1. Business Problem

1.1 Problem Description

The Santander group is a global multinational bank that serves more than 100 million customers through a global network of more than 14,000 branches. The mission of Santander is to help customers and business to prosper.

In this problem, Santander is asking to help them identify which customers will make the specific transaction in the future, irrespective of the amount of money transacted. This is a first that Santander needs to nail in order to personalize the services at scale. The digitalization of everyday lives means that customers expect services to be delivered in a personalized and timely manner. Santander group aims go to a step beyond there is a need to provide a customer a financial service and intends to determine the amount or value of the customer's transaction. This means anticipating customer needs in some concrete, but also simple and personal way.

Loading and Importing the Required Libraries

In [2]:

```
# this is just to know how much time will it take to run this entire ipython notebook
from datetime import datetime
# globalstart = datetime.now()
import pandas as pd
import numpy as np
import matplotlib
matplotlib.use('nbagg')

import matplotlib.pyplot as plt
plt.rcParams.update({'figure.max_open_warning': 0})

import seaborn as sns
sns.set_style('whitegrid')
import os
from scipy import sparse
from scipy.sparse import csr_matrix

from sklearn.decomposition import TruncatedSVD
from sklearn.metrics.pairwise import cosine_similarity
import random
from tqdm import tqdm
from scipy.stats import norm, skew
random_state = 42
np.random.seed(random_state)
from sklearn.model_selection import StratifiedKFold,RepeatedKFold
import lightgbm as lgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
import xgboost as xgb
from sklearn.preprocessing import StandardScaler
import pickle
```

Loading the data

In [3]:

```
train = pd.read_csv('C:/Divakars/Self/AppliedAi/Assgn/Self Case Study - 1/train.csv')
test = pd.read_csv('C:/Divakars/Self/AppliedAi/Assgn/Self Case Study - 1/test.csv')
#features = [c for c in train.columns if c not in ['ID_code', 'target']]
```

Exploratory Data Analysis

In [4]:

```
train.head()
```

Out[4]:

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	...	var_190	var_191	var_192	var_193	var_194
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	...	4.4354	3.9642	3.1364	1.6910	18.5227
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	...	7.6421	7.7214	2.5837	10.9516	15.4305
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	...	2.9057	9.7905	1.6704	1.6858	21.6042
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	...	4.4666	4.7433	0.7178	1.4214	23.0347
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	...	-1.4905	9.5214	-0.1508	9.1942	13.2876

5 rows × 202 columns

In [5]:

```
test.head()
```

Out[5]:

	ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	...	var_190	var_191	var_192	var_193	var_
0	test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	...	-2.1556	11.8495	-1.4300	2.4508	13.7
1	test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	...	10.6165	8.8349	0.9403	10.1282	15.5
2	test_2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233	...	-0.7484	10.9935	1.9803	2.1800	12.9
3	test_3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755	...	9.5702	9.0766	1.6580	3.5813	15.1
4	test_4	11.7058	-0.1327	14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890	...	4.2259	9.1723	1.2835	3.3778	19.5

5 rows × 201 columns

In [6]:

```
train.describe()
```

Out[6]:

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	2000
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.078333	-5.065317	5.408949	
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.623150	7.863267	0.866607	
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.074800	-32.562600	2.347300	
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.883175	-11.200350	4.767700	
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.108250	-4.833150	5.385100	
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.261125	0.924800	6.003000	
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.671400	17.251600	8.447700	

8 rows × 201 columns

In [7]:

```
test.describe()
```

Out[7]:

	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	2000
mean	10.658737	-1.624244	10.707452	6.788214	11.076399	-5.050558	5.415164	16.529143	
std	3.036716	4.040509	2.633888	2.052724	1.616456	7.869293	0.864686	3.424482	
min	0.188700	-15.043400	2.355200	-0.022400	5.484400	-27.767000	2.216400	5.713700	
25%	8.442975	-4.700125	8.735600	5.230500	9.891075	-11.201400	4.772600	13.933900	
50%	10.513800	-1.590500	10.560700	6.822350	11.099750	-4.834100	5.391600	16.422700	
75%	12.739600	1.343400	12.495025	8.327600	12.253400	0.942575	6.005800	19.094550	
max	22.323400	9.385100	18.714100	13.142000	16.037100	17.253700	8.302500	28.292800	

8 rows × 200 columns

Data Overview

- The data contains 200,000 rows of both test and train data. Training set have 202 features and Test set have 201 features.
- One extra set in Train set is Target feature which is the class of target
- The train data contains val_0 to val_199 columns, ‘ID_code’ column and ‘target’ column.
- ‘ID_code’ feature is the unique ID of the row.
- 'Target' feature is binary where 1 = transaction and 0 = no transaction
- The other features are anonymized and are labelled from 'var_0' to 'var_199'

Few observations can be made here:

1. Mean values for test and train data looks quite close.
2. Min, max and std values for train and test data also look quite close

Checking for NaN values

In [44]:

```
print("No of Nan values in train data : ", sum(train.isnull().any()))
```

```
No of Nan values in train data :  0
```

In [6]:

```
print("No of Nan values in train data : ", sum(test.isnull().any()))
```

```
No of Nan values in train data :  0
```

It looks like there is no null value present in the train or test data

Checking Correlation With Target

In [28]:

```
corr = train.corr()
```

In [29]:

```
abs(corr['target']).sort_values(ascending=False)
```

Out[29]:

```
target     1.000000
var_81     0.080917
var_139    0.074080
var_12     0.069489
var_6      0.066731
             ...
var_38     0.000970
var_17     0.000864
var_30     0.000638
var_27     0.000582
var_185    0.000053
Name: target, Length: 201, dtype: float64
```

The largest correlation value to target is 0.08. So, from this it could be concluded that correlation is not enough to make a judgement.

Corelation between the features

In [34]:

```
train_corr = corr.abs().unstack().sort_values(kind="quicksort").reset_index()
train_corr.drop(train_corr.iloc[1::2].index, inplace=True)
train_corr_mod = train_corr.drop(train_corr[train_corr[0] == 1.0].index)
```

Top 5 high correlated features in Train data

In [36]:

```
train_corr_mod.tail()
```

Out[36]:

	level_0	level_1	0
40190	var_110	target	0.064275
40192	var_6	target	0.066731
40194	var_12	target	0.069489
40196	target	var_139	0.074080
40198	target	var_81	0.080917

In [40]:

```
test_corr = test.corr().abs().unstack().sort_values(kind="quicksort").reset_index()
test_corr_mod = test_corr.drop(test_corr[test_corr[0] == 1.0].index)
```

Top 5 high correlated features in Test data

In [43]:

```
test_corr_mod.tail()
```

Out[43]:

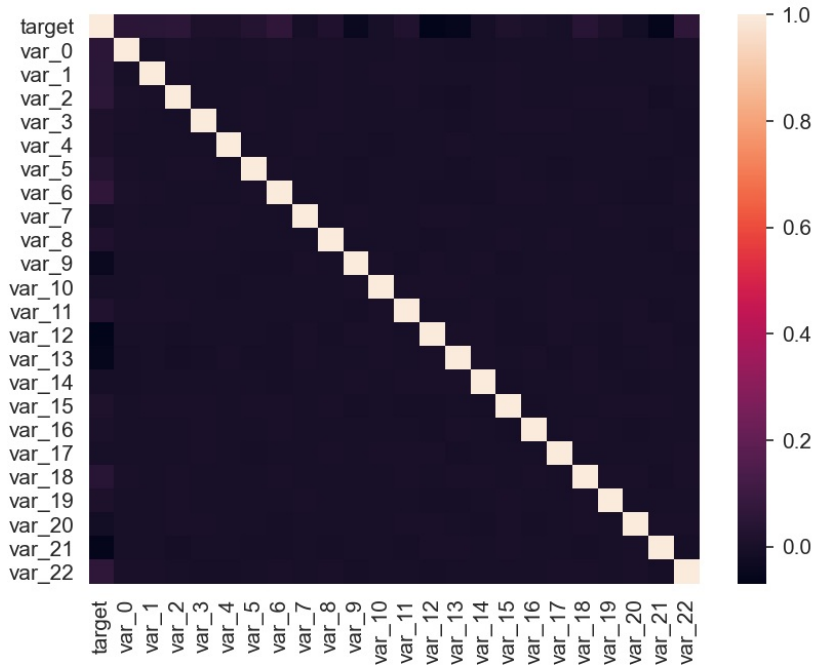
	level_0	level_1	0
39795	var_132	var_31	0.008714
39796	var_96	var_143	0.008829
39797	var_143	var_96	0.008829
39798	var_139	var_75	0.009868
39799	var_75	var_139	0.009868

In [10]:

```
train_50=train.iloc[:,1:25]

train_50_corr = train_50.corr()

sns.heatmap(train_50_corr,xticklabels=train_50_corr.columns,
        yticklabels=train_50_corr.columns)
plt.show()
```



From the heatmap it looks like there has been extremely low correlation between the features. Perhaps the features have been decorrelated by some transformation.

Distribution of Target

In [7]:

```
ones = train['target'].value_counts()[1]

zeros = train['target'].value_counts()[0]

perc_ones = ones / train.shape[0] * 100

perc_zeros = zeros / train.shape[0] * 100

print('{} out of {} training data rows have made transaction which is {:.2f}% of total data'.format(ones, train.s
hape[0], perc_ones))

print('{} out of {} training data rows have not made transaction which is {:.2f}% of total data'.format(zeros, tr
ain.shape[0], perc_zeros))

#sns.countplot(train['target'], palette='Set3')
%matplotlib inline
plt.title('Distribution of targets over Training dataset', fontsize=15)
sns.countplot(train['target'])
plt.xticks((0, 1), ['Class 0 ({0:.2f}%)'.format(perc_zeros), 'Class 1 ({0:.2f}%)'.format(perc_ones)])
plt.show()
```

```
20098 out of 200000 training data rows have made transaction which is 10.05% of total data
179902 out of 200000 training data rows have not made transaction which is 89.95% of total data
```

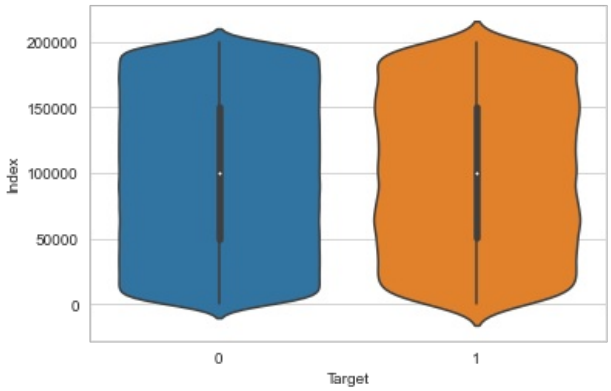
From the above plot it is clear that there is imbalance of target variables. The target is overwhelmingly zero in most of the cases. The number of customers who did not make transaction is much higher than those that made. There are about 175000 samples where the transaction was not made while only in less than 25000 samples the transaction was made.

In [8]:

```
ax = sns.violinplot(x=train.target.values, y=train.index.values)
sns.stripplot(x=train.target.values, y=train.index.values,jitter=True, color="black", size=0.5, alpha=0.5)

ax.set_xlabel("Target")
ax.set_ylabel("Index");
#ax[0].set_xlabel("Target")
#ax[0].set_ylabel("Counts");

plt.show()
```



From the plot it looks like there is no co-relation of the targets with the indexes. From the Violin plot it looks like targets are uniformally distributed across indexes.

In [9]:

```
train.drop(['ID_code'], axis=1, inplace=True)
```

In [10]:

```
test.drop(['ID_code'], axis=1, inplace=True)
```

Unique Values in Features

In [6]:

```
train_unique_values = train.agg(['nunique']).transpose().sort_values(by = 'nunique')
test_unique_values = test.agg(['nunique']).transpose().sort_values(by='nunique')
```

In [7]:

```
train_unique_values.head()
```

Out[7]:

	nunique
target	2
var_68	451
var_91	7962
var_108	8525
var_103	9376

In [8]:

```
unique_combo = train_unique_values.drop('target').reset_index().merge(test_unique_values.reset_index(), how='left
', right_index=True, left_index=True)

unique_combo.drop(columns=['index_y'], inplace=True)

unique_combo.columns = ['Feature Name', 'Training Set Unique Count', 'Test Set Unique Count']

unique_combo.head()
```

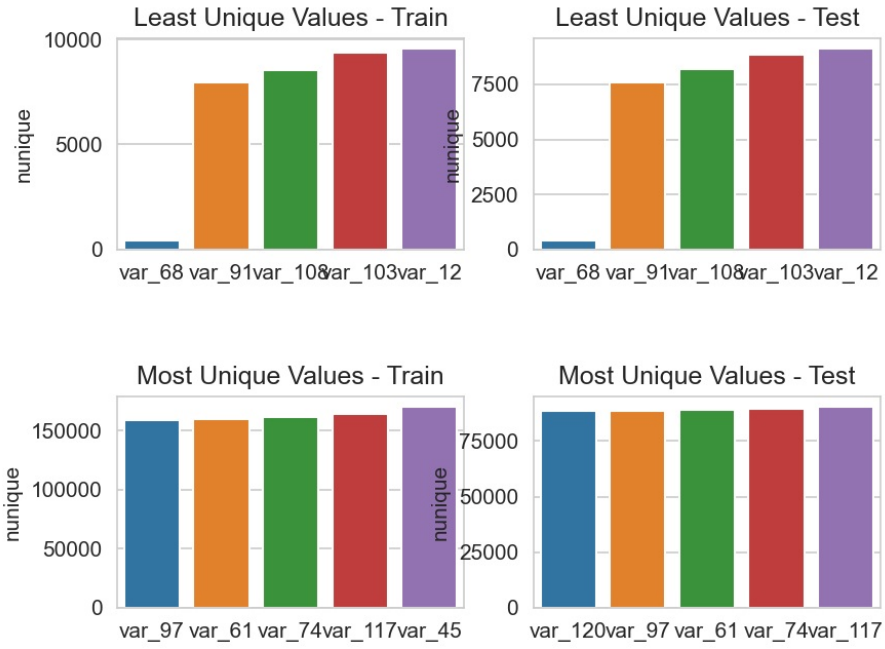
Out[8]:

	Feature Name	Training Set Unique Count	Test Set Unique Count
0	var_68	451	428.0
1	var_91	7962	7569.0
2	var_108	8525	8188.0
3	var_103	9376	8828.0
4	var_12	9561	9121.0

In [55]:

```
fig = plt.figure()
ax1 = fig.add_subplot(221)
sns.barplot(x=train_unique_values.index[1:6], y="nunique", data=train_unique_values[1:].head(), ax=ax1)
ax2 = fig.add_subplot(222)
sns.barplot(x=test_unique_values.index[:5], y="nunique", data=test_unique_values.head(), ax=ax2)
ax3 = fig.add_subplot(223)
sns.barplot(x=train_unique_values.index[-6:-1], y="nunique", data=train_unique_values[-6:-1].tail(), ax=ax3)
ax4 = fig.add_subplot(224)
sns.barplot(x=test_unique_values.index[-6:-1], y="nunique", data=test_unique_values[-6:-1].tail(), ax=ax4)
#fig.tight_layout()
plt.subplots_adjust(hspace = 0.7)

ax1.set_title('Least Unique Values - Train')
ax2.set_title('Least Unique Values - Test')
ax3.title.set_text('Most Unique Values - Train')
ax4.title.set_text('Most Unique Values - Test')
plt.show()
```



Feature Engineering and Data Augmentation

Checking for presence of synthetic samples

The data provided could be checked for the existence of synthetic samples as it would improve the efficiency of the execution of the model.

Given a sample, by going over its features and checking if the features are unique we can check the data for the presence of synthetic samples. If atleast one of the samples feature is unique, then the sample must be a real sample. If the sample has no unique value, then it is synthetic sample.

In [11]:

```
#Ref: https://www.kaggle.com/code/yag320/list-of-fake-samples-and-public-private-lb-split/notebook

unique_samples = []
print(test.head())
df_test = test.values
unique_count = np.zeros_like(df_test)
for feature in tqdm(range(df_test.shape[1])):
    _, index_, count_ = np.unique(df_test[:, feature], return_counts=True, return_index=True)
    unique_count[index_[count_ == 1], feature] += 1

real_samples_indexes = np.argwhere(np.sum(unique_count, axis=1) > 0)[:, 0]
synthetic_samples_indexes = np.argwhere(np.sum(unique_count, axis=1) == 0)[:, 0]


print(len(real_samples_indexes))
print(len(synthetic_samples_indexes))
```

```
     var_0    var_1    var_2   var_3    var_4   var_5   var_6    var_7  \
0  11.0656   7.7798  12.9536  9.4292  11.4327 -2.3805  5.8493  18.2675
1   8.5304   1.2543  11.3047  5.1858   9.1974 -4.0117  6.0196  18.6316
2   5.4827 -10.3581  10.1407  7.0479  10.2628  9.8052  4.8950  20.2537
3   8.5374  -1.3222  12.0220  6.5749   8.8458  3.1744  4.9397  20.5660
4  11.7058  -0.1327  14.1295  7.7506   9.1035 -8.5848  6.8595  10.6048

    var_8   var_9  ...  var_190  var_191  var_192  var_193  var_194  var_195  \
0  2.1337  8.8100  ...  -2.1556  11.8495  -1.4300   2.4508  13.7112   2.4669
1 -4.4131  5.9739  ...  10.6165   8.8349   0.9403  10.1282  15.5765   0.4773
2  1.5233  8.3442  ...  -0.7484  10.9935   1.9803   2.1800  12.9813   2.1281
3  3.3755  7.4578  ...   9.5702   9.0766   1.6580   3.5813  15.1874   3.1656
4  2.9890  7.1437  ...   4.2259   9.1723   1.2835   3.3778  19.5542  -0.2860

   var_196  var_197  var_198  var_199
0   4.3654  10.7200  15.4722  -8.7197
1  -1.4852   9.8714  19.1293 -20.9760
2  -7.1086   7.0618  19.8956 -23.1794
3   3.9567   9.2295  13.0168  -4.2108
4  -5.1612   7.2882  13.9260  -9.1846

[5 rows x 200 columns]

100%|██████████████████████████████████████████████████████████████████████████████| 200/200 [00:0
7<00:00, 27.81it/s]

100000
100000
```

It is evident that there are 100000 each real and synthetic samples present in the test data which is helpful in getting the result. This way by identifying the synthetic data, the unusal bump in the test data can be captured.

In [12]:

```
train.shape
```

Out[12]:

```
(200000, 201)
```

In [13]:

```
target = train['target']
```

In [14]:

```
#Ref: https://www.kaggle.com/code/yag320/list-of-fake-samples-and-public-private-lb-split/notebook
df_test_real = df_test[real_samples_indexes].copy()

generator_for_each_synthetic_sample = []
# Using 20,000 samples should be enough.
# You can use all of the 100,000 and get the same results (but 5 times slower)
for cur_sample_index in tqdm(synthetic_samples_indexes[:20000]):
    cur_synthetic_sample = df_test[cur_sample_index]
    potential_generators = df_test_real == cur_synthetic_sample

    # A verified generator for a synthetic sample is achieved
    # only if the value of a feature appears only once in the
    # entire real samples set
    features_mask = np.sum(potential_generators, axis=0) == 1
    verified_generators_mask = np.any(potential_generators[:, features_mask], axis=1)
    verified_generators_for_sample = real_samples_indexes[np.argwhere(verified_generators_mask)[:, 0]]
    generator_for_each_synthetic_sample.append(set(verified_generators_for_sample))
```

```
100%|██████████████████████████████████████████████████████████████| 20000/20000 [1:13:5
5<00:00,  4.51it/s]
```

In [15]:

```
#Ref: https://www.kaggle.com/code/yag320/list-of-fake-samples-and-public-private-lb-split/notebook
public_LB = generator_for_each_synthetic_sample[0]
for x in tqdm(generator_for_each_synthetic_sample):
    if public_LB.intersection(x):
        public_LB = public_LB.union(x)

private_LB = generator_for_each_synthetic_sample[1]
for x in tqdm(generator_for_each_synthetic_sample):
    if private_LB.intersection(x):
        private_LB = private_LB.union(x)

print(len(public_LB))
print(len(private_LB))
```

```
100%|██████████████████████████████████████████████████████████████| 20000/20000 [01:17
<00:00, 259.35it/s]
100%|██████████████████████████████████████████████████████████████| 20000/20000 [01:15
<00:00, 263.82it/s]

50000
50000
```

In [16]:

```
np.save('public_LB', list(public_LB))
np.save('private_LB', list(private_LB))
np.save('synthetic_samples_indexes', list(synthetic_samples_indexes))
```

In [17]:

```
full = pd.concat([train, pd.concat([test.loc[private_LB], test.loc[public_LB]], sort = False)], sort = False)
```

In [18]:

```
for feat in ['var_' + str(x) for x in range(200)]:
    count_values = full.groupby(feat)[feat].count()
    train['new_' + feat] = count_values.loc[train[feat]].values
    test['new_' + feat] = count_values.loc[test[feat]].values
```

Model - Training and Results

Logistic Regression

In [7]:

```
#X=train.drop(['target','ID_code'], axis=1)
X=train.drop(['target'], axis=1)
y=train.target

logreg = LogisticRegression(solver='sag')

features = [c for c in train.columns if (c not in ['ID_code', 'target'])]

X_train, X_val, y_train, y_val=train_test_split(X, y, test_size=0.2,random_state=42)
```

In [7]:

```
logreg.fit(X_train, y_train)
```

```
C:\Users\dbhat5\Anaconda3\lib\site-packages\sklearn\linear_model\_sag.py:329: ConvergenceWarning: Th
e max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

Out[7]:

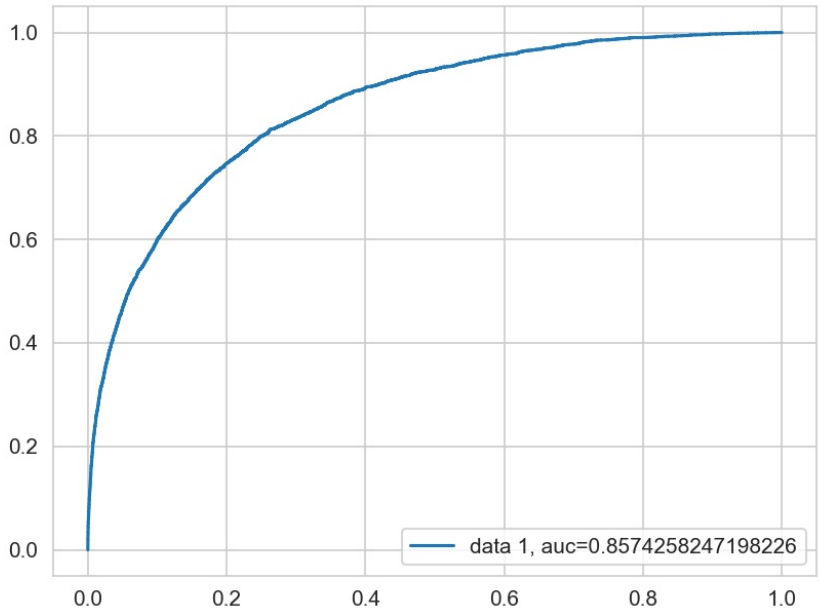
```
LogisticRegression(solver='sag')
```

In [8]:

```
y_pred=logreg.predict(X_val)
```

In [9]:

```
y_pred_proba = logreg.predict_proba(X_val)[::,1]
fpr, tpr, _ = roc_curve(y_val,  y_pred_proba)
auc = roc_auc_score(y_val, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



When working with Logistic Regression an AUC of 0.85742 is achieved.

Naive Bayes

In [8]:

```
clf = GaussianNB()
clf.fit(X_train, y_train)
```

Out[8]:

```
GaussianNB()
```

In [9]:

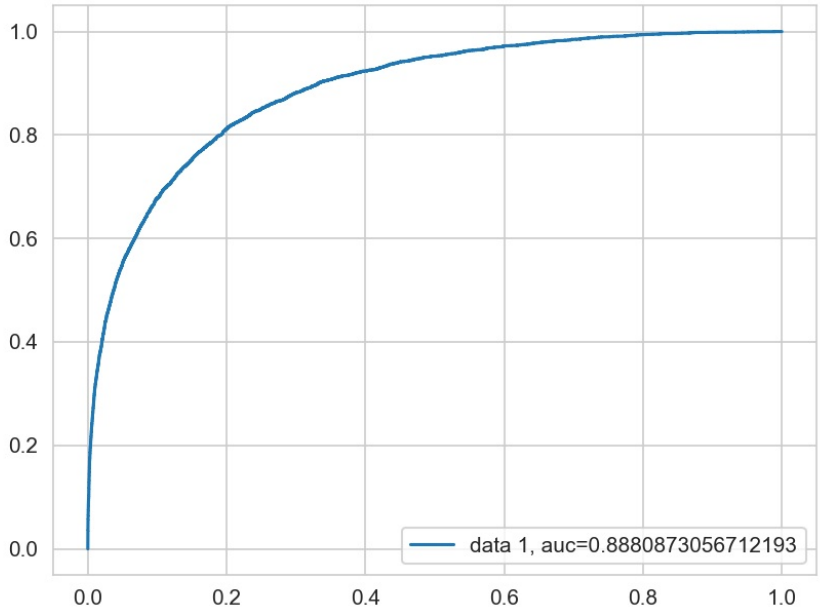
```
y_pred_nb = clf.predict(X_val)
```

In [10]:

```
y_pred_proba = clf.predict_proba(X_val)[::,1]
fpr, tpr, _ = roc_curve(y_val,  y_pred_proba)
auc_gnb = roc_auc_score(y_val, y_pred_proba)
```

In [12]:

```
plt.plot(fpr,tpr,label="data 1, auc="+str(auc_gnb))
plt.legend(loc=4)
plt.show()
```



With Naive-Bayes, the AUC score has slightly improved to 0.880873. Going forward, will try to train with more advanced algorithms to see if it makes any impact on the AUC score.

XGboost

In [59]:

```
xgb_prediction = []
```

In [60]:

```
features = [c for c in train.columns if (c not in ['ID_code', 'target'])]

test_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(train[features], train['target'], test_size = test_size, rand
om_state=42)

d_train = xgb.DMatrix(X_train, y_train)

d_test = xgb.DMatrix(X_test, y_test)

xgb_params = {'max_depth': 8,'objective': 'binary:logistic','eval_metric':'auc'}

watchlist = [(d_train, 'train'), (d_test, 'valid')]

model = xgb.train(xgb_params, d_train,500,watchlist,early_stopping_rounds=20) #500, #150

xgb_pred = model.predict(d_test)

xgb_prediction.append(list(xgb_pred))
```

```
C:\Users\dbhat5\Anaconda3\lib\site-packages\xgboost\core.py:525: FutureWarning: Pass `evals` as keyw
ord args.  Passing these as positional arguments will be considered as error in future releases.
  warnings.warn(
```

```
[0]     train-auc:0.68107       valid-auc:0.65692
[1]     train-auc:0.73782       valid-auc:0.69175
[2]     train-auc:0.76632       valid-auc:0.71062
[3]     train-auc:0.78809       valid-auc:0.72368
[4]     train-auc:0.80970       valid-auc:0.73196
[5]     train-auc:0.82621       valid-auc:0.74119
[6]     train-auc:0.84092       valid-auc:0.74745
[7]     train-auc:0.85692       valid-auc:0.75512
[8]     train-auc:0.87143       valid-auc:0.75916
[9]     train-auc:0.88423       valid-auc:0.76841
[10]    train-auc:0.89532       valid-auc:0.77352
[11]    train-auc:0.90543       valid-auc:0.77794
[12]    train-auc:0.91325       valid-auc:0.78094
[13]    train-auc:0.92141       valid-auc:0.78586
[14]    train-auc:0.92750       valid-auc:0.78950
[15]    train-auc:0.93351       valid-auc:0.79232
[16]    train-auc:0.94032       valid-auc:0.79527
[17]    train-auc:0.94517       valid-auc:0.79873
[18]    train-auc:0.94891       valid-auc:0.80174
[19]    train-auc:0.95342       valid-auc:0.80461
[20]    train-auc:0.95698       valid-auc:0.80759
[21]    train-auc:0.96155       valid-auc:0.80884
[22]    train-auc:0.96476       valid-auc:0.81099
[23]    train-auc:0.96727       valid-auc:0.81231
[24]    train-auc:0.96932       valid-auc:0.81432
[25]    train-auc:0.97150       valid-auc:0.81619
[26]    train-auc:0.97325       valid-auc:0.81731
[27]    train-auc:0.97516       valid-auc:0.81901
[28]    train-auc:0.97702       valid-auc:0.81991
[29]    train-auc:0.97842       valid-auc:0.82126
[30]    train-auc:0.98038       valid-auc:0.82248
[31]    train-auc:0.98224       valid-auc:0.82401
[32]    train-auc:0.98382       valid-auc:0.82531
[33]    train-auc:0.98512       valid-auc:0.82648
[34]    train-auc:0.98621       valid-auc:0.82729
[35]    train-auc:0.98771       valid-auc:0.82846
[36]    train-auc:0.98899       valid-auc:0.82944
[37]    train-auc:0.99000       valid-auc:0.83044
[38]    train-auc:0.99071       valid-auc:0.83077
[39]    train-auc:0.99126       valid-auc:0.83154
[40]    train-auc:0.99203       valid-auc:0.83281
[41]    train-auc:0.99270       valid-auc:0.83362
[42]    train-auc:0.99330       valid-auc:0.83404
[43]    train-auc:0.99372       valid-auc:0.83453
[44]    train-auc:0.99413       valid-auc:0.83571
[45]    train-auc:0.99452       valid-auc:0.83590
[46]    train-auc:0.99486       valid-auc:0.83643
[47]    train-auc:0.99527       valid-auc:0.83695
[48]    train-auc:0.99569       valid-auc:0.83761
[49]    train-auc:0.99597       valid-auc:0.83822
[50]    train-auc:0.99626       valid-auc:0.83854
[51]    train-auc:0.99649       valid-auc:0.83855
[52]    train-auc:0.99682       valid-auc:0.83921
[53]    train-auc:0.99700       valid-auc:0.83932
[54]    train-auc:0.99732       valid-auc:0.83970
[55]    train-auc:0.99752       valid-auc:0.83987
[56]    train-auc:0.99782       valid-auc:0.84002
[57]    train-auc:0.99795       valid-auc:0.84033
[58]    train-auc:0.99809       valid-auc:0.84077
[59]    train-auc:0.99815       valid-auc:0.84123
[60]    train-auc:0.99836       valid-auc:0.84188
[61]    train-auc:0.99852       valid-auc:0.84251
[62]    train-auc:0.99862       valid-auc:0.84337
[63]    train-auc:0.99871       valid-auc:0.84399
[64]    train-auc:0.99883       valid-auc:0.84431
[65]    train-auc:0.99901       valid-auc:0.84453
[66]    train-auc:0.99911       valid-auc:0.84507
[67]    train-auc:0.99917       valid-auc:0.84499
[68]    train-auc:0.99922       valid-auc:0.84529
[69]    train-auc:0.99925       valid-auc:0.84595
[70]    train-auc:0.99929       valid-auc:0.84621
[71]    train-auc:0.99937       valid-auc:0.84685
[72]    train-auc:0.99946       valid-auc:0.84738
[73]    train-auc:0.99951       valid-auc:0.84792
[74]    train-auc:0.99955       valid-auc:0.84811
[75]    train-auc:0.99957       valid-auc:0.84824
[76]    train-auc:0.99960       valid-auc:0.84864
[77]    train-auc:0.99964       valid-auc:0.84899
[78]    train-auc:0.99968       valid-auc:0.84925
[79]    train-auc:0.99973       valid-auc:0.84918
[80]    train-auc:0.99974       valid-auc:0.84934
[81]    train-auc:0.99975       valid-auc:0.84968
[82]    train-auc:0.99976       valid-auc:0.84983
```

```
[83]    train-auc:0.99977       valid-auc:0.84987
[84]    train-auc:0.99979       valid-auc:0.85005
[85]    train-auc:0.99980       valid-auc:0.85043
[86]    train-auc:0.99981       valid-auc:0.85066
[87]    train-auc:0.99983       valid-auc:0.85083
[88]    train-auc:0.99985       valid-auc:0.85122
[89]    train-auc:0.99986       valid-auc:0.85155
[90]    train-auc:0.99987       valid-auc:0.85178
[91]    train-auc:0.99988       valid-auc:0.85187
[92]    train-auc:0.99989       valid-auc:0.85226
[93]    train-auc:0.99990       valid-auc:0.85237
[94]    train-auc:0.99991       valid-auc:0.85273
[95]    train-auc:0.99992       valid-auc:0.85287
[96]    train-auc:0.99993       valid-auc:0.85301
[97]    train-auc:0.99994       valid-auc:0.85336
[98]    train-auc:0.99995       valid-auc:0.85361
[99]    train-auc:0.99995       valid-auc:0.85384
[100]   train-auc:0.99995       valid-auc:0.85407
[101]   train-auc:0.99996       valid-auc:0.85410
[102]   train-auc:0.99996       valid-auc:0.85458
[103]   train-auc:0.99996       valid-auc:0.85515
[104]   train-auc:0.99997       valid-auc:0.85511
[105]   train-auc:0.99998       valid-auc:0.85526
[106]   train-auc:0.99998       valid-auc:0.85515
[107]   train-auc:0.99999       valid-auc:0.85521
[108]   train-auc:0.99999       valid-auc:0.85527
[109]   train-auc:0.99999       valid-auc:0.85541
[110]   train-auc:0.99999       valid-auc:0.85559
[111]   train-auc:0.99999       valid-auc:0.85577
[112]   train-auc:0.99999       valid-auc:0.85562
[113]   train-auc:0.99999       valid-auc:0.85576
[114]   train-auc:0.99999       valid-auc:0.85595
[115]   train-auc:0.99999       valid-auc:0.85603
[116]   train-auc:0.99999       valid-auc:0.85608
[117]   train-auc:0.99999       valid-auc:0.85629
[118]   train-auc:0.99999       valid-auc:0.85658
[119]   train-auc:0.99999       valid-auc:0.85660
[120]   train-auc:0.99999       valid-auc:0.85674
[121]   train-auc:1.00000       valid-auc:0.85684
[122]   train-auc:1.00000       valid-auc:0.85685
[123]   train-auc:1.00000       valid-auc:0.85703
[124]   train-auc:1.00000       valid-auc:0.85730
[125]   train-auc:1.00000       valid-auc:0.85750
[126]   train-auc:1.00000       valid-auc:0.85756
[127]   train-auc:1.00000       valid-auc:0.85785
[128]   train-auc:1.00000       valid-auc:0.85789
[129]   train-auc:1.00000       valid-auc:0.85795
[130]   train-auc:1.00000       valid-auc:0.85782
[131]   train-auc:1.00000       valid-auc:0.85796
[132]   train-auc:1.00000       valid-auc:0.85811
[133]   train-auc:1.00000       valid-auc:0.85808
[134]   train-auc:1.00000       valid-auc:0.85830
[135]   train-auc:1.00000       valid-auc:0.85859
[136]   train-auc:1.00000       valid-auc:0.85863
[137]   train-auc:1.00000       valid-auc:0.85890
[138]   train-auc:1.00000       valid-auc:0.85901
[139]   train-auc:1.00000       valid-auc:0.85898
[140]   train-auc:1.00000       valid-auc:0.85920
[141]   train-auc:1.00000       valid-auc:0.85927
[142]   train-auc:1.00000       valid-auc:0.85930
[143]   train-auc:1.00000       valid-auc:0.85921
[144]   train-auc:1.00000       valid-auc:0.85943
[145]   train-auc:1.00000       valid-auc:0.85952
[146]   train-auc:1.00000       valid-auc:0.85960
[147]   train-auc:1.00000       valid-auc:0.85973
[148]   train-auc:1.00000       valid-auc:0.85979
[149]   train-auc:1.00000       valid-auc:0.85982
[150]   train-auc:1.00000       valid-auc:0.85997
[151]   train-auc:1.00000       valid-auc:0.86018
[152]   train-auc:1.00000       valid-auc:0.86032
[153]   train-auc:1.00000       valid-auc:0.86049
[154]   train-auc:1.00000       valid-auc:0.86071
[155]   train-auc:1.00000       valid-auc:0.86094
[156]   train-auc:1.00000       valid-auc:0.86100
[157]   train-auc:1.00000       valid-auc:0.86101
[158]   train-auc:1.00000       valid-auc:0.86099
[159]   train-auc:1.00000       valid-auc:0.86100
[160]   train-auc:1.00000       valid-auc:0.86120
[161]   train-auc:1.00000       valid-auc:0.86117
[162]   train-auc:1.00000       valid-auc:0.86108
[163]   train-auc:1.00000       valid-auc:0.86118
[164]   train-auc:1.00000       valid-auc:0.86131
[165]   train-auc:1.00000       valid-auc:0.86143
```

```
[166]   train-auc:1.00000       valid-auc:0.86158
[167]   train-auc:1.00000       valid-auc:0.86164
[168]   train-auc:1.00000       valid-auc:0.86186
[169]   train-auc:1.00000       valid-auc:0.86203
[170]   train-auc:1.00000       valid-auc:0.86218
[171]   train-auc:1.00000       valid-auc:0.86234
[172]   train-auc:1.00000       valid-auc:0.86245
[173]   train-auc:1.00000       valid-auc:0.86245
[174]   train-auc:1.00000       valid-auc:0.86244
[175]   train-auc:1.00000       valid-auc:0.86248
[176]   train-auc:1.00000       valid-auc:0.86253
[177]   train-auc:1.00000       valid-auc:0.86262
[178]   train-auc:1.00000       valid-auc:0.86276
[179]   train-auc:1.00000       valid-auc:0.86297
[180]   train-auc:1.00000       valid-auc:0.86306
[181]   train-auc:1.00000       valid-auc:0.86330
[182]   train-auc:1.00000       valid-auc:0.86342
[183]   train-auc:1.00000       valid-auc:0.86364
[184]   train-auc:1.00000       valid-auc:0.86384
[185]   train-auc:1.00000       valid-auc:0.86399
[186]   train-auc:1.00000       valid-auc:0.86419
[187]   train-auc:1.00000       valid-auc:0.86420
[188]   train-auc:1.00000       valid-auc:0.86424
[189]   train-auc:1.00000       valid-auc:0.86447
[190]   train-auc:1.00000       valid-auc:0.86457
[191]   train-auc:1.00000       valid-auc:0.86478
[192]   train-auc:1.00000       valid-auc:0.86490
[193]   train-auc:1.00000       valid-auc:0.86506
[194]   train-auc:1.00000       valid-auc:0.86517
[195]   train-auc:1.00000       valid-auc:0.86517
[196]   train-auc:1.00000       valid-auc:0.86514
[197]   train-auc:1.00000       valid-auc:0.86535
[198]   train-auc:1.00000       valid-auc:0.86551
[199]   train-auc:1.00000       valid-auc:0.86566
[200]   train-auc:1.00000       valid-auc:0.86583
[201]   train-auc:1.00000       valid-auc:0.86604
[202]   train-auc:1.00000       valid-auc:0.86616
[203]   train-auc:1.00000       valid-auc:0.86624
[204]   train-auc:1.00000       valid-auc:0.86628
[205]   train-auc:1.00000       valid-auc:0.86645
[206]   train-auc:1.00000       valid-auc:0.86653
[207]   train-auc:1.00000       valid-auc:0.86662
[208]   train-auc:1.00000       valid-auc:0.86677
[209]   train-auc:1.00000       valid-auc:0.86693
[210]   train-auc:1.00000       valid-auc:0.86716
[211]   train-auc:1.00000       valid-auc:0.86716
[212]   train-auc:1.00000       valid-auc:0.86723
[213]   train-auc:1.00000       valid-auc:0.86728
[214]   train-auc:1.00000       valid-auc:0.86733
[215]   train-auc:1.00000       valid-auc:0.86737
[216]   train-auc:1.00000       valid-auc:0.86737
[217]   train-auc:1.00000       valid-auc:0.86750
[218]   train-auc:1.00000       valid-auc:0.86767
[219]   train-auc:1.00000       valid-auc:0.86793
[220]   train-auc:1.00000       valid-auc:0.86813
[221]   train-auc:1.00000       valid-auc:0.86819
[222]   train-auc:1.00000       valid-auc:0.86830
[223]   train-auc:1.00000       valid-auc:0.86844
[224]   train-auc:1.00000       valid-auc:0.86844
[225]   train-auc:1.00000       valid-auc:0.86857
[226]   train-auc:1.00000       valid-auc:0.86871
[227]   train-auc:1.00000       valid-auc:0.86875
[228]   train-auc:1.00000       valid-auc:0.86886
[229]   train-auc:1.00000       valid-auc:0.86889
[230]   train-auc:1.00000       valid-auc:0.86893
[231]   train-auc:1.00000       valid-auc:0.86899
[232]   train-auc:1.00000       valid-auc:0.86898
[233]   train-auc:1.00000       valid-auc:0.86913
[234]   train-auc:1.00000       valid-auc:0.86926
[235]   train-auc:1.00000       valid-auc:0.86935
[236]   train-auc:1.00000       valid-auc:0.86944
[237]   train-auc:1.00000       valid-auc:0.86952
[238]   train-auc:1.00000       valid-auc:0.86959
[239]   train-auc:1.00000       valid-auc:0.86976
[240]   train-auc:1.00000       valid-auc:0.86979
[241]   train-auc:1.00000       valid-auc:0.86985
[242]   train-auc:1.00000       valid-auc:0.86999
[243]   train-auc:1.00000       valid-auc:0.87008
[244]   train-auc:1.00000       valid-auc:0.87027
[245]   train-auc:1.00000       valid-auc:0.87049
[246]   train-auc:1.00000       valid-auc:0.87061
[247]   train-auc:1.00000       valid-auc:0.87069
[248]   train-auc:1.00000       valid-auc:0.87072
```

```
[249]   train-auc:1.00000       valid-auc:0.87084
[250]   train-auc:1.00000       valid-auc:0.87086
[251]   train-auc:1.00000       valid-auc:0.87088
[252]   train-auc:1.00000       valid-auc:0.87091
[253]   train-auc:1.00000       valid-auc:0.87099
[254]   train-auc:1.00000       valid-auc:0.87106
[255]   train-auc:1.00000       valid-auc:0.87118
[256]   train-auc:1.00000       valid-auc:0.87137
[257]   train-auc:1.00000       valid-auc:0.87133
[258]   train-auc:1.00000       valid-auc:0.87141
[259]   train-auc:1.00000       valid-auc:0.87155
[260]   train-auc:1.00000       valid-auc:0.87160
[261]   train-auc:1.00000       valid-auc:0.87172
[262]   train-auc:1.00000       valid-auc:0.87177
[263]   train-auc:1.00000       valid-auc:0.87192
[264]   train-auc:1.00000       valid-auc:0.87213
[265]   train-auc:1.00000       valid-auc:0.87226
[266]   train-auc:1.00000       valid-auc:0.87234
[267]   train-auc:1.00000       valid-auc:0.87251
[268]   train-auc:1.00000       valid-auc:0.87253
[269]   train-auc:1.00000       valid-auc:0.87263
[270]   train-auc:1.00000       valid-auc:0.87269
[271]   train-auc:1.00000       valid-auc:0.87273
[272]   train-auc:1.00000       valid-auc:0.87274
[273]   train-auc:1.00000       valid-auc:0.87279
[274]   train-auc:1.00000       valid-auc:0.87282
[275]   train-auc:1.00000       valid-auc:0.87294
[276]   train-auc:1.00000       valid-auc:0.87295
[277]   train-auc:1.00000       valid-auc:0.87312
[278]   train-auc:1.00000       valid-auc:0.87323
[279]   train-auc:1.00000       valid-auc:0.87338
[280]   train-auc:1.00000       valid-auc:0.87344
[281]   train-auc:1.00000       valid-auc:0.87353
[282]   train-auc:1.00000       valid-auc:0.87362
[283]   train-auc:1.00000       valid-auc:0.87370
[284]   train-auc:1.00000       valid-auc:0.87380
[285]   train-auc:1.00000       valid-auc:0.87386
[286]   train-auc:1.00000       valid-auc:0.87393
[287]   train-auc:1.00000       valid-auc:0.87402
[288]   train-auc:1.00000       valid-auc:0.87398
[289]   train-auc:1.00000       valid-auc:0.87402
[290]   train-auc:1.00000       valid-auc:0.87395
[291]   train-auc:1.00000       valid-auc:0.87388
[292]   train-auc:1.00000       valid-auc:0.87405
[293]   train-auc:1.00000       valid-auc:0.87415
[294]   train-auc:1.00000       valid-auc:0.87426
[295]   train-auc:1.00000       valid-auc:0.87432
[296]   train-auc:1.00000       valid-auc:0.87440
[297]   train-auc:1.00000       valid-auc:0.87451
[298]   train-auc:1.00000       valid-auc:0.87451
[299]   train-auc:1.00000       valid-auc:0.87456
[300]   train-auc:1.00000       valid-auc:0.87467
[301]   train-auc:1.00000       valid-auc:0.87471
[302]   train-auc:1.00000       valid-auc:0.87477
[303]   train-auc:1.00000       valid-auc:0.87479
[304]   train-auc:1.00000       valid-auc:0.87486
[305]   train-auc:1.00000       valid-auc:0.87486
[306]   train-auc:1.00000       valid-auc:0.87487
[307]   train-auc:1.00000       valid-auc:0.87496
[308]   train-auc:1.00000       valid-auc:0.87509
[309]   train-auc:1.00000       valid-auc:0.87521
[310]   train-auc:1.00000       valid-auc:0.87528
[311]   train-auc:1.00000       valid-auc:0.87537
[312]   train-auc:1.00000       valid-auc:0.87547
[313]   train-auc:1.00000       valid-auc:0.87565
[314]   train-auc:1.00000       valid-auc:0.87575
[315]   train-auc:1.00000       valid-auc:0.87584
[316]   train-auc:1.00000       valid-auc:0.87596
[317]   train-auc:1.00000       valid-auc:0.87604
[318]   train-auc:1.00000       valid-auc:0.87610
[319]   train-auc:1.00000       valid-auc:0.87615
[320]   train-auc:1.00000       valid-auc:0.87620
[321]   train-auc:1.00000       valid-auc:0.87630
[322]   train-auc:1.00000       valid-auc:0.87640
[323]   train-auc:1.00000       valid-auc:0.87640
[324]   train-auc:1.00000       valid-auc:0.87639
[325]   train-auc:1.00000       valid-auc:0.87646
[326]   train-auc:1.00000       valid-auc:0.87657
[327]   train-auc:1.00000       valid-auc:0.87666
[328]   train-auc:1.00000       valid-auc:0.87672
[329]   train-auc:1.00000       valid-auc:0.87673
[330]   train-auc:1.00000       valid-auc:0.87681
[331]   train-auc:1.00000       valid-auc:0.87691
```

```
[332]   train-auc:1.00000       valid-auc:0.87699
[333]   train-auc:1.00000       valid-auc:0.87708
[334]   train-auc:1.00000       valid-auc:0.87709
[335]   train-auc:1.00000       valid-auc:0.87720
[336]   train-auc:1.00000       valid-auc:0.87727
[337]   train-auc:1.00000       valid-auc:0.87732
[338]   train-auc:1.00000       valid-auc:0.87741
[339]   train-auc:1.00000       valid-auc:0.87745
[340]   train-auc:1.00000       valid-auc:0.87743
[341]   train-auc:1.00000       valid-auc:0.87748
[342]   train-auc:1.00000       valid-auc:0.87750
[343]   train-auc:1.00000       valid-auc:0.87760
[344]   train-auc:1.00000       valid-auc:0.87768
[345]   train-auc:1.00000       valid-auc:0.87777
[346]   train-auc:1.00000       valid-auc:0.87780
[347]   train-auc:1.00000       valid-auc:0.87782
[348]   train-auc:1.00000       valid-auc:0.87786
[349]   train-auc:1.00000       valid-auc:0.87791
[350]   train-auc:1.00000       valid-auc:0.87800
[351]   train-auc:1.00000       valid-auc:0.87802
[352]   train-auc:1.00000       valid-auc:0.87803
[353]   train-auc:1.00000       valid-auc:0.87810
[354]   train-auc:1.00000       valid-auc:0.87817
[355]   train-auc:1.00000       valid-auc:0.87822
[356]   train-auc:1.00000       valid-auc:0.87829
[357]   train-auc:1.00000       valid-auc:0.87834
[358]   train-auc:1.00000       valid-auc:0.87831
[359]   train-auc:1.00000       valid-auc:0.87835
[360]   train-auc:1.00000       valid-auc:0.87839
[361]   train-auc:1.00000       valid-auc:0.87843
[362]   train-auc:1.00000       valid-auc:0.87847
[363]   train-auc:1.00000       valid-auc:0.87847
[364]   train-auc:1.00000       valid-auc:0.87848
[365]   train-auc:1.00000       valid-auc:0.87855
[366]   train-auc:1.00000       valid-auc:0.87863
[367]   train-auc:1.00000       valid-auc:0.87863
[368]   train-auc:1.00000       valid-auc:0.87861
[369]   train-auc:1.00000       valid-auc:0.87867
[370]   train-auc:1.00000       valid-auc:0.87870
[371]   train-auc:1.00000       valid-auc:0.87873
[372]   train-auc:1.00000       valid-auc:0.87877
[373]   train-auc:1.00000       valid-auc:0.87880
[374]   train-auc:1.00000       valid-auc:0.87887
[375]   train-auc:1.00000       valid-auc:0.87894
[376]   train-auc:1.00000       valid-auc:0.87897
[377]   train-auc:1.00000       valid-auc:0.87906
[378]   train-auc:1.00000       valid-auc:0.87912
[379]   train-auc:1.00000       valid-auc:0.87916
[380]   train-auc:1.00000       valid-auc:0.87926
[381]   train-auc:1.00000       valid-auc:0.87934
[382]   train-auc:1.00000       valid-auc:0.87934
[383]   train-auc:1.00000       valid-auc:0.87939
[384]   train-auc:1.00000       valid-auc:0.87937
[385]   train-auc:1.00000       valid-auc:0.87945
[386]   train-auc:1.00000       valid-auc:0.87950
[387]   train-auc:1.00000       valid-auc:0.87957
[388]   train-auc:1.00000       valid-auc:0.87963
[389]   train-auc:1.00000       valid-auc:0.87971
[390]   train-auc:1.00000       valid-auc:0.87983
[391]   train-auc:1.00000       valid-auc:0.87989
[392]   train-auc:1.00000       valid-auc:0.87993
[393]   train-auc:1.00000       valid-auc:0.87997
[394]   train-auc:1.00000       valid-auc:0.88003
[395]   train-auc:1.00000       valid-auc:0.88013
[396]   train-auc:1.00000       valid-auc:0.88018
[397]   train-auc:1.00000       valid-auc:0.88025
[398]   train-auc:1.00000       valid-auc:0.88033
[399]   train-auc:1.00000       valid-auc:0.88031
[400]   train-auc:1.00000       valid-auc:0.88032
[401]   train-auc:1.00000       valid-auc:0.88033
[402]   train-auc:1.00000       valid-auc:0.88036
[403]   train-auc:1.00000       valid-auc:0.88036
[404]   train-auc:1.00000       valid-auc:0.88043
[405]   train-auc:1.00000       valid-auc:0.88050
[406]   train-auc:1.00000       valid-auc:0.88054
[407]   train-auc:1.00000       valid-auc:0.88061
[408]   train-auc:1.00000       valid-auc:0.88062
[409]   train-auc:1.00000       valid-auc:0.88069
[410]   train-auc:1.00000       valid-auc:0.88069
[411]   train-auc:1.00000       valid-auc:0.88073
[412]   train-auc:1.00000       valid-auc:0.88072
[413]   train-auc:1.00000       valid-auc:0.88079
[414]   train-auc:1.00000       valid-auc:0.88086
```

```
[415]   train-auc:1.00000       valid-auc:0.88094
[416]   train-auc:1.00000       valid-auc:0.88094
[417]   train-auc:1.00000       valid-auc:0.88100
[418]   train-auc:1.00000       valid-auc:0.88107
[419]   train-auc:1.00000       valid-auc:0.88108
[420]   train-auc:1.00000       valid-auc:0.88112
[421]   train-auc:1.00000       valid-auc:0.88112
[422]   train-auc:1.00000       valid-auc:0.88113
[423]   train-auc:1.00000       valid-auc:0.88120
[424]   train-auc:1.00000       valid-auc:0.88124
[425]   train-auc:1.00000       valid-auc:0.88128
[426]   train-auc:1.00000       valid-auc:0.88131
[427]   train-auc:1.00000       valid-auc:0.88131
[428]   train-auc:1.00000       valid-auc:0.88137
[429]   train-auc:1.00000       valid-auc:0.88142
[430]   train-auc:1.00000       valid-auc:0.88150
[431]   train-auc:1.00000       valid-auc:0.88154
[432]   train-auc:1.00000       valid-auc:0.88155
[433]   train-auc:1.00000       valid-auc:0.88160
[434]   train-auc:1.00000       valid-auc:0.88164
[435]   train-auc:1.00000       valid-auc:0.88162
[436]   train-auc:1.00000       valid-auc:0.88169
[437]   train-auc:1.00000       valid-auc:0.88175
[438]   train-auc:1.00000       valid-auc:0.88174
[439]   train-auc:1.00000       valid-auc:0.88179
[440]   train-auc:1.00000       valid-auc:0.88187
[441]   train-auc:1.00000       valid-auc:0.88186
[442]   train-auc:1.00000       valid-auc:0.88192
[443]   train-auc:1.00000       valid-auc:0.88197
[444]   train-auc:1.00000       valid-auc:0.88205
[445]   train-auc:1.00000       valid-auc:0.88209
[446]   train-auc:1.00000       valid-auc:0.88217
[447]   train-auc:1.00000       valid-auc:0.88222
[448]   train-auc:1.00000       valid-auc:0.88227
[449]   train-auc:1.00000       valid-auc:0.88229
[450]   train-auc:1.00000       valid-auc:0.88234
[451]   train-auc:1.00000       valid-auc:0.88230
[452]   train-auc:1.00000       valid-auc:0.88231
[453]   train-auc:1.00000       valid-auc:0.88234
[454]   train-auc:1.00000       valid-auc:0.88235
[455]   train-auc:1.00000       valid-auc:0.88237
[456]   train-auc:1.00000       valid-auc:0.88240
[457]   train-auc:1.00000       valid-auc:0.88250
[458]   train-auc:1.00000       valid-auc:0.88255
[459]   train-auc:1.00000       valid-auc:0.88260
[460]   train-auc:1.00000       valid-auc:0.88266
[461]   train-auc:1.00000       valid-auc:0.88265
[462]   train-auc:1.00000       valid-auc:0.88275
[463]   train-auc:1.00000       valid-auc:0.88274
[464]   train-auc:1.00000       valid-auc:0.88276
[465]   train-auc:1.00000       valid-auc:0.88276
[466]   train-auc:1.00000       valid-auc:0.88282
[467]   train-auc:1.00000       valid-auc:0.88294
[468]   train-auc:1.00000       valid-auc:0.88297
[469]   train-auc:1.00000       valid-auc:0.88300
[470]   train-auc:1.00000       valid-auc:0.88300
[471]   train-auc:1.00000       valid-auc:0.88303
[472]   train-auc:1.00000       valid-auc:0.88305
[473]   train-auc:1.00000       valid-auc:0.88316
[474]   train-auc:1.00000       valid-auc:0.88318
[475]   train-auc:1.00000       valid-auc:0.88311
[476]   train-auc:1.00000       valid-auc:0.88317
[477]   train-auc:1.00000       valid-auc:0.88316
[478]   train-auc:1.00000       valid-auc:0.88320
[479]   train-auc:1.00000       valid-auc:0.88326
[480]   train-auc:1.00000       valid-auc:0.88330
[481]   train-auc:1.00000       valid-auc:0.88333
[482]   train-auc:1.00000       valid-auc:0.88329
[483]   train-auc:1.00000       valid-auc:0.88330
[484]   train-auc:1.00000       valid-auc:0.88337
[485]   train-auc:1.00000       valid-auc:0.88340
[486]   train-auc:1.00000       valid-auc:0.88351
[487]   train-auc:1.00000       valid-auc:0.88357
[488]   train-auc:1.00000       valid-auc:0.88361
[489]   train-auc:1.00000       valid-auc:0.88365
[490]   train-auc:1.00000       valid-auc:0.88364
[491]   train-auc:1.00000       valid-auc:0.88371
[492]   train-auc:1.00000       valid-auc:0.88375
[493]   train-auc:1.00000       valid-auc:0.88373
[494]   train-auc:1.00000       valid-auc:0.88378
[495]   train-auc:1.00000       valid-auc:0.88385
[496]   train-auc:1.00000       valid-auc:0.88393
[497]   train-auc:1.00000       valid-auc:0.88394
```

```
[498]   train-auc:1.00000       valid-auc:0.88396
[499]   train-auc:1.00000       valid-auc:0.88400
```

With XGboost the AUC score has not improved much. It is somewhat similar to what was achieved with Naive Bayes.

LightGBM

In [19]:

```
param = {
    "objective" : "binary",
    "metric" : "auc",
    "boosting": 'gbdt',
    "max_depth" : -1,
    "num_leaves" : 31,
    "learning_rate" : 0.01,
    "bagging_freq": 5,
    "bagging_fraction" : 0.4,
    "feature_fraction" : 1,
    "min_data_in_leaf": 150,
    "tree_learner": "serial",
    "boost_from_average": "false",
    "bagging_seed" : random_state,
    "verbosity" : 1,
    "seed": random_state}
```

In [20]:

```
features = [c for c in train.columns if (c not in ['ID_code', 'target'])]

test_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(train[features], train['target'], test_size = test_size, rand
om_state=42)

iterations = 110
y_hat = np.zeros([int(200000*test_size), 200])
i=0
for feature in ['var_' + str(x) for x in range(200)]: # loop over all features
    #print(feature)
    feat_choices = [feature, 'new_' + feature]
    lgb_train = lgb.Dataset(X_train[feat_choices], y_train)
    gbm = lgb.train(param, lgb_train, iterations, verbose_eval=-1)
    y_hat[:, i] = gbm.predict(X_test[feat_choices], num_iteration=gbm.best_iteration)
    #test_hat[:, i] = gbm.predict(test_df[feat_choices], num_iteration=gbm.best_iteration)
    i += 1

with open('lgb_model_pkl', 'wb') as files:
    pickle.dump(gbm, files)
sub_preds = (y_hat).sum(axis=1)
score = roc_auc_score(y_test, sub_preds)
print('Your CV score is', score)
```

```
C:\Users\dbhat5\Anaconda3\lib\site-packages\lightgbm\engine.py:239: UserWarning: 'verbose_eval' argu
ment is deprecated and will be removed in a future release of LightGBM. Pass 'log_evaluation()' call
back via 'callbacks' argument instead.
  _log_warning("'verbose_eval' argument is deprecated and will be removed in a future release of Lig
htGBM. "

[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001272 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000788 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000483 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 272
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.001733 se
```

```
conds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 273
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000378 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 275
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000590 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000502 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 286
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000629 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001549 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000430 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 278
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000564 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000627 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000473 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 354
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000598 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000391 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 272
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000508 se
```

```
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 302
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000580 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 271
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000526 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000667 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000488 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000470 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000480 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000459 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000456 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 295
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000461 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000363 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 315
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000551 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
```

```
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000370 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 276
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000369 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 287
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000460 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000485 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000372 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 276
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000441 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 272
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000475 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000478 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 294
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000474 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000466 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000392 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 273
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000552 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
```

```
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000763 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000687 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000494 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000367 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 287
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000829 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 310
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000470 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000524 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 263
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000557 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000495 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000562 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000688 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.001147 se
conds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 289
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
```

```
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000504 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000561 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000472 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 288
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000583 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000582 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000612 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 271
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000624 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 285
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000710 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000782 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 284
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000471 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000512 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 263
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.001920 se
conds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 274
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
```

```
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000468 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000409 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 277
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.002239 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000368 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 279
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000579 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000434 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 481
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000692 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000649 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000513 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 318
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000481 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000575 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000616 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 264
```

```
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000687 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000485 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000587 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000611 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 274
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000509 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 277
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000597 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000601 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 274
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000687 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000616 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000726 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000486 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000502 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
```

```
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000474 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000382 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 272
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000634 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000530 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000374 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 350
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000478 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000551 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 298
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000785 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 272
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000378 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 291
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000484 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 264
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000508 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 264
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000365 se
conds.
You can set `force_row_wise=true` to remove the overhead.
```

```
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 290
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000474 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 275
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000502 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 264
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000479 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000490 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000364 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 334
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000495 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 274
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000454 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 286
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000488 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 274
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000503 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000368 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 368
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000466 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000481 se
conds.
```

```
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000512 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 281
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000574 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 276
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001028 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000370 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 281
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000540 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 273
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000401 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 275
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000520 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 264
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000563 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000554 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000502 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 264
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000537 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 275
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000668 se
```

```
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000988 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000764 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000495 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 311
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000477 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 309
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000487 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000676 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000487 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000520 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 286
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000403 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 298
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.002479 se
conds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 276
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000363 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 309
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000477 se
```

```
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000675 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.002272 se
conds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 263
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000677 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000588 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000639 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000795 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000477 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000586 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000512 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 272
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000405 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 285
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000624 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000499 se
```

```
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 275
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000742 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.002437 se
conds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 337
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.002763 se
conds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000392 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 273
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000553 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000675 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000498 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 272
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000477 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000557 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000392 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 282
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000676 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000486 se
conds.
```

```
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 264
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000482 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000714 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 264
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000371 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 328
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000568 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 277
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000479 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000836 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000708 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000545 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 306
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000736 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.005281 se
conds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000600 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 309
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000739 se
conds.
```

```
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 269
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000501 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000642 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000487 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000484 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000385 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 273
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000623 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000395 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 272
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000490 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.001752 se
conds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 272
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000507 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000395 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 286
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000523 se
conds.
```

```
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 265
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000555 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000652 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000573 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000516 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 271
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000611 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 263
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000672 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 268
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000415 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 282
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000535 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000650 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000415 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 279
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000964 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 267
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000722 se
```

```
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000723 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 277
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001166 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 266
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000453 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 284
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.002169 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 270
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
[LightGBM] [Info] Number of positive: 13954, number of negative: 126046
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000808 se
conds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 264
[LightGBM] [Info] Number of data points in the train set: 140000, number of used features: 2
Your CV score is 0.9106051013785603
```

With LightGBM model a CV score of 0.9106051013 is accomplished.

Summary

In [6]:

```
from prettytable import PrettyTable
tb = PrettyTable()
tb.field_names= ("Model", "AUC")
tb.add_row(["Logistic Regression", 0.85742])
tb.add_row(["Naive Bayes", 0.8808])
tb.add_row(["XGboost", 0.88400])
tb.add_row(["LightGBM", 0.9106051013])
print(tb.get_string(titles = "Observations"))
```

```
+---------------------+--------------+
|        Model        |     AUC      |
+---------------------+--------------+
| Logistic Regression |   0.85742    |
|     Naive Bayes     |    0.8808    |
|       XGboost       |    0.884     |
|       LightGBM      | 0.9106051013 |
+---------------------+--------------+
```

In [ ]: