# BayesianOptimization

import lightgbm as lgb

from bayes\_opt import BayesianOptimization

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_boston

from sklearn.metrics import accuracy\_score,confusion\_matrix

import numpy as np

def lgb\_evaluate(numLeaves, maxDepth, scaleWeight, minChildWeight, subsample, colSam, learing\_rate):

reg=lgb.LGBMRegressor(num\_leaves=31, max\_depth= 2,scale\_pos\_weight= scaleWeight, min\_child\_weight= minChildWeight, subsample= 0.4, colsample\_bytree= 0.4, learning\_rate=0.05, n\_estimators=20, n\_jobs=multiprocessing.cpu\_count() )

# scores = cross\_val\_score(reg, train\_x, train\_y, cv=5, scoring='roc\_auc')

tscv = TimeSeriesSplit(n\_splits=5, test\_size=5, gap=5)

scores = cross\_val\_score(reg, train\_x, train\_y, cv=tscv, scoring='neg\_mean\_squared\_error')

return np.mean(scores)

def bayesOpt(train\_x, train\_y):

lgbBO = BayesianOptimization(lgb\_evaluate, {'learing\_rate': (0.01, 0.1),'numLeaves': (15, 30), 'maxDepth': (2, 90), 'scaleWeight': (1, 10000),'minChildWeight': (0.01, 70), 'subsample': (0.4, 1), 'colSam': (0.4, 1) })

lgbBO.maximize(init\_points=5, n\_iter=50)

print(lgbBO.res)

bayesOpt(train\_x, train\_y)