validation_0-logloss:0.520593
[803]   validation_0-logloss:0.520543
[804]   validation_0-logloss:0.52051
[805]   validation_0-logloss:0.520505
[806]   validation_0-logloss:0.520433
[807]   validation_0-logloss:0.520498
[808]   validation_0-logloss:0.520478
[809]   validation_0-logloss:0.520492
[810]   validation_0-logloss:0.520448
[811]   validation_0-logloss:0.520416
[812]   validation_0-logloss:0.520409
[813]   validation_0-logloss:0.520397
[814]   validation_0-logloss:0.520379
[815]   validation_0-logloss:0.520345
[816]   validation_0-logloss:0.520339
[817]   validation_0-logloss:0.520355
[818]   validation_0-logloss:0.520422
[819]   validation_0-logloss:0.520406
[820]   validation_0-logloss:0.520382
[821]   validation_0-logloss:0.520333
[822]   validation_0-logloss:0.52027
[823]   validation_0-logloss:0.520222
[824]   validation_0-logloss:0.52016
[825]   validation_0-logloss:0.52017
[826]   validation_0-logloss:0.520211
[827]   validation_0-logloss:0.520237
[828]   validation_0-logloss:0.52025
[829]   validation_0-logloss:0.520299
[830]   validation_0-logloss:0.520319
[831]   validation_0-logloss:0.520294
[832]   validation_0-logloss:0.520345
[833]   validation_0-logloss:0.520308
[834]   validation_0-logloss:0.520247
[835]   validation_0-logloss:0.520211
[836]   validation_0-logloss:0.520164
[837]   validation_0-logloss:0.520193
[838]   validation_0-logloss:0.520316
[839]   validation_0-logloss:0.520358
[840]   validation_0-logloss:0.520281
[841]   validation_0-logloss:0.520242
[842]   validation_0-logloss:0.520272
[843]   validation_0-logloss:0.52031
```

```
[844]   validation_0-logloss:0.520291
[845]   validation_0-logloss:0.520273
[846]   validation_0-logloss:0.520258
[847]   validation_0-logloss:0.520172
[848]   validation_0-logloss:0.520086
[849]   validation_0-logloss:0.520121
[850]   validation_0-logloss:0.52004
[851]   validation_0-logloss:0.520083
[852]   validation_0-logloss:0.520069
[853]   validation_0-logloss:0.520123
[854]   validation_0-logloss:0.520115
[855]   validation_0-logloss:0.520176
[856]   validation_0-logloss:0.520189
[857]   validation_0-logloss:0.520184
[858]   validation_0-logloss:0.520148
[859]   validation_0-logloss:0.520147
[860]   validation_0-logloss:0.520087
[861]   validation_0-logloss:0.520132
[862]   validation_0-logloss:0.520096
[863]   validation_0-logloss:0.520114
[864]   validation_0-logloss:0.520122
[865]   validation_0-logloss:0.520089
[866]   validation_0-logloss:0.520091
[867]   validation_0-logloss:0.520167
[868]   validation_0-logloss:0.520185
[869]   validation_0-logloss:0.520239
[870]   validation_0-logloss:0.520145
[871]   validation_0-logloss:0.520141
[872]   validation_0-logloss:0.520156
[873]   validation_0-logloss:0.520223
[874]   validation_0-logloss:0.520199
[875]   validation_0-logloss:0.520214
[876]   validation_0-logloss:0.520164
[877]   validation_0-logloss:0.520173
[878]   validation_0-logloss:0.520199
[879]   validation_0-logloss:0.520191
[880]   validation_0-logloss:0.520171
[881]   validation_0-logloss:0.520188
[882]   validation_0-logloss:0.520156
[883]   validation_0-logloss:0.520159
[884]   validation_0-logloss:0.520115
[885]   validation_0-logloss:0.520153
[886]   validation_0-logloss:0.520194
[887]   validation_0-logloss:0.520197
[888]   validation_0-logloss:0.520206
[889]   validation_0-logloss:0.520203
[890]   validation_0-logloss:0.520253
[891]   validation_0-logloss:0.520147
```

```
[892]   validation_0-logloss:0.520133
[893]   validation_0-logloss:0.520077
[894]   validation_0-logloss:0.520092
[895]   validation_0-logloss:0.520096
[896]   validation_0-logloss:0.520133
[897]   validation_0-logloss:0.520128
[898]   validation_0-logloss:0.520108
[899]   validation_0-logloss:0.520093
[900]   validation_0-logloss:0.520157
Stopping. Best iteration:
[850]   validation_0-logloss:0.52004
```

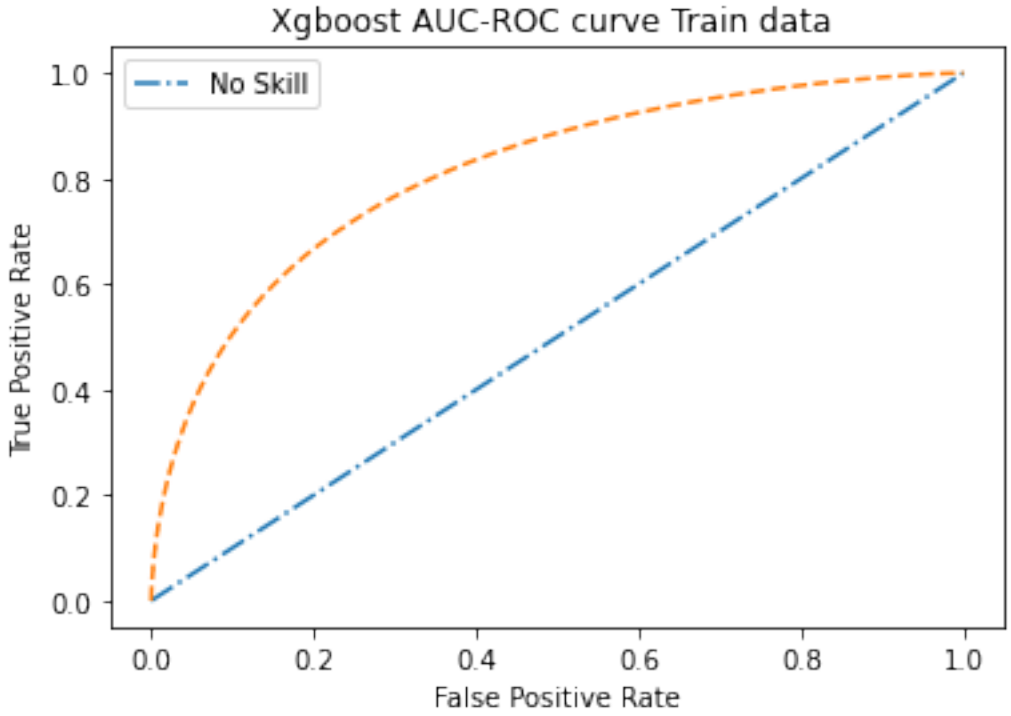
[60]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.7736610394413841, eta=0.1,
              gamma=1, learning_rate=0.1, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=None, n_estimators=1800, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=9, seed=None,
              silent=None, subsample=0.598690048807158, tree_method='gpu_hist',
              verbosity=1)
```

[61]:
```
# making predictions
y_train_proba=xgboost.predict_proba(X_train)[:,1]
y_test_proba=xgboost.predict_proba(X_test)[:,1]
y_train_pred=np.where(y_train_proba>0.5,1 ,0)
y_test_pred=np.where(y_test_proba>0.5,1 ,0)
```

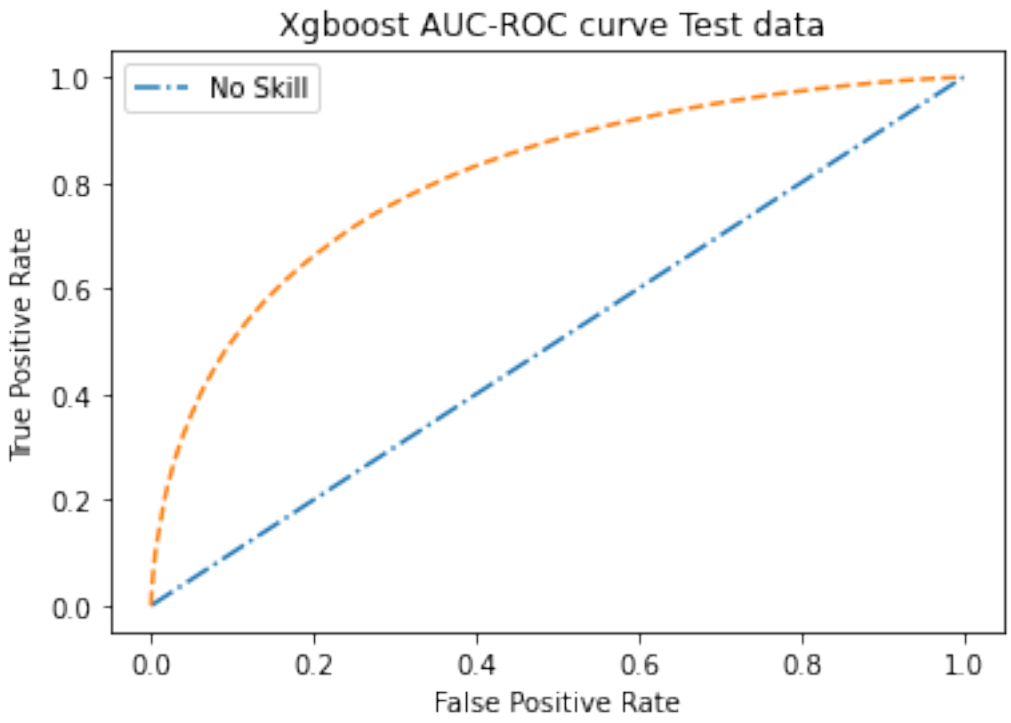
[62]:
```
printModelPerformance(y_train ,y_train_pred,y_test ,y_test_pred  )
plotAUC_ROC_Curve(y_train ,y_train_proba , title ="Xgboost AUC-ROC curve Train␣
 ↪data ")
plotAUC_ROC_Curve(y_test ,y_test_proba , title ="Xgboost AUC-ROC curve Test␣
 ↪data ")
plot_Precision_Recall(y_train ,y_train_proba ,title="Xgboost Precision-Recall␣
 ↪curve Train data")
plot_Precision_Recall(y_test ,y_test_proba ,title="Xgboost Precision-Recall␣
 ↪curve Test data")
```

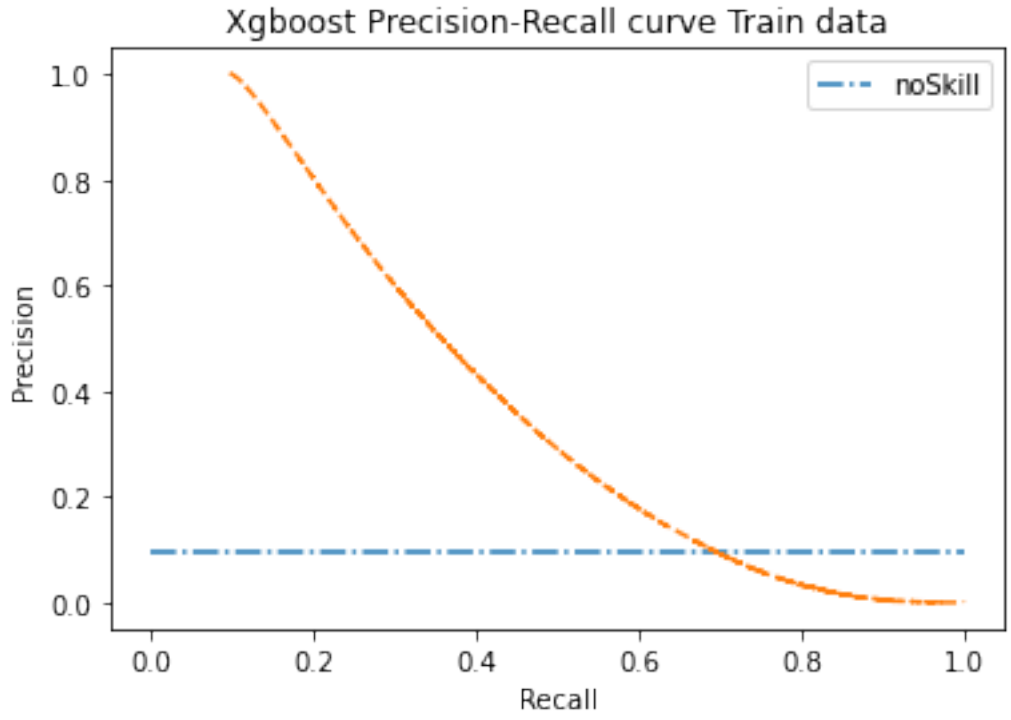
```
Model Performance
Train:
Accuracy 0.7568796270292849
True positive rate  0.7087378347929822
False positive rate  0.23790170761242763
False negative  rate  0.29126216520701775
True negative  rate  0.7620982923875724
Precision  0.24410927011102274
Recall 0.7087378347929822
F1 rate 0.36314215504451036
```

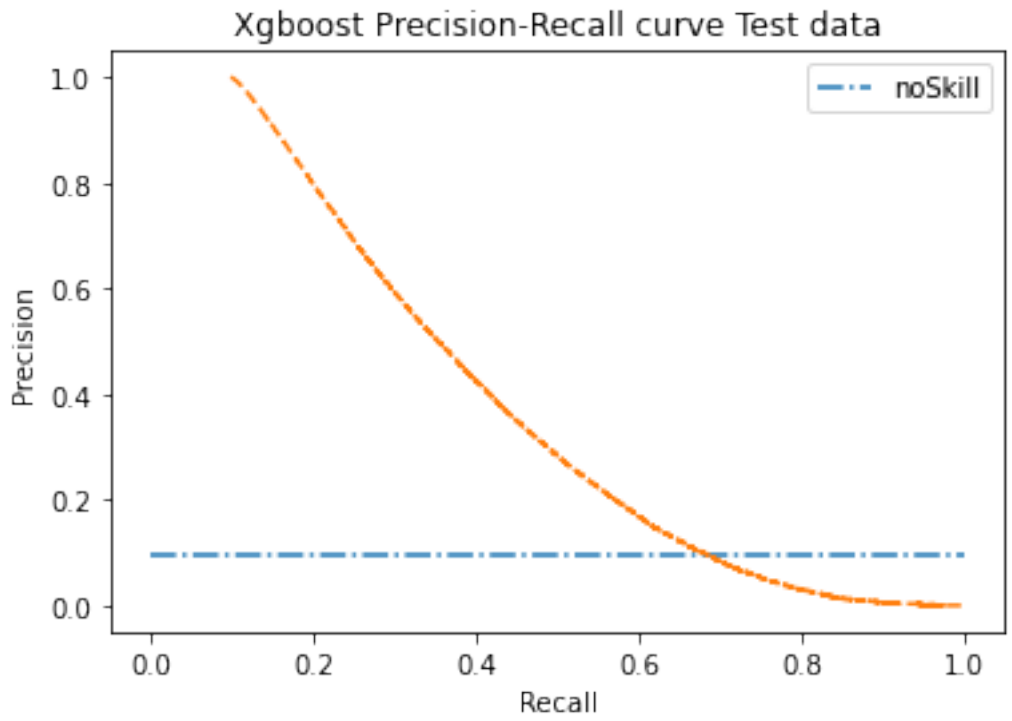
```
Test:
Accuracy 0.7556652681846421
True positive rate  0.7060477181552197
False positive rate  0.2389560859238488
False negative  rate  0.2939522818447803
True negative  rate  0.7610439140761512
Precision  0.24259496478333148
Recall 0.7060477181552197
F1 rate 0.3611130395074413

AUC score  0.8114635662651755
```




```
AUC score  0.8082424536396517
```


Xgboost AUC-ROC curve Test data
No Skill
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

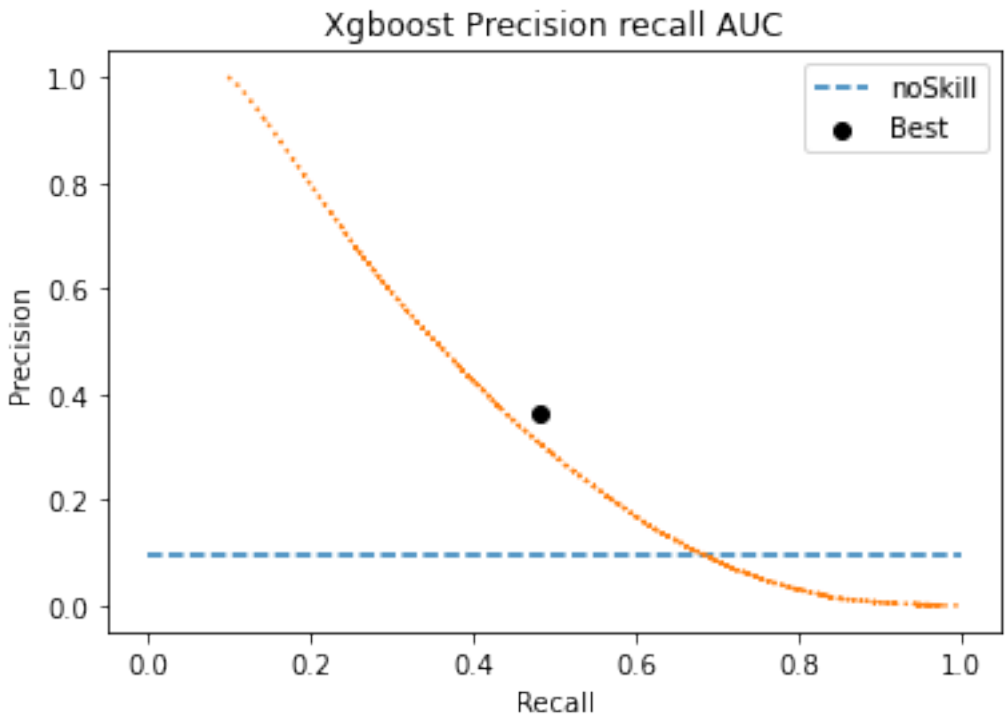

Xgboost Precision-Recall curve Train data
noSkill
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0



- Finding the optimal threshold from precision recall to maximize f1 score

[63]:
```
thresh_xg=get_optimal_F1(y_test ,y_test_proba, label="Xgboost Precision recall␣
↪AUC")
Model_Perpormance_after_Thresholding(y_train,y_train_proba ,y_test,␣
↪y_test_proba , thresh_xg )
```

```
Best Threshold=0.692886, F-Score=0.415
```




```
Model Performance
Train:
Accuracy 0.8682804088895602
True positive rate  0.482855975109304
False positive rate  0.08993882127674392
False negative  rate  0.517144024890696
True negative  rate  0.9100611787232561
Precision  0.3678805177136764
Recall 0.482855975109304
F1 rate 0.41759888661861927
Test:
Accuracy 0.867562316622545
True positive rate  0.4810907006907369
False positive rate  0.09054335429462296
False negative  rate  0.5189092993092631
True negative  rate  0.909456645705377
Precision  0.365474510091474
Recall 0.4810907006907369
F1 rate 0.41538770056957725
```

Preparing data for submission in kaggle

[64]:
```
# reading test data
test =pd.read_parquet(dir+'test.gzip')
```

```
test_temp = test[["order_id", "product_id"]]
# preparing test data
test.drop(columns =["order_id",'ur_pr_count' ,"user_id" ,'product_id'␣
↪,'department_id' ,'aisle_id' ,'ur_pr_count'] , inplace =True)
# standardizing test data
test , test_preprocessing_object=standardize(None, test␣
↪,test_preprocessing_object , flag ='test')
# One hot encoding test data
# test =ohe_Test(test , test_preprocessing_object )
response_code_test( test ,reponse_dict)
# making prediction on test data
predict_xg_test =xgboost.predict_proba(test)[:,1]
```

```
ur_pr_reordered
order_number
ttl_cnt_product_user
Avg_no_prod_perOrder
days_since_prior_order
usr_ro_ratio
product_name_length
```

[65]:
```
submission_xg=generate_SubmissionCsv_colob(test_temp ,predict_xg_test␣
↪,thresh_xg)
```

[66]:
```
submission_xg.to_csv(dir+"xg_boost_850_.csv",index=False)
```

9 Submission Score

- private : 0.35466
- public :0.35528

10 Model performance comparision on test data

[68]:
```
index  =['AUC','TPR' ,'FPR' ,'FNR' ,'TNR' ,'Precision','Recall' ,'F1','thresh'␣
↪,'private_lb','public_lb']
data ={
 'Knn':[0.720,0.276 ,0.043,0.723,0.956,0.407,0.276,0.329,0.661 ,0.308,0.309],
'SVC':[0.61,0.376,0.219,0.623,0.623,0.780,0.155,0.376,0.220 ,0.198,0.202],
'Logistic_Regression':[0.790,0.441,0.083,0.558,0.916,0.363,0.441,0.398,0.661 ,0.
↪34327,0.34449],
'Gaussion_NB':[0.766,0.404,0.0743,0.595,0.925,0.370,0.404,0.387,0.578 ,0.
↪33670,0.333727],
'Bernoulli_NB':[0.767,0.3605,0.0610,0.63,0.93,0.390,0.360,0.374,0.501 ,0.
↪32720,0.32735],
```

```
'Decision_tree':[0.708,0.3939,0.0985,0.6060,0.9014,0.3022,0.3939,0.3420,0.2777␣
↪,0.27510,0.27767],
'Random_Forest':[0.8077,0.4789,0.0880,0.5210,0.9119,0.3708,0.4789,0.4180,0.2085␣
↪,0.34993,0.35191],
'XGboost':[0.8082,0.4810,0.0905,0.5189,0.9094,0.3654,0.4810,0.4153,0.692886 ,0.
↪35466,0.35528]
}

performance=pd.DataFrame(data , index=index)
```

[69]:
```
performance
```

[69]:

	Knn	SVC	…	Random_Forest	XGboost
AUC	0.720	0.610	…	0.80770	0.808200
TPR	0.276	0.376	…	0.47890	0.481000
FPR	0.043	0.219	…	0.08800	0.090500
FNR	0.723	0.623	…	0.52100	0.518900
TNR	0.956	0.623	…	0.91190	0.909400
Precision	0.407	0.780	…	0.37080	0.365400
Recall	0.276	0.155	…	0.47890	0.481000
F1	0.329	0.376	…	0.41800	0.415300
thresh	0.661	0.220	…	0.20850	0.692886
private_lb	0.308	0.198	…	0.34993	0.354660
public_lb	0.309	0.202	…	0.35191	0.355280

[11 rows x 8 columns]

- Auc : Xgboost has the highiest AUC
- TPR/Recall :Xgboost has highiest TPR/Recall
- FPR :KNN has the lowest FPR
- FNR :Xgboost has the lowest FNR
- TNR :KNN has highest TNR
- Precision :Bernouli NB has highest Precision
- F1 : Random Forest has highest F1
- Private lb: Xgboost has highest lb
- Public lb :Xgboost has highest lb
- Over all Xgboost is performing best