import numpy as np

import pandas as pd

from xgboost.sklearn import XGBClassifier

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

from sklearn.model\_selection import train\_test\_split,StratifiedKFold,train\_test\_split,GridSearchCV

from sklearn.metrics import accuracy\_score, confusion\_matrix, mean\_squared\_error,roc\_auc\_score

from xgboost import plot\_importance

from matplotlib import pyplot as plt

import xgboost as xgb

import warnings

warnings.filterwarnings('ignore')

#1 xgb\_model

#%%处理 x1

def x1process(x):

x1 = x.copy()

# A3 无缺失值

x1['A3'] = x1['A3'].str[1:]

# A8 有缺失值

x1['A8'] = x1['A8'].str[2:]

# A13

x1['A13'] = x1['A13'].str[1:]

# A14 无缺失值

x1['A14'] = x1['A14'].apply(lambda x : '7' if x=='M' else '8' if x=='N' else x)

# A15 有缺失值

x1['A15'] = x1['A15'].str[2:]

# A17 无缺失值/365

x1['A17'] = x1['A17'].str[:4].astype(float)/365

x1=x1[['A3','A8', 'A13', 'A14', 'A15', 'A17']]

return x1

x1\_train = pd.read\_csv('X1\_train.csv',header=[0], index\_col=0)

x1\_train = x1process(x1\_train)

x1\_test = pd.read\_csv('X1\_test.csv',header=[0], index\_col=0)

x1\_test = x1process(x1\_test)

x1\_col = ['A3','A8', 'A13', 'A14', 'A15', 'A17']

#%%处理 x2

def x2process(x, x2\_index):

x2 = x.copy()

x2['B1'] = x2['B1'].str[0:4].astype(float)-3468

tmp\_dict = {'A':1,

'B':2,

'C':3,

'D':4,

'E':5,

'F':6,

'G':7}

x2['B3'] = x2['B3'].map(tmp\_dict)

x2['B4'] = x2['B4'].str[1:].astype(float)

x2['B6'] = x2['B6'].str[1:]

x2['B8'] = x2['B8'].str[0:4].astype(float)

x2['B8'].fillna(3834,inplace=True)

x2['B8'] = x2['B8']-3469

x2['B15'] = x2['B15'].str[1:]

x2['B15'] = x2['B15'].astype(float)

x2['B16'] = x2['B16'].str[1:]

x2['B16'] = x2['B16'].astype(float)

x2['B19'] = x2['B19'].str[1:].astype(float)

x2['B20'] = x2['B8']-x2['B2']

'''

x2['B21'] = x2['B8']-x2['B1']

x2\_columns = ['B1','B2','B5','B7','B9','B10',

'B11','B13','B14','B17','B20','B21']

'''

x2\_columns = ['B1','B2','B5','B7','B8','B9','B10',

'B11','B13','B14','B17','B20']

x2\_mean = pd.DataFrame(data=np.zeros((x2\_index.shape[0],len(x2\_columns))),

index=x2\_index,columns=x2\_columns)

x2\_min = pd.DataFrame(data=np.zeros((x2\_index.shape[0],len(x2\_columns))),

index=x2\_index,columns=x2\_columns)

x2\_max = pd.DataFrame(data=np.zeros((x2\_index.shape[0],len(x2\_columns))),

index=x2\_index,columns=x2\_columns)

x2\_range = pd.DataFrame(data=np.zeros((x2\_index.shape[0],len(x2\_columns))),

index=x2\_index,columns=x2\_columns)

x2\_buy\_cnt = pd.DataFrame(data=np.zeros((x2\_index.shape[0],len(x2\_columns))),

index=x2\_index,columns=x2\_columns)

x2\_sum = pd.DataFrame(data=np.zeros((x2\_index.shape[0],len(x2\_columns))),

index=x2\_index,columns=x2\_columns)

x2\_std = pd.DataFrame(data=np.zeros((x2\_index.shape[0],len(x2\_columns))),

index=x2\_index,columns=x2\_columns)

#需要按照月份统计count(\*)

#最值：1、2、5、7、9、10、11、13、14、17、20、21

#方差：懒得搞了

#极差：1、2、5、7、9、10、11、13、14、17、20、21

#均值：1、2、5、7、9、10、11、13、14、17、20、21

#大招-求和：1、2、5、7、9、10、11、13、14、17、20、21

grouped = x2.groupby('客户编号')

for c in x2\_columns:

x2\_mean[c] = grouped[c].mean().to\_frame()

x2\_min[c] = grouped[c].min().to\_frame()

x2\_max[c] = grouped[c].max().to\_frame()

x2\_range[c] = x2\_max[c]-x2\_min[c]

x2\_sum[c] = grouped[c].sum().to\_frame()

#x2\_std[c] = grouped[c].std().to\_frame()

x2\_mean.columns =[i+'mean' for i in x2\_columns]

x2\_min.columns = [i+'min' for i in x2\_columns]

x2\_max.columns = [i+'max' for i in x2\_columns]

x2\_range.columns=[i+'range' for i in x2\_columns]

x2\_buy\_cnt.columns=[i+'cnt' for i in x2\_columns]

x2\_sum.columns=[i+'sum' for i in x2\_columns]

#x2\_std.columns=[i+'std' for i in x2\_columns]

new\_x2 = pd.merge(x2\_mean,x2\_min,on='客户编号')

new\_x2 = pd.merge(new\_x2,x2\_max,on='客户编号')

new\_x2 = pd.merge(new\_x2,x2\_range,on='客户编号')

new\_x2 = pd.merge(new\_x2,x2\_buy\_cnt,on='客户编号')

new\_x2 = pd.merge(new\_x2,x2\_sum,on='客户编号')

#new\_x2 = pd.merge(new\_x2,x2\_std,on='客户编号')

return new\_x2

x2\_train = pd.read\_csv('X2\_train.csv',header=[0], index\_col=0).dropna(how='all',axis=1)

new\_x2\_train = x2process(x2\_train, x1\_train.index)

x2\_test = pd.read\_csv('X2\_test.csv',header=[0], index\_col=0)

new\_x2\_test = x2process(x2\_test, x1\_test.index)

#%% 处理 x3

def x3process(x3):

# 10 12 13 14 18 20 21 22 27 28 29 30 31 32 34 36 39 41 45 47 49

#x3['C2'] = x3['C2'].fillna(x3['C2'].mean())

x3 = x3[['C2','C3', 'C4', 'C5','C8', 'C10', 'C13', 'C14', 'C16', 'C18', 'C20', 'C29',

'C30', 'C32', 'C34', 'C36', 'C43', 'C46', 'C47']]

temp = list(x3['C2'])

for i in range(0, len(temp)):

if temp[i] == -1.2475837768427405:

temp[i] = 1

else: temp[i] = 0

temp = pd.Series(temp)

temp.index=x3.index

x3['C2'] = temp.astype(str).astype(float)

for i in range(x3.shape[1]):

x3.iloc[:,i] = x3.iloc[:,i].fillna(x3.iloc[:,i].mean())

return x3

x3\_train = pd.read\_csv('X3\_train.csv',header=[0], index\_col=0)

new\_x3\_train = x3process(x3\_train)

x3\_test = pd.read\_csv('X3\_test.csv',header=[0], index\_col=0)

new\_x3\_test = x3process(x3\_test)

#%%

# x\_train = pd.merge(new\_x1\_train,new\_x2\_train,on='客户编号')

# x\_train = pd.merge(x\_train,new\_x3\_train,on='客户编号')

cols = []

for i in ['B2','B7','B9']:

for j in ['max','range','sum','mean']:

cols.append(i+j)

'''

for i in ['B1']: 这个如果删了反而下降了 惹不起

for j in ['sum','mean']:

cols.append(i+j)

'''

x\_train = pd.merge(new\_x2\_train, new\_x3\_train,on='客户编号')

x\_train.drop(columns=cols,inplace=True)

# x\_test = pd.merge(new\_x1\_test,new\_x2\_test,on='客户编号')

# x\_test = pd.merge(x\_test,new\_x3\_test,on='客户编号')

x\_test = pd.merge(new\_x2\_test, new\_x3\_test,on='客户编号')

x\_test.drop(columns=cols,inplace=True)

y\_train = pd.read\_csv('y\_train.csv',header=[0], index\_col=0)['复购频率']

ys\_train = y\_train.copy()

#%% 1 回归树

for i in ys\_train.index:

if ys\_train[i] == 2:

ys\_train[i] = 3

xgb\_model = xgb.XGBRegressor(n\_estimators=1000,learning\_rate=0.01, max\_depth=2).fit(x\_train, ys\_train)

y\_test\_p = xgb\_model.predict(x\_test)

for i in range(len(y\_test\_p)):

if y\_test\_p[i] > 1.3:

y\_test\_p[i] =2

elif y\_test\_p[i] > 0.4:

y\_test\_p[i] = 1

else:

y\_test\_p[i] = 0

# 输出结果

result = pd.DataFrame(y\_test\_p,columns=['复购频率'])

result.to\_excel(r'output/result.xlsx',index=0)

result.head()