LightBoost

In [8]:
```
import lightgbm as lgb
import pandas as pd
import numpy as np
import os
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc, accuracy_score,precisio
n_recall_curve,sensitivity
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import recall_score, make_scorer
from sklearn.preprocessing import StandardScaler
from matplotlib import pyplot as plt
import statsmodels.api as sm
import pylab as pl
import yellowbrick
from yellowbrick.classifier import PrecisionRecallCurve
```

read data ¶

In [9]:
```
#os.chdir=('/Users/zhoujingyu/Desktop/ca/lgbt/')
```

In [11]:
```
test=pd.read_csv('test_v3.csv')
#train_rose=pd.read_csv('train_rose.csv')
# train_both=pd.read_csv('train_both.csv',index_col=0)
train_over=pd.read_csv('train_over_v3.csv')
# train_under=pd.read_csv('train_under.csv',index_col=0)
```

In [13]:
```
train_over.hospital_death.describe()
```

Out[13]:
```
count    74212.000000
mean         0.500000
std          0.500003
min          0.000000
25%          0.000000
50%          0.500000
75%          1.000000
max          1.000000
Name: hospital_death, dtype: float64
```

In [15]:
```
train_rose=train_over.drop(['Unnamed: 0'],axis=1)
test=test.drop(['Unnamed: 0'],axis=1)
```

In [16]:
```
print('train_num=',len(train_rose),'test_unm=',len(test))
```

```
train_num= 74212 test_unm= 26242
```

In [17]:
```
#merge data
def get_id(df):
    list_num=[]
    for i in range(len(df)):
        list_num.append(i)
    df['id']=list_num
    return df
def get_test_id(df):
    list_num=[]
    for i in range(len(train_rose),(len(df)+len(train_rose))):
        list_num.append(i)
    df['id']=list_num
    return df
```

In [18]:
```
#get data id
train_rose=get_id(train_rose)
test=get_test_id(test)
train_rose=train_rose.set_index('id')
test=test.set_index('id')
```

In [19]:
```
#concat data
whole_data=pd.concat([train_rose,test],axis=0)
```

In [20]:
```
pd.DataFrame(whole_data.hospital_death).to_csv('death.csv')
```

#transfer to dummy variables

In [21]:
```
#transfer to dummy variables
def get_dum(df):
    df_obj=df.select_dtypes(include=['object'])
    df_no_obj=df.select_dtypes(exclude=['object'])
    obj_dummy=pd.get_dummies(df_obj)
    summary=pd.concat([df_no_obj,obj_dummy],axis=1,join='inner')
    return summary

whole_data=get_dum(whole_data)
```

In [22]:
```
train=whole_data.loc[whole_data.index<len(train_rose)]
test=whole_data.loc[whole_data.index>=len(train_rose)]
```

In [23]:
```
train.hospital_death.describe()
```

Out[23]:
```
count    74212.000000
mean         0.500000
std          0.500003
min          0.000000
25%          0.000000
50%          0.500000
75%          1.000000
max          1.000000
Name: hospital_death, dtype: float64
```

split train and test

In [24]:
```
#split train and test
train=whole_data.loc[whole_data.index<len(train_rose)]
test=whole_data.loc[whole_data.index>=len(train_rose)]

train_x=train.drop(['hospital_death'],axis=1)
train_y=train[['hospital_death']]
test_x=test.drop(['hospital_death'],axis=1)
test_y=test[['hospital_death']]
```

use grid Search to find best estimator

In [1]:
```
#use grid Search to find best estimator
estimator = lgb.LGBMClassifier(objective='binary',num_leaves=30,lea
rning_rate=0.5,n_estimators=60,feature_fraction=0.7,
                               bagging_fraction=0.9,max_bin=255,bagg
ing_freq=5,verbose=0)

param_grid = {
    'learning_rate': [0.05,0.1,0.5],
    #0.5 is best
#     'n_estimators': [20,40,60],
    # 60 is best
#     'num_leaves':[30,31,32],
    #30 is best
#     'bagging_fraction':[0.7,0.8,0.9],
    #0.9 is best
#     'max_bin':[250,255,260],
    #255 is best
#     'feature_fraction':[0.7,0.8,0.9],
    #0.7 is best
    #'bagging_freq':[5,6,7],
    #5 is best

}

scoring=make_scorer(roc_auc_score)
gbm0 = GridSearchCV(estimator, param_grid,scoring,cv=9)

gbm0.fit(train_x, train_y)

print('Best parameters found by grid search are:', gbm0.best_params
_)
```

build model

In [26]:
```
gbm2 = lgb.LGBMClassifier(objective='binary',num_leaves=40,learning
_rate=0.05,feature_fraction=0.7,
                          n_estimators=60,max_bin=260)
gbm2.fit(train_x, train_y,eval_set=[(test_x, test_y)],eval_metric='
AUC',early_stopping_rounds=30)
```

```
[1]     valid_0's auc: 0.826191 valid_0's binary_logloss: 0.67408
Training until validation scores don't improve for 30 rounds
[2]     valid_0's auc: 0.83756  valid_0's binary_logloss: 0.657952
[3]     valid_0's auc: 0.843534 valid_0's binary_logloss: 0.642851
[4]     valid_0's auc: 0.84451  valid_0's binary_logloss: 0.629849
[5]     valid_0's auc: 0.850642 valid_0's binary_logloss: 0.61731
[6]     valid_0's auc: 0.852857 valid_0's binary_logloss: 0.605966
[7]     valid_0's auc: 0.854479 valid_0's binary_logloss: 0.596316
[8]     valid_0's auc: 0.855458 valid_0's binary_logloss: 0.587008
[9]     valid_0's auc: 0.856152 valid_0's binary_logloss: 0.577026
```

```
[10]    valid_0's auc: 0.857865 valid_0's binary_logloss: 0.568978
[11]    valid_0's auc: 0.85858  valid_0's binary_logloss: 0.561104
[12]    valid_0's auc: 0.859728 valid_0's binary_logloss: 0.552667
[13]    valid_0's auc: 0.860887 valid_0's binary_logloss: 0.544874
[14]    valid_0's auc: 0.861267 valid_0's binary_logloss: 0.538154
[15]    valid_0's auc: 0.862786 valid_0's binary_logloss: 0.530824
[16]    valid_0's auc: 0.863268 valid_0's binary_logloss: 0.526048
[17]    valid_0's auc: 0.863883 valid_0's binary_logloss: 0.521379
[18]    valid_0's auc: 0.863989 valid_0's binary_logloss: 0.515495
[19]    valid_0's auc: 0.864592 valid_0's binary_logloss: 0.509851
[20]    valid_0's auc: 0.865427 valid_0's binary_logloss: 0.504798
[21]    valid_0's auc: 0.866152 valid_0's binary_logloss: 0.499527
[22]    valid_0's auc: 0.867362 valid_0's binary_logloss: 0.494245
[23]    valid_0's auc: 0.867692 valid_0's binary_logloss: 0.49137
[24]    valid_0's auc: 0.868026 valid_0's binary_logloss: 0.487813
[25]    valid_0's auc: 0.868124 valid_0's binary_logloss: 0.485252
[26]    valid_0's auc: 0.868917 valid_0's binary_logloss: 0.481716
[27]    valid_0's auc: 0.869251 valid_0's binary_logloss: 0.478279
[28]    valid_0's auc: 0.86952  valid_0's binary_logloss: 0.475993
[29]    valid_0's auc: 0.869535 valid_0's binary_logloss: 0.473503
[30]    valid_0's auc: 0.869875 valid_0's binary_logloss: 0.470717
[31]    valid_0's auc: 0.870602 valid_0's binary_logloss: 0.467313
[32]    valid_0's auc: 0.870788 valid_0's binary_logloss: 0.465329
[33]    valid_0's auc: 0.871781 valid_0's binary_logloss: 0.462372
[34]    valid_0's auc: 0.8721   valid_0's binary_logloss: 0.460914
[35]    valid_0's auc: 0.872389 valid_0's binary_logloss: 0.459166
[36]    valid_0's auc: 0.872746 valid_0's binary_logloss: 0.456647
[37]    valid_0's auc: 0.873076 valid_0's binary_logloss: 0.454844
[38]    valid_0's auc: 0.873524 valid_0's binary_logloss: 0.452981
[39]    valid_0's auc: 0.873836 valid_0's binary_logloss: 0.451651
[40]    valid_0's auc: 0.874062 valid_0's binary_logloss: 0.449787
[41]    valid_0's auc: 0.874394 valid_0's binary_logloss: 0.447302
[42]    valid_0's auc: 0.874524 valid_0's binary_logloss: 0.445343
[43]    valid_0's auc: 0.875016 valid_0's binary_logloss: 0.44293
[44]    valid_0's auc: 0.875302 valid_0's binary_logloss: 0.441327
[45]    valid_0's auc: 0.875732 valid_0's binary_logloss: 0.439372
[46]    valid_0's auc: 0.875733 valid_0's binary_logloss: 0.438686
[47]    valid_0's auc: 0.876181 valid_0's binary_logloss: 0.437183
[48]    valid_0's auc: 0.876347 valid_0's binary_logloss: 0.436117
[49]    valid_0's auc: 0.876465 valid_0's binary_logloss: 0.434729
[50]    valid_0's auc: 0.876446 valid_0's binary_logloss: 0.433944
[51]    valid_0's auc: 0.876654 valid_0's binary_logloss: 0.432747
[52]    valid_0's auc: 0.876972 valid_0's binary_logloss: 0.431265
[53]    valid_0's auc: 0.876816 valid_0's binary_logloss: 0.431325
[54]    valid_0's auc: 0.876786 valid_0's binary_logloss: 0.430965
[55]    valid_0's auc: 0.876886 valid_0's binary_logloss: 0.429741
[56]    valid_0's auc: 0.876909 valid_0's binary_logloss: 0.429094
[57]    valid_0's auc: 0.876887 valid_0's binary_logloss: 0.428595
[58]    valid_0's auc: 0.877056 valid_0's binary_logloss: 0.427576
[59]    valid_0's auc: 0.877261 valid_0's binary_logloss: 0.426366
[60]    valid_0's auc: 0.877491 valid_0's binary_logloss: 0.425414
Did not meet early stopping. Best iteration is:
[60]    valid_0's auc: 0.877491 valid_0's binary_logloss: 0.425414
```

Out[26]:
```
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_
bytree=1.0,
               feature_fraction=0.7, importance_type='split',
               learning_rate=0.05, max_bin=260, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_s
plit_gain=0.0,
               n_estimators=60, n_jobs=-1, num_leaves=40, objectiv
e='binary',
               random_state=None, reg_alpha=0.0, reg_lambda=0.0, s
ilent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_
freq=0)
```

In [27]:
```
params = {
'task': 'train',
'prediction':'test',
'boosting_type': 'gbdt',
'objective': 'binary',
'metric': {'l2', 'auc','binary_error'},
'estimator':60,
'num_leaves': 40,
'learning_rate': 0.05,
'feature_fraction': 0.7,
'max_bin':260,
'verbose': 0,
'seed':0,
}
lgb_train = lgb.Dataset(train_x, train_y)
lgb_eval = lgb.Dataset(test_x, test_y)
gbm = lgb.train(params,
                lgb_train,
                num_boost_round=100,
                valid_sets=lgb_eval,
                early_stopping_rounds=500)
```

```
[1]     valid_0's l2: 0.240505  valid_0's binary_error: 0.288507
valid_0's auc: 0.82371
Training until validation scores don't improve for 500 rounds
[2]     valid_0's l2: 0.232399  valid_0's binary_error: 0.243731
valid_0's auc: 0.834225
[3]     valid_0's l2: 0.226123  valid_0's binary_error: 0.226736
valid_0's auc: 0.845529
[4]     valid_0's l2: 0.219523  valid_0's binary_error: 0.225821
valid_0's auc: 0.847773
[5]     valid_0's l2: 0.212995  valid_0's binary_error: 0.225669
valid_0's auc: 0.852237
[6]     valid_0's l2: 0.207064  valid_0's binary_error: 0.22483 va
lid_0's auc: 0.854093
[7]     valid_0's l2: 0.20226   valid_0's binary_error: 0.228451
valid_0's auc: 0.854967
[8]     valid_0's l2: 0.197208  valid_0's binary_error: 0.229632
valid_0's auc: 0.855665
[9]     valid_0's l2: 0.192739  valid_0's binary_error: 0.228565
```

```
valid_0's auc: 0.856221
[10]    valid_0's l2: 0.188563  valid_0's binary_error: 0.226583
valid_0's auc: 0.857446
[11]    valid_0's l2: 0.184716  valid_0's binary_error: 0.223497
valid_0's auc: 0.858736
[12]    valid_0's l2: 0.181344  valid_0's binary_error: 0.221706
valid_0's auc: 0.859692
[13]    valid_0's l2: 0.178666  valid_0's binary_error: 0.219457
valid_0's auc: 0.859888
[14]    valid_0's l2: 0.176189  valid_0's binary_error: 0.217857
valid_0's auc: 0.86033
[15]    valid_0's l2: 0.174082  valid_0's binary_error: 0.217704
valid_0's auc: 0.86125
[16]    valid_0's l2: 0.171829  valid_0's binary_error: 0.217819
valid_0's auc: 0.861668
[17]    valid_0's l2: 0.170059  valid_0's binary_error: 0.21839 va
lid_0's auc: 0.861958
[18]    valid_0's l2: 0.168256  valid_0's binary_error: 0.2182
valid_0's auc: 0.862424
[19]    valid_0's l2: 0.166001  valid_0's binary_error: 0.215456
valid_0's auc: 0.863694
[20]    valid_0's l2: 0.164417  valid_0's binary_error: 0.215037
valid_0's auc: 0.864334
[21]    valid_0's l2: 0.162662  valid_0's binary_error: 0.213398
valid_0's auc: 0.865181
[22]    valid_0's l2: 0.161367  valid_0's binary_error: 0.213856
valid_0's auc: 0.865469
[23]    valid_0's l2: 0.159762  valid_0's binary_error: 0.212331
valid_0's auc: 0.86605
[24]    valid_0's l2: 0.157827  valid_0's binary_error: 0.211531
valid_0's auc: 0.866849
[25]    valid_0's l2: 0.156511  valid_0's binary_error: 0.211417
valid_0's auc: 0.867507
[26]    valid_0's l2: 0.155574  valid_0's binary_error: 0.210197
valid_0's auc: 0.867702
[27]    valid_0's l2: 0.154075  valid_0's binary_error: 0.209245
valid_0's auc: 0.868703
[28]    valid_0's l2: 0.15261   valid_0's binary_error: 0.208559
valid_0's auc: 0.869392
[29]    valid_0's l2: 0.15099   valid_0's binary_error: 0.206577
valid_0's auc: 0.870145
[30]    valid_0's l2: 0.150312  valid_0's binary_error: 0.206463
valid_0's auc: 0.870405
[31]    valid_0's l2: 0.149089  valid_0's binary_error: 0.204443
valid_0's auc: 0.871145
[32]    valid_0's l2: 0.148265  valid_0's binary_error: 0.204024
valid_0's auc: 0.871671
[33]    valid_0's l2: 0.147811  valid_0's binary_error: 0.204901
valid_0's auc: 0.872002
[34]    valid_0's l2: 0.14732   valid_0's binary_error: 0.204977
valid_0's auc: 0.872092
[35]    valid_0's l2: 0.146204  valid_0's binary_error: 0.203567
valid_0's auc: 0.872969
```

```
[36]    valid_0's l2: 0.145354  valid_0's binary_error: 0.203376
valid_0's auc: 0.873496
[37]    valid_0's l2: 0.144527  valid_0's binary_error: 0.2017
valid_0's auc: 0.873946
[38]    valid_0's l2: 0.143731  valid_0's binary_error: 0.201166
valid_0's auc: 0.87431
[39]    valid_0's l2: 0.143364  valid_0's binary_error: 0.2017
valid_0's auc: 0.87472
[40]    valid_0's l2: 0.14263   valid_0's binary_error: 0.201623
valid_0's auc: 0.874971
[41]    valid_0's l2: 0.14245   valid_0's binary_error: 0.202271
valid_0's auc: 0.875109
[42]    valid_0's l2: 0.141916  valid_0's binary_error: 0.20189 va
lid_0's auc: 0.875457
[43]    valid_0's l2: 0.14152   valid_0's binary_error: 0.201623
valid_0's auc: 0.875658
[44]    valid_0's l2: 0.141109  valid_0's binary_error: 0.201547
valid_0's auc: 0.875909
[45]    valid_0's l2: 0.140312  valid_0's binary_error: 0.200594
valid_0's auc: 0.876309
[46]    valid_0's l2: 0.140009  valid_0's binary_error: 0.200404
valid_0's auc: 0.876393
[47]    valid_0's l2: 0.139206  valid_0's binary_error: 0.199718
valid_0's auc: 0.876936
[48]    valid_0's l2: 0.139066  valid_0's binary_error: 0.199832
valid_0's auc: 0.877065
[49]    valid_0's l2: 0.138586  valid_0's binary_error: 0.199527
valid_0's auc: 0.877317
[50]    valid_0's l2: 0.138425  valid_0's binary_error: 0.199185
valid_0's auc: 0.87728
[51]    valid_0's l2: 0.138545  valid_0's binary_error: 0.199451
valid_0's auc: 0.877237
[52]    valid_0's l2: 0.138542  valid_0's binary_error: 0.200328
valid_0's auc: 0.877201
[53]    valid_0's l2: 0.13865   valid_0's binary_error: 0.200671
valid_0's auc: 0.877191
[54]    valid_0's l2: 0.139068  valid_0's binary_error: 0.202843
valid_0's auc: 0.876972
[55]    valid_0's l2: 0.138708  valid_0's binary_error: 0.202157
valid_0's auc: 0.877205
[56]    valid_0's l2: 0.138633  valid_0's binary_error: 0.201928
valid_0's auc: 0.877125
[57]    valid_0's l2: 0.138818  valid_0's binary_error: 0.20269 va
lid_0's auc: 0.877072
[58]    valid_0's l2: 0.138306  valid_0's binary_error: 0.201661
valid_0's auc: 0.877377
[59]    valid_0's l2: 0.138027  valid_0's binary_error: 0.201395
valid_0's auc: 0.877494
[60]    valid_0's l2: 0.13817   valid_0's binary_error: 0.201814
valid_0's auc: 0.877408
[61]    valid_0's l2: 0.13783   valid_0's binary_error: 0.201509
valid_0's auc: 0.877832
[62]    valid_0's l2: 0.137596  valid_0's binary_error: 0.201242
```

```
valid_0's auc: 0.877818
[63]    valid_0's l2: 0.13763   valid_0's binary_error: 0.201471
valid_0's auc: 0.877702
[64]    valid_0's l2: 0.137919  valid_0's binary_error: 0.202805
valid_0's auc: 0.877567
[65]    valid_0's l2: 0.137705  valid_0's binary_error: 0.202157
valid_0's auc: 0.877534
[66]    valid_0's l2: 0.137582  valid_0's binary_error: 0.201966
valid_0's auc: 0.877575
[67]    valid_0's l2: 0.137775  valid_0's binary_error: 0.203605
valid_0's auc: 0.877434
[68]    valid_0's l2: 0.137676  valid_0's binary_error: 0.203224
valid_0's auc: 0.877439
[69]    valid_0's l2: 0.137694  valid_0's binary_error: 0.203834
valid_0's auc: 0.877323
[70]    valid_0's l2: 0.137108  valid_0's binary_error: 0.202614
valid_0's auc: 0.877587
[71]    valid_0's l2: 0.137493  valid_0's binary_error: 0.203948
valid_0's auc: 0.877376
[72]    valid_0's l2: 0.137214  valid_0's binary_error: 0.203605
valid_0's auc: 0.877432
[73]    valid_0's l2: 0.136917  valid_0's binary_error: 0.203033
valid_0's auc: 0.877582
[74]    valid_0's l2: 0.136826  valid_0's binary_error: 0.202347
valid_0's auc: 0.877603
[75]    valid_0's l2: 0.13658   valid_0's binary_error: 0.202309
valid_0's auc: 0.877675
[76]    valid_0's l2: 0.136793  valid_0's binary_error: 0.202424
valid_0's auc: 0.877471
[77]    valid_0's l2: 0.137294  valid_0's binary_error: 0.204024
valid_0's auc: 0.877145
[78]    valid_0's l2: 0.137354  valid_0's binary_error: 0.204215
valid_0's auc: 0.877158
[79]    valid_0's l2: 0.136968  valid_0's binary_error: 0.20311 va
lid_0's auc: 0.877251
[80]    valid_0's l2: 0.136677  valid_0's binary_error: 0.203148
valid_0's auc: 0.877416
[81]    valid_0's l2: 0.136538  valid_0's binary_error: 0.203338
valid_0's auc: 0.87747
[82]    valid_0's l2: 0.136305  valid_0's binary_error: 0.202843
valid_0's auc: 0.877432
[83]    valid_0's l2: 0.136154  valid_0's binary_error: 0.202233
valid_0's auc: 0.877479
[84]    valid_0's l2: 0.136608  valid_0's binary_error: 0.203376
valid_0's auc: 0.877147
[85]    valid_0's l2: 0.136994  valid_0's binary_error: 0.204177
valid_0's auc: 0.876872
[86]    valid_0's l2: 0.136599  valid_0's binary_error: 0.203681
valid_0's auc: 0.877042
[87]    valid_0's l2: 0.136185  valid_0's binary_error: 0.203148
valid_0's auc: 0.877254
[88]    valid_0's l2: 0.136209  valid_0's binary_error: 0.202957
valid_0's auc: 0.877159
```

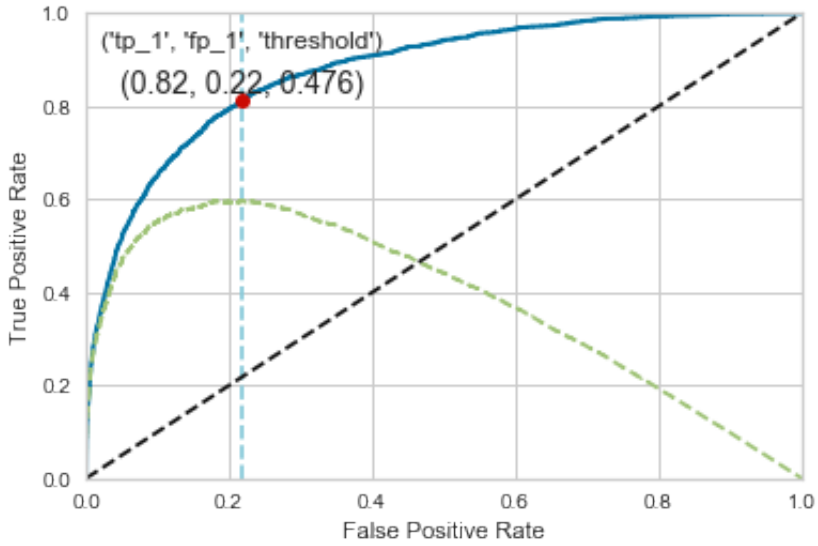
```
[89]    valid_0's l2: 0.1367    valid_0's binary_error: 0.203452
valid_0's auc: 0.876793
[90]    valid_0's l2: 0.136441  valid_0's binary_error: 0.203338
valid_0's auc: 0.876838
[91]    valid_0's l2: 0.136895  valid_0's binary_error: 0.204519
valid_0's auc: 0.876489
[92]    valid_0's l2: 0.136271  valid_0's binary_error: 0.203262
valid_0's auc: 0.876913
[93]    valid_0's l2: 0.136014  valid_0's binary_error: 0.202881
valid_0's auc: 0.87694
[94]    valid_0's l2: 0.135913  valid_0's binary_error: 0.202728
valid_0's auc: 0.876985
[95]    valid_0's l2: 0.136317  valid_0's binary_error: 0.204405
valid_0's auc: 0.876734
[96]    valid_0's l2: 0.136363  valid_0's binary_error: 0.204215
valid_0's auc: 0.876666
[97]    valid_0's l2: 0.136221  valid_0's binary_error: 0.204253
valid_0's auc: 0.876776
[98]    valid_0's l2: 0.136596  valid_0's binary_error: 0.204977
valid_0's auc: 0.87653
[99]    valid_0's l2: 0.136996  valid_0's binary_error: 0.206272
valid_0's auc: 0.876207
[100]   valid_0's l2: 0.136277  valid_0's binary_error: 0.204024
valid_0's auc: 0.876632
Did not meet early stopping. Best iteration is:
[94]    valid_0's l2: 0.135913  valid_0's binary_error: 0.202728
valid_0's auc: 0.876985
```

In [28]:
```
proba=gbm2.predict_proba(test_x,num_iteration=gbm.best_iteration)
proba2=proba[:,1]
train_proba=gbm2.predict_proba(train_x,num_iteration=gbm.best_itera
tion)
train_proba=train_proba[:,1]
```

chose threshold

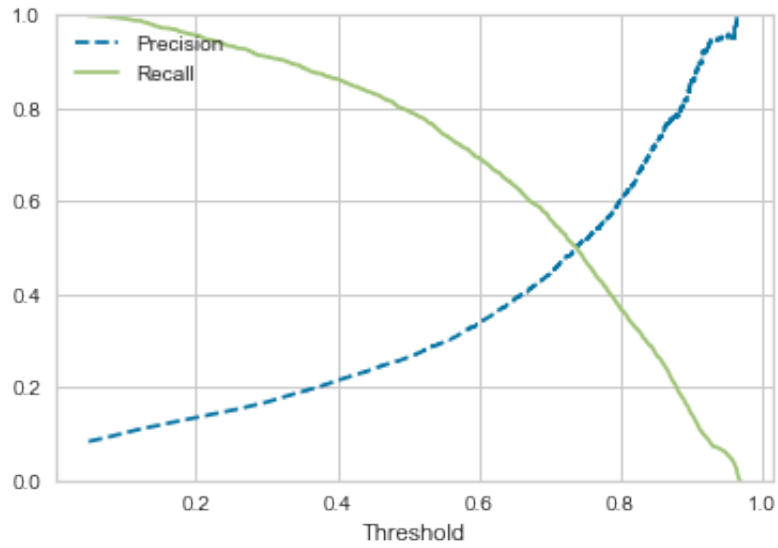
In [29]:
```
def plot_precision_recall_vs_threshold (precisions, recalls, thresh
olds) :
    plt.plot(thresholds, precisions[: -1 ],  "b--" , label= "Precis
ion" )
    plt.plot(thresholds, recalls[: -1 ],  "g-" , label= "Recall" )
    plt.xlabel( "Threshold" )
    plt.legend(loc= "upper left" )
    plt.ylim([ 0 ,  1 ])
    plt.show()

def plot_roc_curve (y_true,pre, label=None) :
    fpr, tpr, thresholds = roc_curve(y_true,pre)
    diff=tpr-fpr #array
    index=list(diff).index(max(tpr-fpr))
    threshold=thresholds[index]
    fp_1=fpr[index]
    tp_1=tpr[index]

    plt.plot(fpr, tpr, linewidth= 2 , label=label)
    plt.plot(fpr,diff,c='g',linestyle='dashed',label='ks')
    plt.plot([ 0 ,  1 ], [ 0 ,  1 ],  'k--' )
    plt.plot( fp_1,tp_1 ,'ro')
    plt.text(fp_1, tp_1, (round(tp_1,2)
                          , round(fp_1,2)
                          ,round(threshold,3)),
             ha='center', va='bottom', fontsize=14)
    plt.text(fp_1, tp_1+0.1, ('tp_1'
                          , 'fp_1'
                          ,'threshold'),
             ha='center', va='bottom', fontsize=12)

    plt.vlines(fp_1, 0, 1, colors = "c", linestyles = "dashed")
    plt.axis([ 0 ,  1 ,  0 ,  1 ])
    plt.xlabel( 'False Positive Rate' )
    plt.ylabel( 'True Positive Rate' )
```

In [30]:
```
plot_roc_curve (test_y,proba2, label=None)
```



In [31]:
```
precision, recall, thresholds = precision_recall_curve(test_y, prob
a2)
```

In [32]:
```
plot_precision_recall_vs_threshold (precision, recall, thresholds)
```



In [33]:
```
threshold = 0.476

def get_result(prob):
    result=[]
    for pred in prob:
        if pred > threshold:
            result.append(1)
        else:
            result.append(0)
    return result
```

In [34]:
```
train_result=get_result(train_proba)
result=get_result(proba2)
```

result for train

In [35]:
```
print(classification_report(train_y, train_result))
print('accuracy_score',accuracy_score(train_y, train_result))
print('roc:',roc_auc_score(train_y, train_result))
print('confusion_matrix:\n',confusion_matrix(train_y, train_result)
)
print('sensitivity:',roc_auc_score(train_y, train_result))
```

```
              precision    recall  f1-score   support

           0       0.89      0.80      0.85     37106
           1       0.82      0.90      0.86     37106

    accuracy                           0.85     74212
   macro avg       0.86      0.85      0.85     74212
weighted avg       0.86      0.85      0.85     74212

accuracy_score 0.8530695844337843
roc: 0.8530695844337842
confusion_matrix:
 [[29812  7294]
 [ 3610 33496]]
```

result for test

In [36]:
```
print(classification_report(test_y, result))
print('accuracy_score',accuracy_score(test_y, result))
print('roc:',roc_auc_score(test_y, result))
print('confusion_matrix:\n',confusion_matrix(test_y, result))
```

```
              precision    recall  f1-score   support

           0       0.98      0.78      0.87     24060
           1       0.25      0.82      0.39      2182

    accuracy                           0.78     26242
   macro avg       0.62      0.80      0.63     26242
weighted avg       0.92      0.78      0.83     26242

accuracy_score 0.7844676472829815
roc: 0.7984889403439157
confusion_matrix:
 [[18807  5253]
 [  403  1779]]
```