import pandas as pd

import numpy as np

from sklearn.ensemble import BaggingRegressor

from sklearn.linear\_model import LinearRegression

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

from sklearn import linear\_model

import xgboost as xgb

from sklearn.metrics import r2\_score

from sklearn.metrics import explained\_variance\_score

def get\_OHE():

data = pd.read\_csv('ct\_temp.csv')

data = pd.read\_csv('mri\_1.csv')

print(data.columns)

data['diff'] = data['total']

data = data.drop(['total'], axis=1)

# 排除極端值

data = data[data['diff'] > 60]

data = data[data['diff'] < 90 \* 60]

data = data.dropna(how='any')

temp = data.copy()

# OHE:將類別型變數區分出來

cate = ['ITEM', 'ORDERDR', 'IO', 'DEPT', 'SEX', 'BED/chair', 'POS']

col = cate + ['AGE', 'MD\_NO', 'diff']

temp = temp[col]

# 轉成OHE

for c in cate:

# 把小數點去掉 欄位名稱比較漂亮lol

if temp[c].dtype != 'object':

temp[c] = temp[c].astype(int)

value = pd.unique(temp[c])

for label in value:

temp[str(c) + '\_' + str(label)] = temp[c].apply(lambda d: 1 if d == label else 0)

# break #只測試第一個類別

print(c, 'is finished')

temp = temp.drop(cate, axis=1)

temp.to\_csv('mri\_ohe.csv')

# temp.to\_csv('ct\_ohe.csv')

def get\_PNO():

data = pd.read\_csv('ct\_temp.csv')

data = data.groupby(['DATE', 'MD\_NO']).count().reset\_index()

print(data.groupby('MD\_NO')['Unnamed: 0'].describe())

def mape(true, pre):

diff = np.abs(np.array(true) - np.array(pre))

return round(np.mean(diff \* 100 / true), 2)

def shoot(true, pre):

diff = np.abs(np.array(true) - np.array(pre))

diff = pd.DataFrame({'diff': diff})

diff['diff'] = diff['diff'].apply(lambda s: 1 if s < 400 else 0)

return np.mean(diff['diff']) \* 100

def bagging():

data = pd.read\_csv('MR\_meanencoding.csv', index\_col=0)

data = data.drop("PNO",axis=1)

print(1)

data = data[data['AGE'] <= 120]

data = data[data['AGE'] >= 1]

print(2)

data = data[data['total'] < 90 \* 60]

data = data[data['total'] > 60]

print(3)

# 標準化

# min = data['AGE'].min()

# max = data['AGE'].max()

# data['AGE'] = (data['AGE'] - min) / (max - min)

# mlist = pd.unique(data['PLACE\_n'])

# print(mlist)

# 第五台沒資料

for m in range(1):

pno = {405108: 50, 405568: 23, 405984: 11, 406750: 83}

# pno = {405108: 54, 405568: 23, 405984: 12, 406750: 89}

temp = data.copy()

print(temp.columns)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

temp.drop('total', axis=1), temp['total'], test\_size=0.3, random\_state=42

)

def find\_alpha(X\_train, y\_train):

reg = linear\_model.RidgeCV(alphas=np.logspace(-6, 6, 13))

reg.fit(X\_train, y\_train)

print(m, ':', reg.alpha\_)

# find\_alpha(X\_train, y\_train)

alpha = {405108: 100, 405568: 10, 405984: 100, 406750: 10}

bagging = BaggingRegressor(base\_estimator=linear\_model.LinearRegression(),

max\_samples=.1, max\_features=1)

bagging.fit(X\_train, y\_train)

# print(m)

print('bagging\_linear:', mape(y\_test, bagging.predict(X\_test)))

# print('bagging\_linear:', mape(y\_test, bagging.predict(X\_test)), r2\_score(y\_test, bagging.predict(X\_test)))

# print(shoot(y\_test, bagging.predict(X\_test)))

bagging = BaggingRegressor(base\_estimator=linear\_model.Ridge(alpha=.5),

max\_samples=.1, max\_features=1)

bagging.fit(X\_train, y\_train)

print('bagging\_Ridge:', mape(y\_test, bagging.predict(X\_test)))

# print('bagging\_Ridge:', mape(y\_test, bagging.predict(X\_test)), r2\_score(y\_test, bagging.predict(X\_test)))

# print(shoot(y\_test, bagging.predict(X\_test)))

bagging = BaggingRegressor(base\_estimator=linear\_model.Lasso(alpha=0.5),

max\_samples=.1, max\_features=1)

bagging.fit(X\_train, y\_train)

print('bagging\_Lasso:', mape(y\_test, bagging.predict(X\_test)))

# print('bagging\_Lasso:', mape(y\_test, bagging.predict(X\_test)), r2\_score(y\_test, bagging.predict(X\_test)))

# print(shoot(y\_test, bagging.predict(X\_test)))

bagging = BaggingRegressor(max\_samples=.1, max\_features=1)

bagging.fit(X\_train, y\_train)

print('bagging\_decision:', mape(y\_test, bagging.predict(X\_test)))

# print('bagging\_decision:', mape(y\_test, bagging.predict(X\_test)), r2\_score(y\_test, bagging.predict(X\_test)))

# print(shoot(y\_test, bagging.predict(X\_test)))

reg = linear\_model.LinearRegression()

reg.fit(X\_train, y\_train)

print('linear', mape(y\_test, reg.predict(X\_test)))

# print('linear', mape(y\_test, reg.predict(X\_test)), r2\_score(y\_test, bagging.predict(X\_test)))

# print(shoot(y\_test, reg.predict(X\_test)))

reg = linear\_model.Ridge(alpha=.5)

reg.fit(X\_train, y\_train)

print('Ridge', mape(y\_test, reg.predict(X\_test)))

# print('Ridge', mape(y\_test, reg.predict(X\_test)), r2\_score(y\_test, bagging.predict(X\_test)))

# print(shoot(y\_test, reg.predict(X\_test)))

reg = linear\_model.Lasso(alpha=0.5)

reg.fit(X\_train, y\_train)

print('Lasso', mape(y\_test, reg.predict(X\_test)))

# print('Lasso', mape(y\_test, reg.predict(X\_test)), r2\_score(y\_test, bagging.predict(X\_test)))

# print(shoot(y\_test, reg.predict(X\_test)))

# break #只做一台

def result():

pass

# bagging default

"""

#每個模型都抓50人

406750 : 192.31

405108 : 60.58

405984 : 72.17

405568 : 91.05

依照個別平均數計算

83 406750 : 179.74

50 405108 : 55.85

11 405984 : 71.48

23 405568 : 95.19

#中位數

89 406750 : 188.62

54 405108 : 56.88

12 405984 : 72.65

23 405568 : 89.66

alpha值

406750 : 10.0

405108 : 100.0

405984 : 100.0

405568 : 10.0

Ridge 調整alpha值使用50

89 406750 : 172.64

54 405108 : 59.89

12 405984 : 65.07

23 405568 : 94.14

Ridge 調整alpha值使用個別平均

89 406750 : 173.11

54 405108 : 58.29

12 405984 : 64.84

23 405568 : 87.89

Lasso 0.1,50pno

89 406750 : 177.83

54 405108 : 57.67

12 405984 : 69.47

23 405568 : 89.32

Lasso 0.1,median pno

89 406750 : 171.18

54 405108 : 59.43

12 405984 : 67.73

23 405568 : 95.9

Lasso 0.1,mean pno

83 406750 : 177.26

50 405108 : 58.97

11 405984 : 77.91

23 405568 : 93.49

"""

def xgboost():

data = pd.read\_csv('ct\_ohe.csv')

data = data[data['AGE'] <= 120]

data = data[data['AGE'] >= 1]

# 標準化

min = data['AGE'].min()

max = data['AGE'].max()

data['AGE'] = (data['AGE'] - min) / (max - min)

# print(data['AGE'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

data.drop('total', axis=1), data['total'], test\_size=0.3, random\_state=42

)

# xgb

dtrain = xgb.DMatrix(X\_train)

y\_train = xgb.DMatrix(y\_train)

dtest = xgb.DMatrix(X\_test)

# print(X\_train)

#

param = {'max\_depth': 2, 'eta': 1, 'objective': 'binary:logistic'}

param['nthread'] = 4

param['eval\_metric'] = 'auc'

evallist = [(dtest, 'eval'), (dtrain, 'train')]

num\_round = 10

# bst = xgb.train(param, dtrain, num\_round, evallist)

gbm = xgb.XGBRegressor(objective="reg:linear", n\_estimators=50, learning\_rate=0.05).fit(dtrain, y\_train)

print(mape(y\_test, gbm.predict(dtest)))

def get\_mri():

data = pd.read\_csv('mri.csv', index\_col=0)

print(data.columns)

data['diff'] = data['total']

cate = ['BED/chair', 'ITEM', 'ORDERDR', 'MD\_NO',

'IO', 'DEPT', 'SEX', 'AGE', 'POS', 'diff']

date = data[cate]

# print(data['diff'])

data.to\_csv('mri\_1.csv')

if \_\_name\_\_ == '\_\_main\_\_':

# get\_OHE() #建立OHE的資料

# get\_PNO() #計算不同機台的平均人數

# get\_mri()

bagging()

# data = pd.read\_csv('mri\_ohe.csv', index\_col=0)

# print(data.columns)

# print(data['diff'])

# data['pre'] = 2765.0

# print(mape(data['diff'], data['pre']))

# print(data['diff'].describe())

# xgboost(